



EMERGING 2025

The Seventeenth International Conference on Emerging Networks and Systems
Intelligence

ISBN: 978-1-68558-292-0

September 28th - October 2nd, 2025

Lisbon, Portugal

EMERGING 2025 Editors

Nader Mir, San Jose State University, USA

EMERGING 2025

Forward

The Seventeenth International Conference on Emerging Networks and Systems Intelligence (EMERGING 2025), held between September 28th, 2025, and October 2nd, 2025, in Lisbon, Portugal, continued a series of international events to present and evaluate the advances in emerging solutions for next-generation architectures, devices, and communications protocols. Particular focus was aimed at optimization, quality, discovery, protection, and user profile requirements supported by special approaches such as network coding, configurable protocols, context-aware optimization, ambient systems, anomaly discovery, and adaptive mechanisms.

Next generation large, distributed networks and systems require substantial reconsideration of exiting ‘de facto’ approaches and mechanisms to sustain an increasing demand on speed, scale, bandwidth, topology and flow changes, user complex behavior, security threats, and service and user ubiquity. As a result, growing research and industrial forces are focusing on new approaches for advanced communications considering new devices and protocols, advanced discovery mechanisms, and programmability techniques to express, measure and control the service quality, security, environmental and user requirements.

We take here the opportunity to warmly thank all the members of the EMERGING 2025 technical program committee, as well as all the reviewers. The creation of such a high-quality conference program would not have been possible without their involvement. We also kindly thank all the authors who dedicated much of their time and effort to contribute to EMERGING 2025. We truly believe that, thanks to all these efforts, the final conference program consisted of top-quality contributions. We also thank the members of the EMERGING 2025 organizing committee for their help in handling the logistics of this event.

We hope that EMERGING 2025 was a successful international forum for the exchange of ideas and results between academia and industry for the promotion of progress in the field of emerging networks and systems intelligence.

EMERGING 2025 Chairs

EMERGING 2025 Steering Committee

Euthimios (Thimios) Panagos, Perspecta Labs, USA

Raj Jain, Washington University in St. Louis, USA

Dimitris Kanellopoulos, University of Patras, Greece

Robert Bestak, Czech Technical University in Prague, Czech Republic

Jason Zurawski, Lawrence Berkeley National Laboratory / Energy Sciences Network (ESnet), USA

Emilio Insfran, Universitat Politècnica de València, Spain

Linda R. Elliott, Army Research Laboratory, USA

EMERGING 2025 Publicity Chairs

Laura Garcia, Universidad Politècnica de Cartagena, Spain

Lorena Parra Boronat, Universidad Politècnica de Madrid, Spain

EMERGING 2025 Committee

EMERGING 2025 Steering Committee

Euthimios (Thimios) Panagos, Perspecta Labs, USA
Raj Jain, Washington University in St. Louis, USA
Dimitris Kanellopoulos, University of Patras, Greece
Robert Bestak, Czech Technical University in Prague, Czech Republic
Jason Zurawski, Lawrence Berkeley National Laboratory / Energy Sciences Network (ESnet), USA
Emilio Insfran, Universitat Politècnica de València, Spain
Linda R. Elliott, Army Research Laboratory, USA

EMERGING 2025 Publicity Chairs

Laura Garcia, Universidad Politécnica de Cartagena, Spain
Lorena Parra Boronat, Universidad Politécnica de Madrid, Spain

EMERGING 2025 Technical Program Committee

Andrea F. Abate, University of Salerno, Italy
Dorel Aiordachioaie, Dunarea de Jos University of Galati, Romania
Mohammed GH. AL Zamil, Yarmouk University, Jordan
Firkhan Ali Bin Hamid Ali, Universiti Tun Hussein Onn Malaysia, Malaysia
Nik Bessis, Edge Hill University, UK
Robert Bestak, Czech Technical University in Prague, Czech Republic
Adil Bilal, University of Canterbury, New Zealand
Tetiana Biloborodova, G.E. Pukhov Institute for Modelling in Energy Engineering - National Academy of Sciences of Ukraine, Kyiv, Ukraine
Lucas Botoni de Souza, Federal University of Technology - Paraná, Brazil
Alessandro Casavola, University of Calabria, Italy
Graziana Cavone, Polytechnic of Bari, Italy
Chin-Chen Chang, Feng Chia University, Taiwan
Hyung Jae (Chris) Chang, Troy University, USA
Hongxu Chen, University of Technology, Sydney
Arun Das, Visa Inc., USA
David de Andrés, Universitat Politècnica de València, Spain
Renato De Leone, University of Camerino, Italy
Ramadan Elaïess, University of Benghazi, Libya
Linda R. Elliott, Army Research Laboratory, USA
Stefka Fidanova, Institute of Information and Communication Technologies - Bulgarian Academy of Sciences, Bulgaria
Nuno Gonçalves Rodrigues, Polytechnic Institute of Bragança, Portugal
Chunhui Guo, San Diego State University, USA
Mohd Helmy Abd Wahab, Universiti Tun Hussein Onn Malaysia, Malaysia
Seyed Mohsen Hosseini, Free University of Bozen-Bolzano, Italy

Sergio Ilarri, University of Zaragoza, Spain
Oleg Illiashenko, National Aerospace University "KhAI", Ukraine
Emilio Insfran, Universitat Politècnica de Valencia, Spain
Raj Jain, Washington University in St. Louis, USA
Jin-Hwan Jeong, SK telecom, Republic of Korea
Dimitris Kanellopoulos, University of Patras, Greece
Ah-Lian Kor, Leeds Beckett University, UK
Igor Kotsiuba, PIMEE NAS of Ukraine, Ukraine
Marcos Levano, Catholic University of Temuco, Chile
Vitaly Levashenko, University of Zilina, Slovakia
Keqian Li, Yahoo Research, USA
YanJun Liu, Feng Chia University, Taiwan
Zoubir Mammeri, IRT - Paul Sabatier University, Toulouse, France
Christopher Mansour, Mercyhurst University, USA
Nada Matta, University of Technology of Troyes, France
Andrea Michienzi, University of Pisa, Italy
Nader Mir, San Jose State University, USA
Ioannis Moscholios, University of Peloponnese, Greece
Chika Oshima, Saga University, Japan
Euthimios (Thimios) Panagos, Perspecta Labs, USA
Isidoros Perikos, University of Patras, Greece
Przemyslaw (Przemek) Pocheć, University of New Brunswick, Canada
Chuan Qin, University of Shanghai for Science and Technology, China
Danda B. Rawat, Howard University, USA
Muhammad Hassan Raza, Dalhousie University, Halifax, Canada
Federica Rollo, University of Modena and Reggio Emilia, Italy
Francesco Rundo, STMicroelectronics Srl - ADG Central R&D, Italy
Khair Eddin Sabri, The University of Jordan, Jordan
Christian Schulz, Heidelberg University, Germany
Ivan Serina, University of Brescia, Italy
Hiroyasu Shimauchi, Future University Hakodate, Japan
Masakazu Soshi, Hiroshima City University, Japan
Olarik Surinta, Mahasarakham University, Thailand
Bashir Tenuche, Kogi State University, Anyigba, Nigeria
Hui Tian, National Huaqiao University, China
Joseph G. Vella, University of Malta, Malta
Antonio Viridis, University of Pisa, Italy
Wuyi Yue, Konan University, Japan
Daqing Yun, Harrisburg University, USA
Elena Zaitseva, University of Zilina, Slovakia
Jie Zhang, Amazon AWS, USA
Minfan Zhang, Aviva, Canada
Sotirios Ziaouras, New Jersey Institute of Technology, USA
Jason Zurawski, Lawrence Berkeley National Laboratory / Energy Sciences Network (ESnet), USA

Copyright Information

For your reference, this is the text governing the copyright release for material published by IARIA.

The copyright release is a transfer of publication rights, which allows IARIA and its partners to drive the dissemination of the published material. This allows IARIA to give articles increased visibility via distribution, inclusion in libraries, and arrangements for submission to indexes.

I, the undersigned, declare that the article is original, and that I represent the authors of this article in the copyright release matters. If this work has been done as work-for-hire, I have obtained all necessary clearances to execute a copyright release. I hereby irrevocably transfer exclusive copyright for this material to IARIA. I give IARIA permission to reproduce the work in any media format such as, but not limited to, print, digital, or electronic. I give IARIA permission to distribute the materials without restriction to any institutions or individuals. I give IARIA permission to submit the work for inclusion in article repositories as IARIA sees fit.

I, the undersigned, declare that to the best of my knowledge, the article does not contain libelous or otherwise unlawful contents or invading the right of privacy or infringing on a proprietary right.

Following the copyright release, any circulated version of the article must bear the copyright notice and any header and footer information that IARIA applies to the published article.

IARIA grants royalty-free permission to the authors to disseminate the work, under the above provisions, for any academic, commercial, or industrial use. IARIA grants royalty-free permission to any individuals or institutions to make the article available electronically, online, or in print.

IARIA acknowledges that rights to any algorithm, process, procedure, apparatus, or articles of manufacture remain with the authors and their employers.

I, the undersigned, understand that IARIA will not be liable, in contract, tort (including, without limitation, negligence), pre-contract or other representations (other than fraudulent misrepresentations) or otherwise in connection with the publication of my work.

Exception to the above is made for work-for-hire performed while employed by the government. In that case, copyright to the material remains with the said government. The rightful owners (authors and government entity) grant unlimited and unrestricted permission to IARIA, IARIA's contractors, and IARIA's partners to further distribute the work.

Table of Contents

Latency Optimization in IoT Networks Using SD-WAN Edge Computing <i>Rashmi Vaidya and Nader Mir</i>	1
Intelligent Pest Identification for Precision Agriculture using Deep Learning <i>Anika Bhat and Atul Dubey</i>	6
A Neural Approach to Ray Tracing for Realistic Wireless Channel Simulation in Indoor and Urban Scenarios <i>Francisco Javier Somolinos-Simon, Adina Murg, Hanli Liu, Carlos Javier Hellin, Josefa Gomez, and Abdelhamid Tayebi</i>	13
A Lightweight Hybrid AI Framework for Cataract Detection Using Fundus Images: Real-World Evaluation on Clinical Data <i>Ishaan Kunwar</i>	20

Latency Optimization in IoT Networks Using SD-WAN Edge Computing

Rashmi S. Vaidya

Department of Electrical Engineering
San Jose State University in California
San Jose, CA, 95195, U.S.A
email: rashmi.s.vaidya@sjsu.edu

Nader F. Mir

Department of Electrical Engineering
San Jose State University in California
San Jose, CA, 95195, U.S.A
email: nader.mir@sjsu.edu

Abstract — This paper aims to reduce latency in Internet of Things (IoT) sensor networks using Software-Defined Wide Area Network (SD-WAN) edge computing. The motivation comes from the strong necessity of rapid decision-making in the emerging IoT applications such as smart cities and autonomous vehicles. In the real communications world, the processing speed of devices and sensor nodes in an IoT network is slow as nodes and devices have limited power and processing resources, while cloud-based methods are time-consuming to access distant servers. This paper uses a custom SD-WAN controller to coordinate the task allocation to nearby edge servers to lower the response time as well as data processing delays. Algorithms such as the ones for traffic prioritization based on QoS requirements, edge computing resource allocation, and edge caching techniques are also deployed to achieve the goal. Software implementation and simulations are used to quantify the latency reduction achieved. Through SD-WAN and edge computing, the paper focuses on techniques for reducing the IoT network latency to improve IoT operations' safety and efficiency.

Keywords—IoT Networks; Sensors; Latency Optimization; Software-Defined Wide Area Networking (SD-WAN); Traffic Prioritization; Quality of Service (QoS); Cloud Computing; Edge Computing; Edge Caching; Smart Cities; Autonomous Vehicles.

I. INTRODUCTION

Internet of Things (IoT) networks consist of interconnected sensor nodes that collect and transmit data over the Internet [1]. These networks are key to the ecosystem, enabling various industry applications such as smart homes, healthcare, industrial automation, and agriculture by integrating data collection, transmission, processing, storage, and user interaction.

Latency, defined as the round-trip delay in data transmission, is a critical factor in IoT networks where real-time responses are essential. IoT applications engage large amounts of data transfers requiring response rates and efficiency. For instance, autonomous cars need instant decisions for safety, and any data delay could cause accidents. Similarly, quick responses are crucial for smart city applications like traffic management and emergency response at the time of a natural disaster. Latency may occur due to various factors like propagation times, transmission delays, processing delays, and queuing delays [2]. Factors like distance, network congestion, transmission medium, and hardware efficiency may also affect latency.

Reducing latency in IoT networks is crucial, however, IoT devices often use small, low-cost sensors with limited memory and power, and poor network connectivity. Computing tasks can be slow or impossible on IoT devices. Cloud computing

on the other hand can help, but the fact that servers might be in distant locations may add wait time, which can be hectic for IoT applications needing low delays [3].

Software-Defined Wide-Area Network (SD-WAN) is a technology that uses defined software to optimize wide area network connections resulting in more flexibility and control over traditional WAN architectures. With SD-WAN, certain network management features allow organizations to efficiently connect users across multiple locations. Similar to the software-defined networking in data centers, as long as configuration messages are supported by all the network hardware device makers, SD-WAN decouples the networking hardware from its control system and creates a central control plane in the WAN. This concept is like how software-defined networking [2] implements virtualization technology resulting in significant simplification in managing a network. The most promising feature of SD-WAN is its ability to construct higher-performance software-defined based networks. SD-WAN creates an environment through which a wide area network uses software-defined networking control for communicating over the Internet using overlay tunnels which are encrypted when destined for internal organization locations.

The current trend of designing IoT networks is traditionally concentrated on connecting IoT devices to wide area networks [4]. We try in this paper to explore certain strategies including the deployment of SD-WAN to allow centralized control of the network, enabling dynamic adjustments to traffic flow. This deployment provides easy programmability for efficient network management and automates network behavior through software applications. We also demonstrate the deployment of QoS-based traffic prioritization by prioritizing critical IoT data over less time-sensitive information, where the Quality of Service (QoS) [5][6] ensures that high-priority tasks, like emergency alerts are transmitted with minimal delay. Additionally, we apply edge computing [7] in our study where processing data is placed closer to the sensors, at the edge of the network, to minimize the need to send data to distant cloud servers. This reduces the round-trip time and enables quicker decision-making for time-sensitive applications. Finally, we run our study in this paper with the use of edge caching [8][9] through which frequently accessed data at the edge helps reduce latency by minimizing the need to retrieve data from the cloud, speeding up access times for repeated requests.

The rest of this paper is organized as follows. In Section II, we present the details of a proposed architecture for IoT networks that optimizes the latency. In Section III, we present

detailed results of our analysis, and in Section IV, a conclusive statement is presented.

II. PROPOSED ARCHITECTURE

The proposed architecture uses SD-WAN to enhance the latency performance of IoT sensor networks. This can be done by implementing a network topology with a central SD-WAN controller, OpenFlow SD-WAN switches, QoS based traffic prioritization, edge servers and edge cache, as shown in Figure 1. The key components of the architecture are SD-WAN controller to manage the traffic flow and implement adaptive strategies, and SD-WAN switches that perform switching and forwarding in layers 2 and 3. The SD-WAN controller optionally uses Ryu software [10]. Ryu is an open, software-defined networking controller, written in Python and is supported and used in cloud data centers. In this networking set-up, IoT sensor devices send data of varying rates and sizes while edge servers and caches preprocess data and store frequently accessed information.

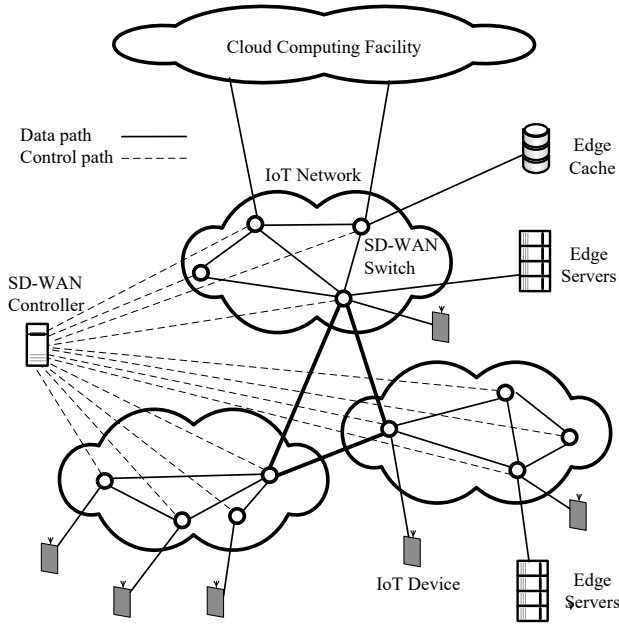


Figure 1. The setup for the IoT network with SD-WAN features.

For implementing effective traffic management, we utilize the Type of Service (ToS) byte existing in the IP packet header, as shown in Figure 2. The ToS field is used for enforcing Quality of Service (QoS) and prioritizing packets. The field uses its first 6-bits as Differentiated Services Code Point (DSCP) to classify traffic based on better QoS. The remaining two bits of ToS is called Explicit Congestion Notification (ECN) which is used for signaling a network for congestion. Based on this available feature, we classify packets from IoT devices into three types; each assigned a specific DSCP value to facilitate QoS prioritization. The SD-WAN controller uses DSCP values to prioritize certain traffic based on its importance, ensuring that real-time data receives preferential treatment over less critical traffic. Also, to

implement QoS-based prioritization, each SD-WAN-enabled switch or router maintains three separate queues.

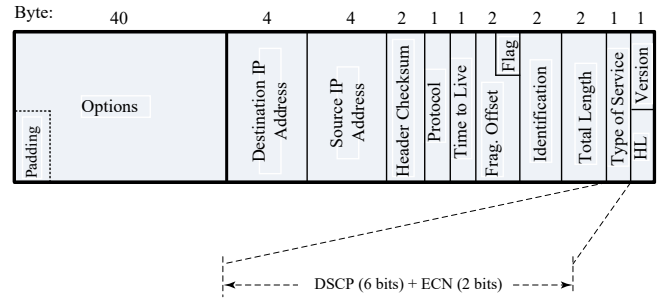


Figure 2. DSCP fields in IP header for QoS enforcement.

The following DSCP values and queue types are used in this paper to differentiate between three types of traffic: **EF 46**: high-priority time-sensitive traffic, such as real-time sensor data. The traffic is marked as Expedited Forwarding (EF) with a DSCP value of 46 (binary 101110), guaranteeing it is processed ahead of other traffic. The High-Priority Queue is used for this real-time traffic ensuring minimal latency. **AF31 26**: medium-priority routine sensor data, such as periodic status updates. The traffic is marked as Assured Forwarding (AF) with a DSCP value of 26 (binary 011010), providing medium-level priority. This queue is processed only when the high-priority queue is empty. **CS1-8**: low-priority non-urgent data, such as software updates or system logs, which do not require immediate processing and can tolerate longer delays. The traffic is assigned a low priority DSCP value, typically CS1 with a value of 8 (binary 001000), processed only after higher-priority traffic. For this traffic (CS1-8 or unmarked), a low priority queue is used and is processed only when the high and medium-priority queues are empty, ensuring real-time traffic is not delayed.

For the analysis of our network set-up, we utilized several software tools and platforms such as Virtual Machine, Mininet, Ryu controller, Python test scripts, Wireshark, iperf, and ping tools. The virtual machine setup includes VMware Fusion running Ubuntu OS Desktop 22.04 LTS “Jammy Jellyfish” Daily Build for Arm64 architecture on macOS (Apple ATM M2 silicon), with 4 GB of RAM and 2 CPU cores. The VM uses the same network adapter as the host OS. Mininet serves as the primary tool for network emulation, capable of creating both traditional non-SD-WAN- and SD-WAN-enabled IoT networks with edge computing capabilities. Although Mininet provides its own controller by default, this project uses the Ryu SD-WAN controller. The SD-WAN controller using the Ryu controller has been installed within the Ubuntu VM and serves as the central control unit for the SD-WAN-enabled network. It enables dynamic network management and policy implementation. Different network topologies are defined using Python scripts that run Mininet and instantiate hosts, switches, and routers. Various tests, such as the QoS test, Edge Server offloading test, and Edge Cache test, were also implemented using Python scripts. The QoS test requires C code to generate traffic of different sizes and QoS values. These test scripts save latency values into CSV files, which are later imported

by another set of Python scripts to generate graphs from the results.

The network simulation also uses Wireshark to capture and analyze network traffic within the Mininet environment. It is used for verifying QoS policies, monitoring data flow, and measuring latency improvements. Ping and iperf command-line tools are used for network performance analysis. Ping measures Round-Trip Time (RTT) between hosts, providing information on minimum, average, and maximum latency, while iperf measures maximum achievable bandwidth on IP networks.

The SD-WAN topology is set up using Mininet emulator with the inclusion of an external Ryu controller. Initial packet captures show the exchange of OpenFlow messages between the SD-WAN controller and SD-WAN switches, establishing the SD-WAN control plane and subsequent network operations.

Upon initializing the topology, OpenFlow Hello messages are exchanged between the SD-WAN controller and each switch to negotiate the OpenFlow version. Following this, Feature Request and Feature Reply messages are exchanged, with the controller sending a Feature Request to each switch and the switches responding with Feature Reply messages containing their capabilities and available ports. Packet-In messages are sent from switches to the controller when packets without matching flow entries are encountered, and the controller responds with Flow-Mod messages instructing the switches on packet handling. The flow tables can be examined to verify the appropriate flow entries installed by the SD-WAN controller. The following SD-WAN request response pairs are used to make QoS settings on the SD-WAN switches. The request response pairs show how the queues are set along with DSCP flow rules [10]. For example, note the queue type in response to the request 46 which is a high priority traffic, as explained earlier.

Request:

```
{ "port_name": "s1-eth1", "type": "linux-htb", "max_rate": "1000000", "queues": [ { "min_rate": "800000", "min_rate": "500000", "max_rate": "500000" } ] }
```

Response:

```
[ { "switch_id": "00000000000000001", "command_result": { "result": "success", "details": { "0": { "config": { "min-rate": "800000" } }, "1": { "config": { "min-rate": "500000" } }, "2": { "config": { "max-rate": "500000" } } } } ] }
```

Request:

```
{ "match": { "ip_dscp": "46" }, "actions": { "queue": "1" } }
```

Response:

```
[ { "switch_id": "00000000000000001", "command_result": [ { "result": "success", "details": "QoS added. : qos_id=1" } ] }
```

III. ANALYSIS AND RESULTS

A. Effect of QoS Marking on Latency

To capture the effect of QoS marking on the network latency, a test was set up that sends three types of messages continuously on the network link and shows how the network performs with and without QoS settings. The test creates three concurrent threads, each simulating a different traffic type: real-time sensor readings, periodic status messages, and system logs. Each thread runs for 30 seconds, both with and without QoS settings to obtain latency values. The results included in Table I show a comparison of latencies from three readings: high priority real-time readings, medium status readings, and low priority periodic readings. QoS effectively manages network congestion and prioritizes traffic. Without QoS, latency rapidly increases for all traffic types, reaching several seconds by the end of the test. In contrast, the QoS-enabled scenario shows consistent, low latencies for all traffic types, with a preference for higher-priority traffic.

TABLE I. COMPARISON OF LATENCIES FROM THREE READINGS: HIGH PRIORITY REAL-TIME SENSORS, MEDIUM STATUS MESSAGES, AND LOW PRIORITY PERIODIC MESSAGES.

Sim. Run Time (sec)	High Priority real-time sensor readings (100-byte messages)		Medium Priority real-time sensor readings (250-byte messages)		Low Priority real-time sensor readings (2000-byte messages)	
	Latency (ms) Without QoS	Latency (ms) With QoS (DSCP value EF 46)	Latency (ms) Without QoS	Latency (ms) With QoS (DSCP value AF31 26)	Latency (ms) Without QoS	Latency (ms) With QoS (DSCP value CSI-8)
0	18.4	15.8	17.8	16.7	26.7	33.9
5	1261	80.8	1485	81.2	1538	95.4
10	2782	82.9	3081	72.4	3083	93.9
15	4345	80.2	4508	90.8	4659	92.8
20	5054	87.3	5096	84.3	5100	97.7
25	5349	74.6	5448	95.1	5446	90.2
30	8244	80.5	7906	87.6	8172	85.4

B. Effect of Edge Server Offloading on Latency

The next test measures the latency of the SD-WAN topology when an edge server offloads part of the workload of the main cloud server. This task implements offloading percentages ranging from 10% to 90% and measures the latency for each scenario. The results presented in Figure 3 show a trend across different distances between the edge and cloud servers (1,600 km to 8,000 km). As the offloading rate increases due to more tasks processed in the edge server, latency decreases, particularly for greater distances. At lower rates offloading near 10% (mostly cloud computing), the

latency ranges from about 7.5ms (1,600 km) to about 30.5ms (8,000 km). At higher rates offloading near 90% (mostly edge processing), latency ranges from about 0.5ms (1,600 km) to 5.0ms (8,000 km). Figure 3 also expresses that edge computing significantly reduces latency, especially with higher offloading percentages. The impact is more pronounced as the distance to the cloud server increases. Even a small percentage of edge offloading can provide noticeable latency improvements, particularly for longer distances. It is noticeable that implementing 90% offloading reduces latency significantly.

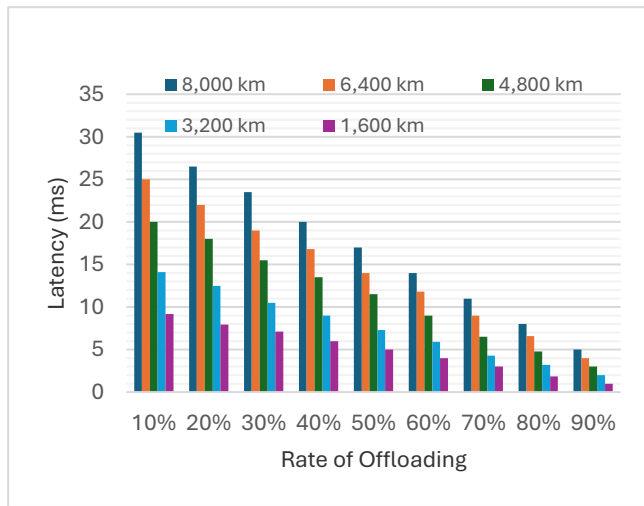


Figure 3. Latency reduction with edge server offloading.

C. Effect of Edge Cache Size on Latency

This study evaluates the impact of different edge cache sizes and request frequencies on latency in the SD-WAN topology. It checks how the latency is impacted by increasing request frequencies (10, 100, 500 and 1000 requests/sec) on increasing Edge Cache sizes (50, 100, 150 and 200 KB). Random requests are generated to derive different hit ratios based on the cache size and the Round-Trip Time (RTT) is measured for requests to both a cloud server and an edge server. Cache eviction policy is Least Recently Used (LRU). The code reads the Minimum, Average and Maximum Latency obtained from the test and plots the maximum latency values on the graph.

The results in Figure 4 show that increasing the edge cache size improves performance, with lower latencies and higher hit rates. However, the improvement is not linear and gives diminishing returns as cache size grows. Higher request frequencies lead to slightly increased latencies due to increased system load and cache contention, however, this is not consistently seen across all cache sizes. In general, it can be concluded that a four times bigger cache gives about 20% lower latency values.

D. Future Trends Based on the Use of AI

The future research and development efforts of this project can focus on various aspects such as security enhancement, and the deployment of Artificial Intelligence (AI). Security

measures such as blockchain and encryption techniques could be incorporated to protect against cyberattacks. AI integration could particularly enable predictive traffic management, and resource allocation resulted in enhancing network performance of the IoT networks.



Figure 4. Latency reduction with edge cache sizes.

IV. CONCLUSIONS AND FUTURE WORK

This paper addressed the improvement of latency in IoT networks by using the combination of Software-Defined Wide-Area Network (SD-WAN), traffic prioritization, and edge computing. IoT applications, such as smart cities and autonomous vehicles, require rapid data processing. We discussed in this paper how IoT device processing and power limitations and the use of cloud computing introduce noticeable delays. The framework we suggested required SD-WAN's centralized control and edge computing's localized processing to create a low-latency IoT infrastructure. We created SD-WAN, QoS prioritization, and edge computing, and demonstrated substantial latency reductions. In the QoS network setup with the help of DSCP field of IP packet headers, high-priority traffic experienced a dramatic decrease in latency compared to the network setup without QoS. This highlighted the effectiveness of SD-WAN's centralized control in managing network congestion and prioritizing critical data. The edge server offloading test revealed noticeable reduction in latency when high-rate tasks were offloaded to nearby edge servers, particularly in scenarios involving greater distances between edge and cloud servers. Edge caching study confirmed its role in minimizing data retrieval delays. By increasing the size of cache, the traffic latency was reduced. The future work must mainly concentrate on the inclusion of Artificial Intelligence (AI) in the structure of SD-WAN to enable predictive traffic management, resource allocation, and anomaly detection.

REFERENCES

- [1] H. Xu, W. Liu, L. Li, and Q. Zhou, "An IoT-based low-cost architecture for smart libraries using SDN," *Scientific Reports*, 2024.
- [2] J. Kurose and K. W. Ross, "Computer Networks, A Top-Down Approach," Pearson, 2017.
- [3] K. Liu, Y. Meng, and G. Sun, "An Overview on Edge Computing Research," *IEEE Access*, pp. 1-1, 2020, [Online]. Available from:

- https://www.researchgate.net/publication/341096184_An_Overview_of_Edge_Computing_Research [Accessed Apr. 9, 2024].
- [4] W. Stallings, "Foundations of Modern Networking SDN, NFV, QoE, IoT, and Cloud," Pearson Education, Inc, 2016.
 - [5] M. Beshley, N. Kryvinska, H. Beshley, O. Panchenko, and M. Medvetskyi, "Traffic Engineering and QoS/QoE Supporting Techniques for Emerging Service-Oriented Software-Defined Network", *Journal of Communications and Networks*, vol. 26, no. 1, Feb. 2024.
 - [6] Hillstone Networks, "Introduction to QoS." [Online]. Available from: https://www.hillstonenet.com/support/4.5/en/preface.html#config_qos_intro.html, [Accessed Sept. 11, 2024].
 - [7] S. Douch, M. R. Abid, K. Zine-Dine, D. Bouzidi and D. Benhaddou, "Edge Computing Technology Enablers: A Systematic Lecture Study," *IEEE Access*, vol. 10, pp. 69264-69302, 2022. [Online]. Available from: <https://ieeexplore.ieee.org/document/9797685> [Accessed Apr. 9, 2024].
 - [8] A. Jebamani and G. Winster, "A Survey of Edge Computing in IOT devices," *Proceedings of the International Conference on Innovative Computing & Communication (ICICC) 2022*. [Online]. Available from: <https://ssrn.com/abstract=4023176> [Accessed Apr. 9, 2024].
 - [9] H. Li, M. Sun, F. Xia, X. Xu, and M. Bilal. "A Survey of Edge Caching: Key Issues and Challenges", *Tsinghua Science and Technology*, ISSN 1007-0214 14/20 pp. 818–842 DOI: 10.26599/TST.2023.9010051, vol. 29, no. 3, June 2024.
 - [10] RYU Project Team, "RYU SDN Framework: Using OpenFlow 1.3", 2014. [Online]. Available from: <https://osrg.github.io/Ryu-book/en/Ryubook.pdf>, [Accessed Nov. 30, 2024].

Intelligent Pest Identification for Precision Agriculture using Deep Learning

Anika Bhat

Moreau Catholic High School

Hayward, CA, USA

email: anika.bhat1022@gmail.com

Atul Dubey

AIClub Research Institute

Mountain View, CA, USA

email: atul.dubey@aiclub.world

Abstract—Global wheat production suffers annual losses of ~157 million metric tons due to pests, causing food insecurity and economic damages exceeding \$70 billion. Traditional detection methods, such as manual inspections, are slow, labor-intensive, and often fail to identify early infestations, forcing farmers to use excessive pesticides. To address this, an image analysis system driven by Artificial Intelligence (AI) was developed, trained on pest imagery, and deployed via an accessible web application, enabling early detection to prevent crop losses. The IP102 dataset with wheat pest categories only was used to train the Machine Learning (ML) models. Two approaches were used to build an ML model that can detect wheat pests. The first employed transfer learning on MobileNetV2, and it gave the best validation accuracy of 55.32%. The second used ConvNeXtLarge to extract robust image features of 9 categories of wheat pests. Four ML algorithms, K-Nearest Neighbors (KNN), Random Forest, Multi-Layer Perceptron (MLP), and XGBoost, were trained and evaluated. The MLP model, optimized with 30 epochs and a learning rate of 0.001, achieved the highest validation accuracy of ~79% and test accuracy of ~75%. The system was integrated into a user-friendly web application, paired with a low-cost, WiFi-enabled camera device for field image capture. This system facilitated early-stage pest detection, enabling farmers to remotely monitor and take preventative measures promptly. This AI-driven model can contribute to efficient, sustainable, and precise agricultural practices and bolster global food security.

Keywords—Wheat pest identification; Deep Learning; Machine Learning; MobileNetV2; ConvNeXtLarge; Internet of Things.

I. INTRODUCTION

Wheat is the second most produced grain in the world, 785 tons in 2023-24 [1], and the US is the 4th largest producer. Wheat provides 21% of the global food requirement, but pests destroy ~157 million tons of grain/yr, causing food insecurity [1][2]. 20–40% of global crop loss/yr due to pests: \$70 billion loss/yr [3].

The most common wheat pests are Aphids, Green bugs, Ceredonta Denticonis, Spider mites, Pentahaleus Major, Wheat Blossom midges, and Wheat sawflies. In current pest monitoring methods, farmers rely on reactive and delayed pest detection, leading to irreversible crop damage and overuse of pesticides. They rely on visual monitoring, which is time-consuming and labor-intensive. Some of them use satellite/drone imagery, but it provides low-resolution images and delayed information, which can lead to missed detections and hinder early intervention efforts.

Current pest monitoring challenges include variable life cycle, nighttime activity, being hidden within plant canopies, and the need for precise timing to catch peak activity. The overuse of broad-spectrum pesticides increases pest resistance.

AI-powered agricultural image analysis is crucial in modern agriculture [4]. The AI-driven image analysis in wheat pest control can enable continuous monitoring and early detection of pests to prevent yield loss. The Deep Learning (DL) based classification and detection techniques could be very effective in identifying the type of pest from the images. Specifically, different types of Convolutional Neural Network (CNN) architectures can identify image features easily and can be effective in pest classification. This can enable rapid, scalable, and precise pest identification. Figuring out a way to put a pest detection model in a device and be able to deploy it in the field will increase the effectiveness of early detection.

In this study, we used 2 different approaches to build classifiers to identify wheat pests. In the first approach, we performed training on transfer learning on MobileNetV2 [5] by removing the top layer and adding some custom neural network layers. In the second approach, we used ConvNeXtLarge [6] model to extract features from the images and used the features and dataset to build various classifiers using techniques like KNN, Random Forest, XGBoost, and MLP. The best-performing model was used with a web app to classify different types of pests. The web app gets input from a device that captures images of wheat pests in the field. The detection information is displayed in a web application with suggestions for remedy.

The rest of the paper is laid out as follows: Section II discusses the existing research done in the pest identification and control domain. Section III elaborates on the steps that were followed to perform the study and build the required ML model and device with the web application. Section IV contains the results from various experiments performed. In Section V, we present the behavior of the models and the results obtained from different experiments. Finally, Section VI concludes the research and mentions future work.

II. RELATED WORK

Multiple studies have been done to identify wheat crop diseases. The study by Mundada and Gohokar [7] focused on the detection and classification of pests in greenhouses using traditional image processing techniques. Images of infected leaves were captured, and properties like entropy, mean, standard deviation, contrast, energy, correlation, and eccentricity were extracted. These features were then used to train a Support Vector Machine (SVM) for classification. They developed a software prototype system for early pest detection in greenhouses, achieving a training accuracy of 100% with the

SVM classifier. By utilizing featurization techniques, images from the existing dataset were filtered to form a new dataset, enhancing the ML model's effectiveness.

Shi et al. [8] conducted research on pest and disease detection in winter wheat using spectral indices and kernel discriminant analysis. Hyperspectral reflectance datasets at both leaf and canopy levels were utilized, involving fourteen Spectral Vegetation Indices (SVIs) as input. The approach showed better performance over conventional linear methods, achieving classification accuracies between 76% and 95% for various infestations.

Haider et al. [9] explored a generic approach to wheat disease classification, incorporating field surveys, expert opinion, and crowd-sourced data. Using symptoms as predictor variables, a decision tree model was developed for disease classification, utilizing a reduced error pruning algorithm (RepTree). This approach was supplemented by expert opinions to verify data, achieving a classification accuracy of 97.2% with a CNN model. Their methodology demonstrates a blend of traditional approaches with modern data collection methods.

The research by Kasinathan et al. [10] delved into insect classification and detection using modern ML techniques. They utilized datasets, such as the Wang dataset and the Xie dataset, extracting nine insect shape features after preprocessing images to grayscale. These shape features were classified using various algorithms, like Artificial Neural Network (ANN), SVM, KNN, and Naive Bayes (NB). The study demonstrated that CNNs provided the highest classification accuracy of 91% on certain insect datasets, showcasing a gradual shift towards more complex ML methods.

Abbas et al. [11] employed fuzzy logic-based histogram equalization to enhance image quality for better disease recognition in wheat leaves. This approach leverages fuzzy logic to improve image contrast, leading to more accurate disease recognition. The application of this technique represents an advancement beyond basic image processing, enhancing the quality and interpretability of images for subsequent analysis.

In their work, Kang et al. [12] proposed a DL model for pest detection, introducing an attention mechanism-enhanced single-stage object detection framework with multiscale feature fusion. This model focused particularly on identifying small-scale pests in complex backgrounds. It outperformed models, such as You Only Look Once (YOLO), EfficientDet, RetinaDet, and MobileNet, achieving the highest mean Average Precision (mAP) value of 0.91.

Xu et al.'s research [13] on wheat leaf disease identification leveraged an integrated DL framework called Recursive Feature Elimination-Convolutional Neural Network (RFE-CNN), which incorporates Residual Channel Attention Blocks (RCAB), Feedback Blocks (FB), and Elliptic Metric Learning (EML). It begins with using parallel CNNs to extract features from healthy and diseased leaves, optimizing them with RCABs, training them with FBs, and concluding with a CNN for classification. This approach resulted in an overall classification accuracy of 98.83% and a maximum testing accuracy of 99.95%.

Liu et al. [14] introduced PestNet, a DL model for multi-class

pest detection and classification. PestNet blends a Channel-Spatial Attention (CSA) mechanism with a CNN backbone, utilizing a Region Proposal Network (RPN) and a Position-Sensitive Score Map (PSSM). This integration of attention mechanisms and advanced network architectures produced an mAP of 75.46%, outperforming other methods, indicating complexity in both design and application.

Chamara et al.'s [15] project is an effort toward real-time crop monitoring utilizing edge devices. They deployed a stack of Deep Convolutional Neural Networks (DCNN) models: CropClassiNet for crop type classification, CanopySegNet for canopy cover quantification, PlantCountNet for plant and weed counting, and InsectNet for insect identification. With CropClassiNet achieving 94.5% accuracy and CanopySegNet 92.83% accuracy, the project illustrated an implementation of DCNNs for integrated crop management solutions.

There are multiple dimensions explored in the earlier research; however, no ready-to-use practical solution is currently available to be used in the field. Some of them built edge devices, but they are not power-efficient and can not run for long in the field. The solutions also lack real-time monitoring capabilities. In our study, we used the IP102 dataset [16] with DL to create a real-time system for identifying pests in wheat crops. Our method improves upon past approaches by combining ConvNeXtLarge for feature extraction and MLP for classification, making it both accurate and efficient. Unlike earlier studies that relied on manual inspections or limited models, our solution includes a solar-powered device and a web app for easy, continuous monitoring. This makes pest detection faster, cheaper, and more practical for farmers, helping reduce crop losses early on. Table I compares the results and techniques of existing literature with this research.

III. MATERIALS AND METHODS

A. Dataset

A database of images of pests was downloaded [16], and the images of wheat crop pests were extracted (9 categories). The distribution of images in each class is presented in Figure 1. The wheat crop pests database was split into training (2048), validation (340), and testing (1030) sets.

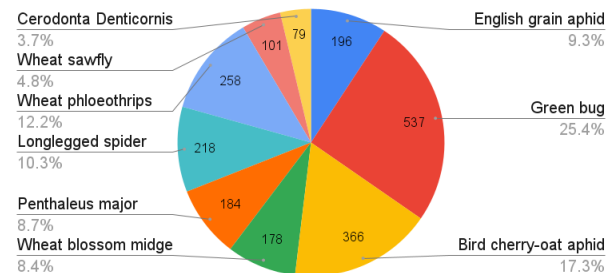


Figure 1. Distribution of images of wheat crop pests in the dataset.

B. Deep Learning models

1) *MobileNetV2*: A lightweight CNN model with 3.5M parameters and the ability to run on resource-constrained mobile

TABLE I. SUMMARY OF DATASETS, TECHNIQUES, AND RESULTS FROM DIFFERENT STUDIES.

Ref	Dataset Type	ML/DL Technique	Results
[7]	Camera captured whiteflies & aphids images	SVM	Training to the SVM is done with 100% accuracy
[8]	Leaf (314) and canopy (187) level hyperspectral reflectance datasets	Spectral Vegetation Indices-based Kernel Discriminant Approach (SVIKDA)	At leaf level: 89.2% At canopy level: 87%
[9]	2324 symptoms samples from our symptom-based text dataset	Decision Tree, Error Pruning Tree (Rep-Tree), CNN	97.20%
[10]	Wang dataset (225 images) with nine insect classes and Xie dataset with 24 classes	ANN, SVM, KNN, and NB classifier	91%
[12]	Rice pest images with complex backgrounds and small-sized pests	Attention Mechanism, Multi-Scale Feature Fusion, Single-Stage Object Detection Model, YOLO, EfficientDet, RetinaDet, and MobileNet	91% mAP
[13]	CGIAR, Plant Diseases, LWDCD 2020	RCAB, FB, EML, and CNN	98.83%
[14]	Multi-class Pest Dataset 2018 (MPD2018) with over 80,000 images and 580,000 labeled pests across 16 classes	CNN with CSA, RPN, and PSSM	75.46% mAP
[15]	43,000 field crop images collected offline	Stack of four DCNN models: Crop-ClassiNet, CanopySegNet, PlantCountNet, and InsectNet	94.5% accuracy
This research	IP102 dataset with wheat pest categories	MobileNetV2 (transfer learning), ConvNeXtLarge (featurization), KNN, Random Forest, MLP, and XGBoost	Validation accuracy: 78.72% Test accuracy: 74.51%

devices. This model introduces the concept of inverted residuals with linear bottlenecks. This approach preserves the input and output dimensions while performing the intermediate layers in a lower-dimensional space, reducing the computational cost.

2) *ConvNeXtLarge*: A CNN model with 197.7M parameters, with weights trained on the ImageNet dataset. ConvNeXt is a type of neural network that is built based on another design called Vision Transformers (ViTs). It uses a technique called depth-wise convolution. It is a special way of processing images where the network looks at different parts of the image separately. This technique helps to reduce the amount of calculations needed while still maintaining accuracy. We have used this model to featurize the images.

C. Machine Learning

The development of the ML model follows two main approaches. The first approach uses transfer learning on MobileNetV2, and the second approach uses feature extraction using ConvNeXtLarge and the application of classical ML techniques on the output features. Transfer learning with MobileNetV2 is similar to featurizing using MobileNetV2 and building a classifier using MLP. The dataset was featurized using ConvNeXtLarge and those features were used to create different ML models. The performance of these models was compared with the performance of the models created using MobileNetV2.

1) *Transfer Learning on MobileNetV2*: Considering the size of our dataset, it will not be an appropriate approach to train an architecture from scratch. This is where transfer learning works better, where we have the possibility of building high-performing models even with smaller datasets. This is the reason we decided to take this approach.

As mentioned in the left part of the flowchart in Figure 2, the final layer of a pre-trained MobileNetV2 model is removed to perform transfer learning. GlobalAveragePooling2D was applied to the output from the second-to-last layer to reduce the spatial dimensions of the input tensors. Next, a neural network layer with 100 neurons and a Rectified Linear Unit (ReLU) activation function was added. Finally, another neural network layer with 9 neurons and a softmax activation function was added to give the classification probabilities.

The model is then trained using the training dataset and then validated on the validation dataset. While training, the hyperparameters, such as learning rate (between 0.0001 and 0.01) and the number of epochs (between 10 and 50), are tuned to find the best-performing model. The ranges of learning rates and epochs are arbitrarily selected, and the plan was to extend them if the model converges in the right direction. The best model is then tested on the test dataset, and the performance is recorded in a Google sheet.

2) *ConvNeXtLarge and Classical ML*: As shown in the right part of the flowchart in Figure 2, ConvNeXtLarge without the last layer was used as a feature extractor to convert the images

into a set of features, and the extracted features are saved into a CSV file. Generally, a CNN architecture is designed to extract various features, such as edges, shapes, etc., and the extracted features are nothing but a set of numbers. Those numbers stored in tabular data can be used with classical ML techniques to build various classifiers. Using ConvNeXtLarge, we have converted each image into a vector of size 2048. We have used various classical ML techniques, including KNN, Random Forest, XGBoost, and MLP, to train a model. The KNN technique was selected due to its simplicity and ability to work with fewer resources. During training, the K-values were varied from 2 to 14, and performance was recorded for the validation dataset. Similarly, the Random Forest technique is an ensemble of various decision trees, which are again faster to run and consume less computational resources. During training, the number of trees was varied from 10 to 100, and the depth from 1 to 7, to find the best-performing model on the validation dataset. The same settings were used for the XGBoost (which is an improvement on the Random Forest model by adding gradient boosting to it), and the performance on the validation dataset was recorded. The MLP technique is one of the simplest DL techniques and works well with datasets with fewer features. While training, the number of epochs varied from 10 to 100, and the learning rate varied from 0.00001 to 0.05; the validation results were recorded in a Google sheet. The best-performing model is then tested on the test dataset.

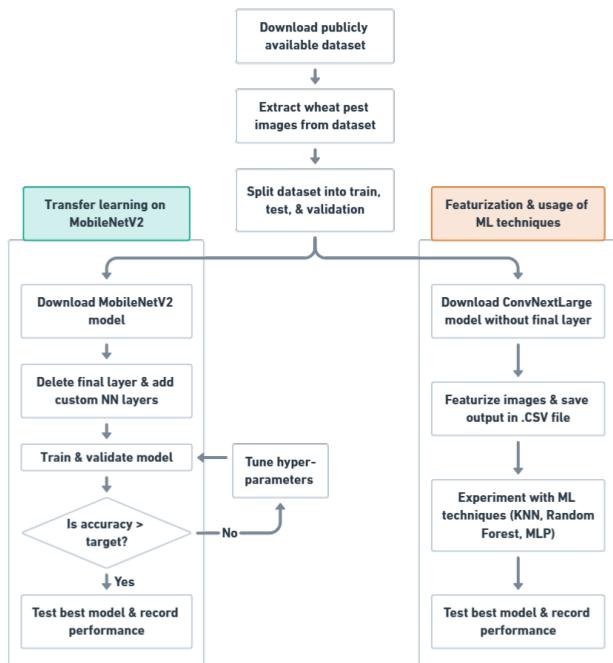


Figure 2. Machine Learning pipeline: steps followed for building ML models.

D. Device Development and Model Deployment

As depicted in the hardware architecture diagram in Figure 3, for the device, the controller is a Seeed Studio XIAO ESP32S3 Sense, optimized for ML and suitable for real-time image recognition tasks. The controller has a dual-core 32-bit

processor with a 240 MHz Frequency and 512 KB of SRAM. The module also has 8 MB of PSRAM that allows it to process images faster and run them through the neural network model. The controller is WiFi-enabled, which ensures it can be made part of the internet, and the captured data can be sent to an app via the cloud. It includes a built-in camera sensor with a maximum resolution of 1600 x 1200 pixels, with a Camera Serial Interface (CSI) connecting the camera to the controller. The system uses a solar-powered power bank for energy supply to ensure nonstop working even in a remote setup. As shown in the firmware flowchart in Figure 4, once the controller and camera are initialized, an image is captured and saved. The image is then serialized and sent to the web app for pest detection, where the type of pest is identified. After each image capture, the device waits five minutes before capturing the next image to ensure continuous monitoring. For demonstration purposes, pest predictions in the app are user-initiated.



Figure 3. Hardware architecture diagram.

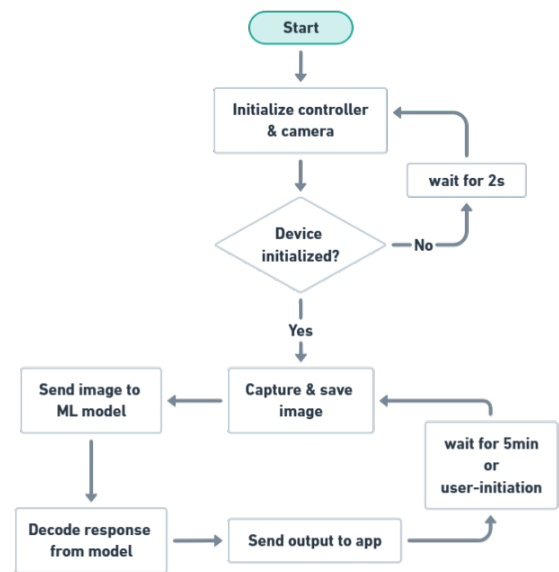


Figure 4. Firmware flowchart: logic implemented in the device.

E. Web App

The web app is developed using the Streamlit Python library and then deployed in the Streamlit cloud. The required featurizer and model are deployed as part of the app itself.

As shown in the application flowchart presented in Figure 5, when an image is received from the device in the web app, it is resized to 224 x 224 pixels and then featurized using ConvNeXtLarge. The image is processed, and the features are passed through the MLP classifier. The image and prediction of the model are then displayed to the user. Users can also see the time of detection and targeted pest removal suggestions. Each pest prediction is saved in the app for future reference.

The code for the web application and notebooks used for featurization and training the models can be found in [17].

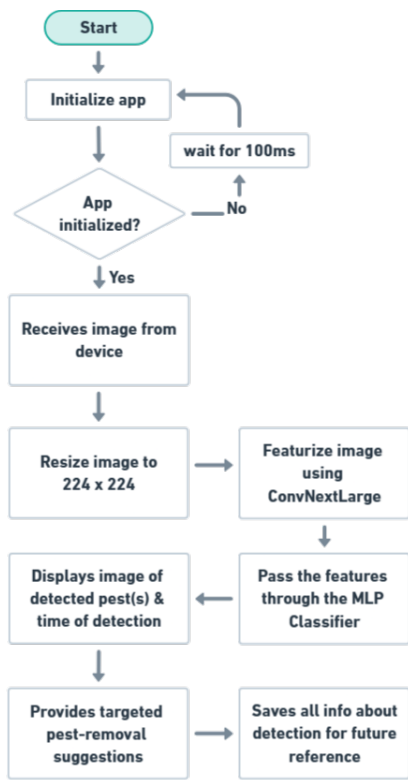


Figure 5. Application flowchart.

IV. RESULTS

We have experimented with various ML techniques using 2 approaches. In the first, we performed transfer learning on MobileNetV2, and in the second, we used the ConvNeXtLarge model to featurize the images and then used features with ML techniques, such as KNN, Random Forest, XGBoost, and MLP, to build various models.

A. MobileNetV2

A total of 25 experiments were performed by varying the learning rates from 0.0001 to 0.01 and the epochs from 10 to 50. The best validation accuracy of 55.32% was achieved at a learning rate of 0.0005 with 40 epochs. The variation in validation accuracies with epochs for different learning rates is presented in Figure 6.

B. KNN

A total of 13 experiments were performed by varying the K values from 2 to 14. The best validation accuracy of 66.81% was achieved at a K value of 8. The variation in validation accuracies with the K values can be found in Figure 7.

C. Random Forest

A total of 70 experiments were performed by varying the depth from 1 to 7 and the number of trees from 10 to 100. The best validation accuracy of 62.98% was achieved at a depth of

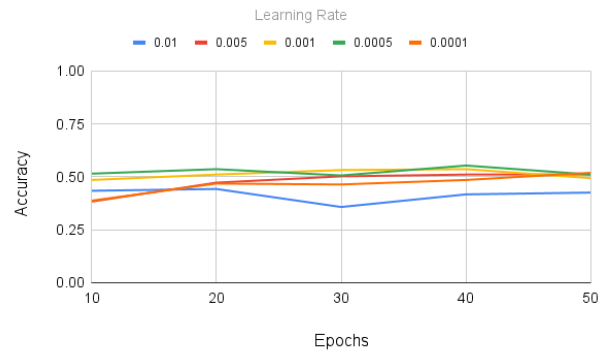


Figure 6. MobileNetV2: Validation accuracy vs. epochs for different learning rates.

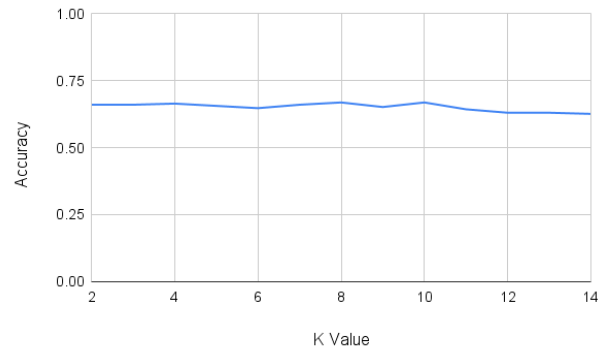


Figure 7. KNN: Validation accuracy vs. K values.

7 with 70 trees. The variation in validation accuracies with the number of trees for different depths is presented in Figure 8.

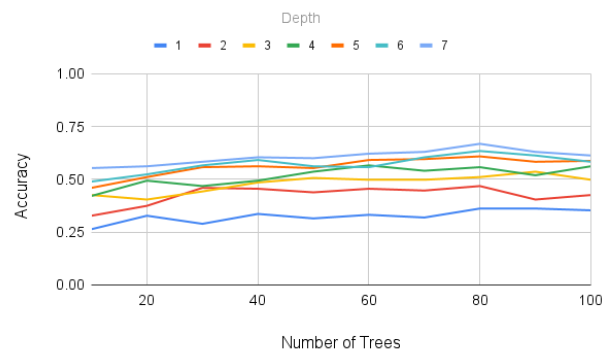


Figure 8. Random Forest: Validation accuracy vs. number of trees for different depths.

D. XGBoost

A total of 70 experiments were performed by varying the depth from 1 to 7 and the number of trees from 10 to 100. The best validation accuracy of 71.91% was achieved at a depth of 4 with 50 trees. The variation in validation accuracies with the number of trees for different depths is presented in Figure 9.

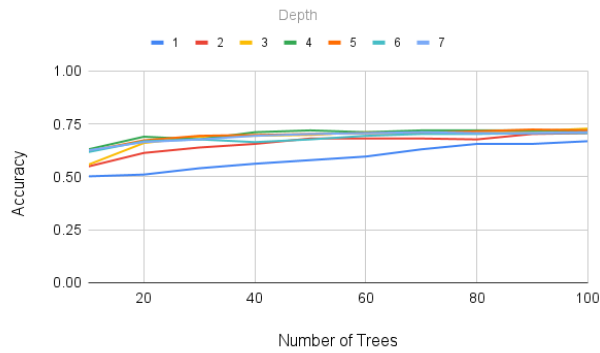


Figure 9. XGBoost: Validation accuracy vs. number of trees for different depths.

E. MLP

A total of 50 experiments were performed by varying the learning rates between 0.0001 and 0.01, and epochs between 10 and 50. The best validation accuracy of 78.72% was achieved at a learning rate of 0.001 with 30 epochs. The variation in validation accuracies with epochs for different learning rates is presented in Figure 10.

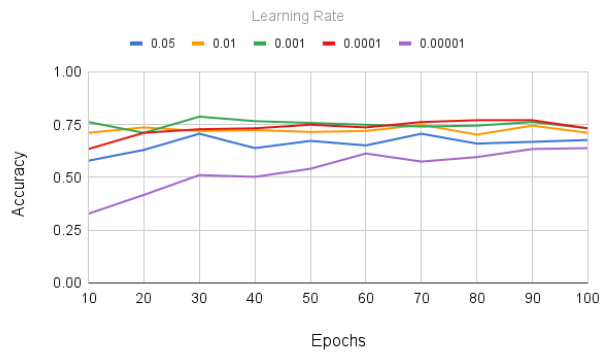


Figure 10. MLP: Validation accuracy vs. epochs for different learning rates.

F. Results summary

TABLE II. VALIDATION ACCURACIES FOR THE BEST MODELS.

ML Model	Validation Accuracy	Precision	Recall	F1 Score
MobileNetV2	55.32%	46%	47%	46%
KNN	66.81%	72%	67%	67%
Random Forest	62.98%	64%	63%	62%
XGBoost	71.91%	74%	73%	73%
MLP	78.72%	77%	77%	77%

Out of all the ML techniques, MLP gave the best validation accuracy of 78.72%. The final model was then tested with the test dataset and achieved an accuracy of 74.51%.

As per the confusion matrix in Figure 11, the model has performed well for most of the categories except "Wheat sawfly" and "English grain aphid". To make it perform well for those 2 categories as well, we might need to add more dataset and clear images for those categories.

G. Real-world test results

The model was deployed with a web application in Streamlit Community Cloud. The device was connected to the application using a port forwarder. Once the application is ready, it takes ~6 seconds to perform inference.

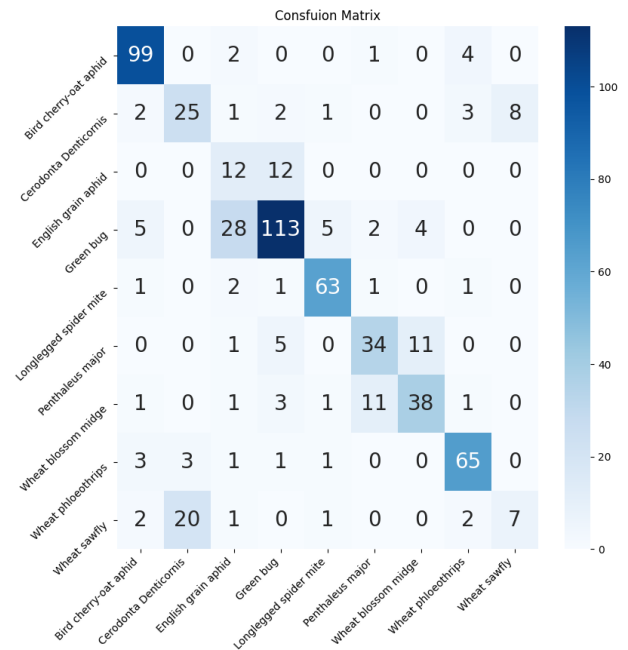


Figure 11. MLP: Confusion matrix of test results.

V. DISCUSSION

For MobileNetV2, at a learning rate of 0.01, the accuracy varied with no consistent improvement as epochs increased, indicating possible instability, likely due to the higher learning rate causing weight oscillations. Conversely, at a lower learning rate of 0.0005, a more stable accuracy was achieved, peaking at 0.5531 with 40 epochs, suggesting better gradual convergence despite minor fluctuations at 30 epochs. The learning rate of 0.001 emerged as particularly effective, demonstrating a continual improvement across epochs, achieving a maximum accuracy of 0.5362 at 40 epochs.

For KNN experiments, as the K value increased beyond 10, a decrease in precision was observed, with K = 11, 12, 13, and 14 producing lower precision, reaching a minimum of 0.6255 at K = 14. This trend suggests that larger K values might oversmooth the decision boundary, leading to underfitting.

Regarding Random Forest, increasing both the number of trees and the depth contributes to improved accuracy, but with varying degrees of effectiveness. At a depth of 1, performance is limited in all tree counts, with the highest accuracy reaching only 36.17% at 80 and 90 trees. This underperformance is expected due to the insufficient depth to make complex decisions. The results suggest that deeper trees offer better performance, provided that there is a sufficiently large number of trees to offset the variance associated with deeper models. However, increasing the number of trees beyond 70 generally

yields diminishing returns with high-depth models, likely due to saturation in ensemble benefits.

Regarding XGBoost, the influence of depth is prominent at levels 4 and 5, where accuracies consistently approach or exceed 71.91% beyond 50 trees, indicating that deeper trees can effectively tackle complexity and provide robust performance as the ensemble size grows. However, with depths 6 and 7, the incremental gain in accuracy decreases, suggesting a potential overfitting tendency or saturation, where additional depth does not necessarily translate to substantial improvements.

Regarding MLP, at a higher learning rate of 0.05, performance was moderate with fluctuating accuracies across epochs, peaking at 70.64% at both 30 and 70 epochs. This suggests that while a higher learning rate allows for rapid convergence early on, it may cause instability, leading to non-consistent improvements. As the learning rate decreases to 0.001, we observe some of the highest performance metrics, achieving a maximum accuracy of 78.72% at 30 epochs. This learning rate allows the model to explore the solution space, leading to consistent performance and better generalization. Lowering the learning rate further to 0.0001, the model maintained stability, with accuracies consistently high, peaking at 77.02% at both 80 and 90 epochs. This stability reflects the advantage of smaller learning rates, although it requires more epochs to reach effective solutions.

Overall, the best accuracy achieved out of all the experiments was from the MLP technique, suggesting the effectiveness of DL techniques with complex image datasets. However, the performance can improve with architectural changes or experimenting with other CNN models for featurization.






VI. CONCLUSION AND FUTURE WORK

A DL model is developed for wheat pest detection, as well as a web application for real-time pest identification and targeted pest-removal suggestions. The best model achieved the best test accuracy of ~75%, and once deployed with a device, it can help mitigate crop loss in the early stages of pest infestation. The device offers low-cost, easy-to-use, efficient, convenient, and remote monitoring capabilities, eliminating the need for manual pest monitoring. The device and web app can lead to better pest control, leading to less economic loss and improved food security. However, there are limitations and areas for future study. The current image resolution is not ideal, and better images could improve model accuracy. Additionally, multiple devices (cameras) are needed to cover a large wheat field; future studies may explore the use of drone cameras with higher resolution. Notification features can also be added to the app so that the user can be notified when a pest is detected, as well as building a mobile app to enhance functionality.

REFERENCES

- [1] *Wheat Production by Country 2025* — *worldpopulationreview.com*, <https://worldpopulationreview.com/country-rankings/wheat-production-by-country>, [Accessed 08-21-2025].
- [2] *Wheat* — *agmrc.org*, <https://www.agmrc.org/commodities-products/grains-oilseeds/wheat>, [Accessed 08-21-2025].
- [3] S. P. A. S. Lori Tyler Gula, *Researchers Helping Protect Crops From Pests*, <https://www.nifa.usda.gov/about-nifa/blogs/researchers-helping-protect-crops-pests>, [Accessed 08-21-2025], 2023.
- [4] J. Yang and Y. Zhou, "Efficient pest classification using lightweight neural networks for sustainable pest control", in *2024 4th International Conference on Computer Communication and Artificial Intelligence (CCAI)*, IEEE, 2024, pp. 119–123.
- [5] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.
- [6] Z. Liu *et al.*, "A convnet for the 2020s", in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 11976–11986.
- [7] R. G. Mundada and V. V. Gohokar, "Detection and classification of pests in greenhouse using image processing", *IOSR Journal of Electronics and Communication Engineering*, vol. 5, no. 6, pp. 57–63, 2013.
- [8] Y. Shi, W. Huang, J. Luo, L. Huang, and X. Zhou, "Detection and discrimination of pests and diseases in winter wheat based on spectral indices and kernel discriminant analysis", *Computers and electronics in agriculture*, vol. 141, pp. 171–180, 2017.
- [9] W. Haider, A.-U. Rehman, N. M. Durrani, and S. U. Rehman, "A generic approach for wheat disease classification and verification using expert opinion for knowledge-based decisions", *IEEE Access*, vol. 9, pp. 31104–31129, 2021.
- [10] T. Kasinathan, D. Singaraju, and S. R. Uyyala, "Insect classification and detection in field crops using modern machine learning techniques", *Information Processing in Agriculture*, vol. 8, no. 3, pp. 446–457, 2021.
- [11] F. I. Abbas, N. M. Mirza, A. H. Abbas, and L. H. Abbas, "Enhancement of wheat leaf images using fuzzy-logic based histogram equalization to recognize diseases", *Iraqi Journal of Science*, pp. 2408–2417, 2020.
- [12] H. Kang *et al.*, "A novel deep learning model for accurate pest detection and edge computing deployment", *Insects*, vol. 14, no. 7, p. 660, 2023.
- [13] L. Xu *et al.*, "Wheat leaf disease identification based on deep learning algorithms", *Physiological and Molecular Plant Pathology*, vol. 123, p. 101940, 2023.
- [14] L. Liu *et al.*, "Pestnet: An end-to-end deep learning approach for large-scale multi-class pest detection and classification", *Ieee Access*, vol. 7, pp. 45301–45312, 2019.
- [15] N. Chamara, G. Bai, and Y. Ge, "Aicropcam: Deploying classification, segmentation, detection, and counting deep-learning models for crop monitoring on the edge", *Computers and Electronics in Agriculture*, vol. 215, p. 108420, 2023.
- [16] X. Wu, C. Zhan, Y. Lai, M.-M. Cheng, and J. Yang, "Ip102: A large-scale benchmark dataset for insect pest recognition", in *IEEE CVPR*, 2019, pp. 8787–8796.
- [17] A. Bhat, *GitHub - anikaaa22/Pest-Identification* — *github.com*, <https://github.com/anikaaa22/Pest-Identification>, [Accessed 08-21-2025].

A Neural Approach to Ray Tracing for Realistic Wireless Channel Simulation in Indoor and Urban Scenarios

Francisco Javier Somolinos-Simón , Adina Murg, Hanli Liu ,
Carlos J. Hellín , Josefa Gómez  and Abdelhamid Tayebi 

Department of Computer Science

University of Alcalá

Alcalá de Henares, Spain

e-mail: {francisco.somolinos | adina.murg | hanli.liu
| carlos.hellin | josefa.gomezp | hamid.tayebi}@uah.es

Abstract—Accurate modeling of wireless channels is essential for the design and optimization of next generation communication networks such as 6G. Traditional ray tracing techniques provide physically consistent simulations but suffer from high computational complexity, limiting their scalability and real-time applicability. This work proposes a neural network based surrogate model for ray tracing in complex 3D environments. This approach leverages multilayer perceptrons to predict the interaction of electromagnetic rays with surfaces, estimating critical channel parameters such as gain, time-of-flight, and propagation angles. The model is trained and validated using datasets generated by the Sionna ray tracing engine in both indoor and large urban scenarios. Results demonstrate that the neural surrogate achieves low prediction errors in key metrics and generalizes well across different environments. This neural ray tracing framework offers a scalable, flexible, and efficient alternative to conventional physics based simulators.

Keywords—neural networks; ray tracing; MIMO systems.

I. INTRODUCTION

In recent years, many 6G network research topics have required the simulation of specific radio environments using ray tracing. This requirement arises from the need for a spatially consistent correspondence between a physical location in a scene and the impulsive channel response, a feature not readily provided by widely used stochastic channel models. The main challenges for the design, deployment and optimization of wireless communication networks are based on understanding and accurately modeling the characteristics of the real-time propagation channel, allowing fast and accurate simulations of complex scenarios such as massive Multiple-Input Multiple-Output (MIMO) systems and the deployment of digital twins, among others [1].

The physics of such ElectroMagnetic (EM) wave propagation between a transmit and receive point are analytically given by the Maxwell equations: the transmitted wave undergoes different interactions with the environment (e.g., reflection), and the receiver gets the wave through multiple paths with different times-of-flight and powers, and from different directions. However, solving the Maxwell equations with boundary conditions requires in-depth knowledge of the propagation environment, therefore, classically modeling EM propagation is intractable for most engineering applications [2].

Ray tracing-based simulators are commonly employed for modeling wireless channel properties [3]–[6]. In the ray tracing

process, electromagnetic rays are uniformly launched from the transmitter antenna, undergoing reflections, transmissions, and diffractions with various buildings and floors, ultimately reaching the receiver locations. These ray paths and interactions yield valuable wireless channel information, such as channel gain, channel transfer function, and channel impulse response [7].

While ray tracing has been a popular tool in wireless channel modeling, its computational complexity escalates with the number of ray-object interactions. To address these needs, neural network based forward surrogate models emerge as an attractive solution.

A neural network is a mechanism that takes inputs and learns associations to predict some outputs [8]. Artificial Neural Network (ANN) models are gaining importance in the field of predictive modeling because of their capability to model nonlinear relationships in a high-dimensional dataset. ANN models can predict a complex relationship between variables, which is not otherwise possible with other models such as logistic regression models [9].

ANN models work on the principles of biological neural networks containing nodes (analogous to cell bodies) that communicate with other nodes through connections (analogous to axons and dendrites) [10]. An ANN consists of an input layer, hidden layers and an output layer while the information is fed into the model through the input layer, processed through the hidden layers and put out from the output layer [9] (Figure 1).

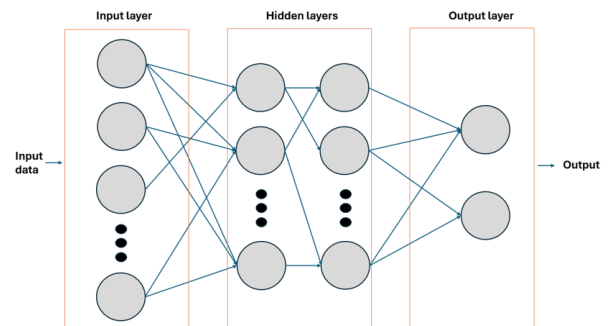


Figure 1. Structure of a neural network showing input features, hidden layers for pattern learning, and output neurons representing predicted quantities.

Among the main types of neural network models generally used in predicting, Multi-Layer Perceptron (MLP) stands out. It is one of the feedforward networks where the input values are multiplied with their corresponding weights and fed into the hidden layer while the hidden layer process transfers the weighted input to the output layer with values of multiplied weights corresponding to the output layer. MLP uses the backpropagation algorithm to recalculate the weights [9]. It uses commonly conventional activation functions such as the Sigmoid function or the Tanh function, however, any non-linear and continuous function, such as Rectified Linear Units (ReLU), is suitable for use in an MLP [11].

The major highlights of this model are as follows [12]:

- The neural network contains one or more intermediate layers between the input and the output nodes, which are hidden from both input and output nodes.
- Each neuron in the network includes a non-linear activation function that is differentiable.
- The neurons in each layer are connected with some or all the neurons in the previous layer.

Then, the objective of this paper is to train neural networks to predict interactions between wireless beams and objects in a three-dimensional environment. This approach involves simulating ray-surface interactions to estimate transmission-reception paths, considering characteristics such as time of flight and gain. The proposed neural network learns how surfaces impact wireless ray propagation, predicting factors such as attenuation and direction of the outgoing ray based on attributes of the incident ray. This approach offers the advantage of applying to new scenarios, improving its versatility to adapt to different situations.

Compared to existing neural ray tracing methods in the literature, the proposed work excels in scalability and flexibility, accommodating diverse levels of geometric complexity while maintaining high-quality channel prediction. This neural ray tracing framework is validated across indoor and outdoor scenes and shows potential for real-time or large-scale deployment, particularly in the context of next generation 6G communication systems and digital twin technologies.

The remainder of this paper is organized as follows: Section II gives a short overview of the work related to the idea to be put forward. Section III details how datasets are generated using the Sionna ray tracing engine in both indoor and outdoor 3D scenes. Section IV introduces the architecture of the proposed model. Section V explains the model training process. Section VI presents three metrics: Overall Error, Geometry Error, and Average Delay Mean Absolute Error (MAE), to assess how accurately the model predicts the physical behavior of wireless paths compared to ground-truth data. Section VII presents the results of the proposed neural ray tracing model in various scenarios and evaluates its performance using the defined metrics. Finally, Section VIII summarizes the main contributions of the paper and outlines potential directions for future research.

II. RELATED WORK

Physically based simulation guided by neural networks is gaining popularity across various scientific domains. In the field of applied and computational electromagnetics, several approaches leveraging neural networks have been proposed to accelerate or approximate ray-based simulations [13]–[15]. For instance, Jin et al. [15] redefine ray trajectory generation as a sequential decision making problem, introducing the SANDWICH framework, a fully differentiable, scene aware neural architecture that jointly learns optical, physical, and signal properties of the environment.

On the other hand, other approaches from the domain of neural rendering and computer graphics also employ neural networks to model ray tracing and light transport. Knodt et al. [16] explicitly model light transport between scene surfaces using disentangled neural representations of geometry and reflectance, allowing for efficient inverse rendering. Zeng et al. [17] propose MirrorNeRF, a neural rendering framework capable of learning accurate geometry and mirror reflection, supporting scene manipulations such as adding new objects or modifying reflective surfaces and synthesizing corresponding reflections.

Many of these neural surrogates aim to learn the scattering process involving obstacles in free space.

A recent work addressing this task is WiNeRT [2]. In the authors' approach, a neural surrogate to model wireless electromagnetic propagation effects in indoor environments is implemented. Such neural surrogates provide a fast, differentiable, and continuous representation of the environment and enable end-to-end optimization for downstream tasks. Specifically, they render the wireless signal (e.g., time of flight, power of each path) in an environment as a function of the sensor's spatial configuration (e.g., placement of transmit and receive antennas). That is to say, their approach inscribes within the ray tracing channel modeling paradigm, where wireless propagation is precisely modeled by tracing wireless rays.

Another work in this area is RayProNet [7]. The authors introduce a novel machine learning-empowered methodology for wireless channel modeling. The key ingredients include a point-cloud-based neural network and a Spherical Harmonics encoder with light probes. Their approach offers the flexibility to adjust antenna radiation patterns and transmitter/receiver locations, the capability to predict radio path loss maps, and the scalability of large-scale wireless scenes. This work is validated in various outdoor and indoor radio environments.

Additionally, widely adopted datasets such as DeepMIMO [18] support the training and benchmarking of data-driven channel models in ray-traced environments, and are instrumental in standardizing the evaluation of neural surrogates. More recently, physics-informed learning techniques have been proposed to embed propagation physics into deep models, improving generalizability and interpretability in scenario aware channel modeling [19].

III. DATA COLLECTION

The datasets in this project are generated using an open-source ray tracing simulator: Sionna [20]. Sionna Ray Tracing (RT) is a ray tracing extension for radio propagation modeling that is built on top of Mitsuba 3 [21] and TensorFlow [22]. Sionna RT relies on Mitsuba 3 for the rendering and scene handling, e.g., its XML-file format, as well as the computation of ray intersections with scene primitives, i.e., triangles forming a mesh modeling a surface [23]. Scene files for Mitsuba 3 may be created, edited, and exported using the popular open-source 3D content creation suite Blender [24] and the Mitsuba-Blender add-on. The dataset is generated in various scenes such as a cube (small indoor room scene) (Figure 2), and Munich (large urban city scene) (Figure 3).

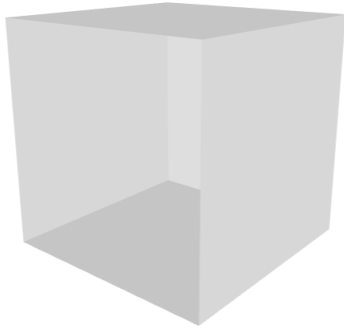


Figure 2. Cube: small indoor room scene which contains marble material.

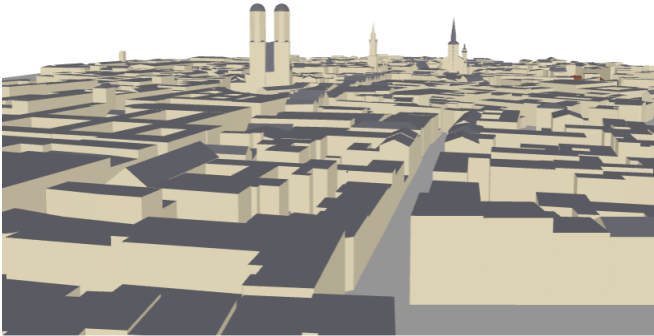


Figure 3. Munich: large urban city scene with a rich material composition, including both marble and metal.

The first scene contains marble material consistent with its simple indoor layout. In contrast, the second scene features a richer material composition, including both marble and metal, to better reflect the diversity of an urban landscape [25]. Table I shows the properties of these materials, as relative permittivity (ϵ_r) and conductivity (σ).

TABLE I. CONSTITUENT MATERIAL PROPERTIES (RELATIVE PERMITTIVITY ϵ_r AND CONDUCTIVITY σ)

Material name	ϵ_r	σ [S/m]
Marble	7.074	0.018
Metal	1	10^7

As described in Table II, datasets comprise 20 transmitter locations with Third Generation Partnership Project (3GPP) TR 38.901 pattern [26] and 40 sampled receiver locations with short dipole pattern with linear polarization pattern for each scene. To position the transmitters and receivers in the scene, a random uniform sampling strategy was adopted. Specifically, antenna positions were sampled uniformly in the plane within the bounding box of the scene, centered around the scene's geometric center. A fixed random seed was used to ensure reproducibility. This uniform spatial sampling avoids positional bias and ensures that the antennas are evenly distributed across the area of interest (671 m^2 for small indoor room scene and $4,082,653 \text{ m}^2$ for large urban city). The operating frequency is 3.5 GHz. Reflection is activated whereas diffraction and scattering are not activated.

TABLE II. DATA COLLECTION: CONFIGURATION DETAILS OF DATASETS

Dataset	Cube	Munich
Scale	Small indoor room scene	Large urban city
Covered area (m^2)	671	4,082,653
Transmitters	20	20
Receivers	40	40
Antenna pattern	Transmitter: 3GPP TR 38.901[26] Receiver: short dipole pattern with linear polarization	Transmitter: 3GPP TR 38.901[26] Receiver: short dipole pattern with linear polarization
Frequency (GHz)	3.5	3.5
Number of bounces	3	3
Number of rays to trace	10^6	10^6
Reflected paths are computed	True	True
Diffacted paths are computed	False	False
Scattered paths are computed	False	False

IV. NEURAL MODEL

The general goal of the project is to contribute to the advancement of next generation networks by designing, developing, and testing an innovative neural network based ray tracer. The model takes three configuration parameters as input: a 3D representation of the environment and the spatial coordinates of the transmitter and receiver devices. The model predicts the wireless scene where the output is a variably-sized set of K paths. A path consists of a sequence of ray segments ($r, r+1 \dots$) connecting a transmitter to a receiver. Each path encodes three channel attributes: gain (a_k), time-

of-flight (τ_k) and angles (Φ_k). This approach allows for the effective encoding of interacting objects, with a particular emphasis on learning geometric features. Therefore, the final state is modeled as an evaluation of the interactions the ray experiences with its environment which is represented as a 3D mesh composed of F faces and V vertices, where each face corresponds to some surface on a wall.

An MLP model is built to predict the transformation to the incident ray (the new direction ($d_k^{(r+1)}$) and gain ($a_k^{(r+1)}$)). The network predicts an attenuation factor s and a rotation matrix A (4-dim Euler-Rodrigues parameterization), which is then used to determine the updated gain and direction.

Specifically, this neural model consists of two MLP networks, an MLP spatial network with 2 hidden layers, each with 64 hidden units and ReLU activation to encode EM properties specific to a spatial region, but independent of the incidence direction and an MLP directional network with 1 hidden layer with 64 hidden units and ReLU activation which predicts the rotation a ray incident taking into account the direction.

The first network takes as inputs: f_i , a one-hot encoded identifier of the face where the relay point $x_k^{(r+1)}$ lies; n_i , the surface normal vector at that face, representing the geometric orientation of the surface; and, b_i , a 3D conditioning vector based on signed distances (sdf). These signed distance functions measure how far a given coordinate (e.g., the transmitter, receiver, or relay point) is from the face f_i , taking into account whether the point is inside or outside the face. This helps the network contextualize the interaction based on geometry.

The second network takes as input the spacial encoding v_i (output of the first network) and incorporates the direction of incidence $d_k^{(r)}$ (the direction of the incoming ray) to model how the ray interacts with the surface, including how much of it is reflected or transmitted, and how its direction changes.

The final output is scaling and additive coefficients s for the gain magnitude ($a_k^{(r+1)} = s_1 a_k^{(r)} + s_2$) and 4-dim parameters for rotation (based on Euler-Rodrigues formulation). The rotation parameters ρ_i are mapped to a 3×3 rotation matrix A to transform the incident to outgoing ray $d_k^{(r+1)} = A d_k^{(r)}$.

The new angles and time-of-flight have been calculated from the new direction. The new angle $\Phi_k^{(r+1)}$ describes the horizontal direction in which the ray travels after the interaction, measured in the XY plane, then it has been calculated using the arctangent function which gives the angle between the Y and X components of the direction vector. The new time-of-flight $\tau_k^{(r+1)}$ represents the time it takes for the ray to travel from the current point of interaction to the next point where it hits an object or reaches the receiver. It is a measure of propagation delay. To calculate it, the path of the ray is simulated using the Mitsuba renderer. For each predicted outgoing direction: a new ray is created starting from the interaction point; this ray is traced through the 3D scene using Mitsuba's *ray_intersect* method; if the ray intersects with an object, the distance along the ray to that point is recorded; and, this distance is then divided by the speed of light (3×10^8 m/s) to convert it into time. Only rays that intersect with valid surfaces are used; rays

that go off to infinity are ignored.

V. TRAINING

In this section, the implementation details are introduced in the training settings of this project.

In these experiments, K rays are initially launched omnidirectionally from the transmitter location, agnostic to the environment and location of the receiver location. For each ray, its interaction with the environment is evaluated. These data are separated into training and validation sets. Among them, about 85% are used for training, with the remaining 15% reserved for validation.

The MLP architecture models were coded in Python 3 (v3.10.6), using the PyTorch framework (v2.7.0), while Mitsuba 3 (v3.5.2) was employed for physically based rendering and ray intersection computations. Supporting tools include NumPy (v1.23.5) and SciPy (v1.15.2) (Rotation module) for quaternion operations.

The models were trained and tested on a laptop computer with a GPU environment NVIDIA GeForce RTX 2070 and 8 GB of RAM. Each of these models is trained in a supervised setting for 100 epochs with a learning rate of 0.001 and batch size of 1. Adam optimizer and the Mean Square Error (MSE) loss function for scalar-valued attributes and cosine distances loss function between angular attributes are utilized for received path loss optimization in this project. Set based Channel Loss compares two sets of multi-path channels: the predicted set and the ground-truth set. This comparison provides feedback to improve model training. To compare two paths the difference between each pair is measured. For scalar attributes, the differences directly are calculated whereas for angular attributes, the angular difference by treating the angles as unit vectors in Cartesian coordinates is measured and using a cosine-based distance.

This approach ensures that the loss considers both the matching of paths and the accuracy of their attributes, leading to meaningful guidance for training the model.

Training time for the model was approximately 4.70 seconds per epoch for indoor scenes and 0.03 seconds per epoch for outdoor scenes on an NVIDIA RTX 2070 GPU. The computational complexity scales linearly with the number of rays and interactions due to the feedforward architecture of the MLP. This efficiency enables the training of larger models or generalization to new environments using standard computational resources.

The inputs include normalized geometric features such as surface normals and signed distance functions, while face identifiers are one-hot encoded. The model architecture, described in Section IV, comprises two MLPs with ReLU activations, trained jointly to predict path direction and gain.

VI. EVALUATION METRIC

The evaluation metric serves as a quantitative measure to assess the performance of the proposed method in the prediction. In this work, absolute error metrics are used rather than relative errors, because some channel parameters, such

as gain and delay, can be close to zero, which would make relative errors unstable or less meaningful. Therefore, absolute errors provide a more reliable and interpretable assessment of the accuracy of the prediction. Three evaluation metrics are considered to evaluate this approach:

- **Overall prediction error ('Overall'):** This metric measures how well the entire set of predicted paths matches the set of ground-truth paths. To compare the two sets of paths (predicted vs. ground truth), correspondences between them are established using a linear sum assignment problem (also known as the Hungarian algorithm). This algorithm finds the best one-to-one matching between predicted and ground-truth paths that minimizes the total error. The final error considers all relevant path attributes, including gain, angles and time-of-flight. A lower value indicates better overall alignment between predicted and true multipath components.

$$\text{Err}_{\text{overall}} = \frac{1}{N} \sum_{k=1}^N \left(|a_k - \hat{a}_k| + |\tau_k - \hat{\tau}_k| + |\Phi_k - \hat{\Phi}_k| \right)$$

where:

- N is the total number of predicted paths.
- a_k and \hat{a}_k are the true and predicted gain of path k , respectively.
- τ_k and $\hat{\tau}_k$ are the true and predicted time-of-flight of path k .
- Φ_k and $\hat{\Phi}_k$ are the true and predicted angles of path k .
- **Geometry prediction error ('Geometry'):** This is a more focused version of the overall error. It still uses the matching mechanism from metric, but instead of evaluating all attributes, it specifically looks at two that describe the geometry of the path: angles and time-of-flight. This metric evaluates whether the predicted rays follow the same geometric routes as the ground truth, meaning they bounce off the same surfaces and follow similar trajectories between the transmitter and receiver. As with the overall error, lower values indicate better geometric consistency.

$$\text{Err}_{\text{geometry}} = \frac{1}{N} \sum_{k=1}^N \left(|\tau_k - \hat{\tau}_k| + |\Phi_k - \hat{\Phi}_k| \right)$$

where:

- N is the total number of predicted paths.
- τ_k and $\hat{\tau}_k$ are the true and predicted time-of-flight of path k .
- Φ_k and $\hat{\Phi}_k$ are the true and predicted angles of path k .
- **Average Delay Time - MAE ('AvgDelay'):** This metric summarizes the average time delay (τ) of all the predicted paths in a channel and compares it to the average delay of the ground-truth paths. For each path, its average time-of-flight is calculated and weighted by the linear power of the path. Then, the Mean Absolute Error (MAE) between

the predicted average delay and the true average delay is computed. Lower values here indicate that the temporal structure of the predicted channel closely matches the true one.

$$\text{Err}_{\text{avg_delay}} = \left| \frac{\sum_{k=1}^N a_k \cdot \tau_k}{\sum_{k=1}^N a_k} - \frac{\sum_{k=1}^N \hat{a}_k \cdot \hat{\tau}_k}{\sum_{k=1}^N \hat{a}_k} \right|$$

where:

- N is the total number of predicted paths.
- a_k and \hat{a}_k are the true and predicted linear gains of path k .
- τ_k and $\hat{\tau}_k$ are the true and predicted time-of-flight of path k .

VII. EXPERIMENTAL RESULTS

This section presents the performance evaluation of the proposed neural network based ray tracing model through experiments conducted in both indoor and outdoor environments. The model's ability is evaluated to predict key wireless propagation characteristics, including path gain, direction or time-of-flight, using the datasets generated with the Sionna ray tracing simulator. Performance metrics, as described in previous section, are used to evaluate the model's accuracy, generalization, and ability to learn complex interactions in realistic 3D scenarios.

Table III presents the quantitative evaluation of the model's performance in two distinct scenarios: a small indoor room and a large urban city environment. The results are reported across three metrics: Overall, Geometry and AvgDelay.

TABLE III. QUANTITATIVE RESULTS. COMPARING ERRORS OF THIS APPROACH IN TWO DIFFERENT SCENARIOS

Metrics	Small indoor room scene	Large urban city
Overall	2.374163	2.313505
Geometry	2.372437	2.312989
AvgDelay	0.001727	0.000516

The Overall error reflects the model's ability to predict complete ray paths accurately. The values are quite similar across both scenes (2.37 for the indoor scene vs. 2.31 for the city), indicating consistent overall performance regardless of scene complexity.

The Geometry error focuses specifically on the accuracy of the predicted ray geometry. Again, the model shows similar performance in both environments (2.37 vs. 2.31), suggesting that it effectively captures the geometric characteristics of the propagation paths.

The AvgDelay error, measured as MAE, shows a greater difference between scenes. The model achieves better delay prediction in the large urban city (0.000516) compared to the indoor room (0.001727). This may be attributed to the richer variety of multipath effects in urban settings, which enhance the model's ability to learn delay patterns effectively.

When compared to WiNeRT [2], the proposed model demonstrates competitive performance. Although WiNeRT achieves a superior geometry error of 0.084 in controlled

indoor settings, the AvgDelay error of this model (0.0005 in urban environments) is significantly lower than WiNeRT's best-case error of 0.828. These results underscore the model's strong predictive accuracy, particularly in complex outdoor scenarios with greater environmental variability.

A time performance evaluation is also proposed comparing this model to traditional ray tracing (Table IV).

TABLE IV. RUNTIME COMPARISON BETWEEN THE PROPOSED MODEL AND SIONNA RAY TRACING SIMULATOR

Dataset	Small indoor room scene	Large urban city
Runtime (neural model)	470.07 s	2.97 s
Runtime (Sionna ray tracing)	1.07 s	14.39 s

The runtime comparison between the proposed neural model and the traditional Sionna ray tracing simulator reveals interesting behavior across different environments. As shown in Table IV, the neural model achieves significantly faster inference in the large urban city scenario (2.97 seconds vs. 14.39 seconds for Sionna), demonstrating its potential for efficient large-scale deployment. However, in the small indoor room, the neural model exhibits a notably higher runtime (470.07 seconds compared to 1.07 seconds with Sionna). This disparity is likely due to the higher density of multipath reflections in indoor environments, increasing the computational load for the neural network.

When compared to RayProNet [7], which leverages continuous neural point-field representations for efficient runtime performance, the proposed model exhibits a more scene dependent behavior. RayProNet achieves inference times under 100s for complex indoor and outdoor environments, while the proposed model excels in outdoor scenarios with sparse geometries.

Overall, the proposed neural network based model demonstrates robust generalization across diverse environments, with notable strengths in predicting propagation delays and efficient runtime performance in large-scale urban settings. These findings suggest that the model is well-suited for applications requiring high accuracy and scalability in wireless environments.

VIII. CONCLUSION AND FUTURE WORK

In this work, a neural network based surrogate model for ray tracing in wireless communication environments has been presented. By learning how electromagnetic rays interact with 3D surfaces, this proposed model effectively predicts critical channel attributes such as gain, angle of departure/arrival, and time-of-flight. This approach has been validated using the Sionna ray tracing simulator in both indoor and outdoor settings, demonstrating consistent performance across different levels of scene complexity. Notably, this model shows strong generalization capabilities and achieves low error in average delay prediction, especially in urban environments where multipath effects are more diverse.

These findings suggest that neural ray tracing offers a scalable and efficient alternative to traditional physics-based simulators, with the potential for real-time or large-scale deployment in the context of 6G and digital twin technologies.

Despite the promising results, this study has several limitations that should be addressed in future work: simplified material properties, the materials used in the dataset have fixed relative permittivity and conductivity values. While this ensures consistency, it does not capture the variability found in real-world materials; limited scene diversity, only two scenes were considered for dataset generation. These environments, while representative of some scenarios, do not encompass the full range of conditions encountered in practical wireless communication system; exclusion of diffraction and scattering, the dataset was generated without accounting for diffraction and scattering effects. These phenomena, however, can significantly influence propagation characteristics, particularly in environments with sharp edges or complex surfaces; and, fixed frequency, all experiments were conducted at a single operating frequency of 3.5 GHz.

Future work could explore the scalability to larger and more diverse scenes, extend to additional propagation phenomena, and use different frequency ranges to further enhance the robustness and applicability of the model. In particular, future extensions of this work could target environments such as multi-floor indoor settings and dense urban areas with complex obstructions. The inclusion of additional physical effects like diffraction and scattering would improve modeling in scenarios with sharp edges and rough surfaces. Evaluating the model at other frequency bands, especially sub-THz ranges relevant to 6G, would further validate its generalizability. Moreover, integrating neural surrogates with physics-based modules could help balance efficiency with physical interpretability. Finally, online learning or reinforcement learning approaches could enable the model to adapt in real time within digital twin applications.

ACKNOWLEDGMENT

This work was supported by the program "Programa de Ayudas para la Realización de Proyectos de Investigación UAH" of the Vice-Rectorate for Research and Knowledge Transfer of the University of Alcala (Spain) through project PIUAH24/IA-076, and by the contract Programa Investigo ref. 04-UAH-INV2024 provided by the Comunidad de Madrid.

REFERENCES

- [1] T. K. Sarkar, *The physics and mathematics of electromagnetic wave propagation in cellular wireless communication*, 1st edition. 2018, ISBN: 9781119393139. [Online]. Available: <https://onlinelibrary.wiley.com/doi/book/10.1002/9781119393146>.
- [2] T. Orekondy *et al.*, "WineRT: Towards neural ray tracing for wireless channel modelling and differentiable simulations," in *The Eleventh International Conference on Learning Representations*, 2023. [Online]. Available: <https://openreview.net/forum?id=tPKKXW33YU>.

- [3] F. Aguado Agelet, A. Formella, J. Hernando Rabanos, F. Isasi de Vicente, and F. Perez Fontan, "Efficient ray-tracing acceleration techniques for radio propagation modeling," *IEEE Transactions on Vehicular Technology*, vol. 49, no. 6, pp. 2089–2104, 2000. DOI: 10.1109/25.901880.
- [4] Z. Ji, B.-H. Li, H.-X. Wang, H.-Y. Chen, and T. Sarkar, "Efficient ray-tracing methods for propagation prediction for indoor wireless communications," *IEEE Antennas and Propagation Magazine*, vol. 43, no. 2, pp. 41–49, 2001. DOI: 10.1109/74.924603.
- [5] D. He *et al.*, "The design and applications of high-performance ray-tracing simulation platform for 5g and beyond wireless communications: A tutorial," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 10–27, 2019. DOI: 10.1109/COMST.2018.2865724.
- [6] F. Saez de Adana, O. Gutierrez Blanco, I. Gonzalez Diego, J. Perez Arriaga, and M. Catedra, "Propagation model based on ray tracing for the design of personal communication systems in indoor environments," *IEEE Transactions on Vehicular Technology*, vol. 49, no. 6, pp. 2105–2112, 2000. DOI: 10.1109/25.901882.
- [7] G. Cao and Z. Peng, "Raypronet: A neural point field framework for radio propagation modeling in 3d environments," *IEEE Journal on Multiscale and Multiphysics Computational Techniques*, vol. PP, pp. 1–12, Jan. 2024. DOI: 10.1109/JMMCT.2024.3464373.
- [8] N. Purkait, *Hands-on neural networks with Keras : design and create neural networks using deep learning and artificial intelligence principles*, 1st edition, 2019, ISBN: 9781789533347. [Online]. Available: <https://github.com/PacktPublishing/Hands-On-Neural-Networks-with-Keras?tab=readme-ov-file>.
- [9] V. Renganathan, "Overview of artificial neural network models in the biomedical domain," vol. 120, pp. 536–540, Jul. 2019. DOI: 10.4149/BLL_2019_087.
- [10] E. Choi, M. T. Bahadori, A. Schuetz, W. F. Stewart, and J. Sun, *Doctor ai: Predicting clinical events via recurrent neural networks*, 2016. arXiv: 1511.05942 [cs.LG]. [Online]. Available: <https://arxiv.org/abs/1511.05942>.
- [11] A. Nguyen, K. Pham, D. Ngo, T. Ngo, and L. Pham, "An analysis of state-of-the-art activation functions for supervised deep neural network," *CoRR*, vol. abs/2104.02523, 2021. arXiv: 2104.02523. [Online]. Available: <https://arxiv.org/abs/2104.02523>.
- [12] S. Dutt, *Machine learning*, [First edition], 2019, ISBN: 9789353067373. [Online]. Available: <https://www.oreilly.com/library/view/machine-learning/9789389588132/>.
- [13] Y. Ge, L. Guo, and M. Li, "Physics-informed deep learning for time-domain electromagnetic radiation problem," in *2022 IEEE MTT-S International Microwave Biomedical Conference (IMBioC)*, 2022, pp. 114–116. DOI: 10.1109/IMBioC52515.2022.9790302.
- [14] L. Li *et al.*, "Deepnis: Deep neural network for nonlinear electromagnetic inverse scattering," *IEEE Transactions on Antennas and Propagation*, vol. 67, no. 3, pp. 1819–1825, 2019. DOI: 10.1109/TAP.2018.2885437.
- [15] Y. Jin *et al.*, *Sandwich: Towards an offline, differentiable, fully-trainable wireless neural ray-tracing surrogate*, 2025. arXiv: 2411.08767 [cs.NI]. [Online]. Available: <https://arxiv.org/abs/2411.08767>.
- [16] J. Knodt, S. Baek, and F. Heide, "Neural ray-tracing: Learning surfaces and reflectance for relighting and view synthesis," *CoRR*, vol. abs/2104.13562, 2021. arXiv: 2104.13562. [Online]. Available: <https://arxiv.org/abs/2104.13562>.
- [17] J. Zeng *et al.*, "Mirror-nerf: Learning neural radiance fields for mirrors with whitted-style ray tracing," in *Proceedings of the 31st ACM International Conference on Multimedia*, ser. MM '23, ACM, Oct. 2023, pp. 4606–4615. DOI: 10.1145/3581783.3611857. [Online]. Available: <http://dx.doi.org/10.1145/3581783.3611857>.
- [18] A. Alkhateeb, "Deepmimo: A generic deep learning dataset for millimeter wave and massive MIMO applications," *CoRR*, vol. abs/1902.06435, 2019. arXiv: 1902.06435. [Online]. Available: <http://arxiv.org/abs/1902.06435>.
- [19] E. Zhu, H. Sun, and M. Ji, *Physics-informed generalizable wireless channel modeling with segmentation and deep learning: Fundamentals, methodologies, and challenges*, 2024. arXiv: 2401.01288 [cs.IT]. [Online]. Available: <https://arxiv.org/abs/2401.01288>.
- [20] J. Hoydis *et al.*, *Sionna*, version 1.1.0, <https://nvlabs.github.io/sionna/>, 2022.
- [21] W. Jakob *et al.*, *Mitsuba 3 renderer*, version 3.1.1, <https://mitsuba-renderer.org>, 2022.
- [22] M. Abadi *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *CoRR*, vol. abs/1603.04467, 2016. arXiv: 1603.04467. [Online]. Available: <http://arxiv.org/abs/1603.04467>.
- [23] J. Hoydis *et al.*, "Sionna RT: Differentiable Ray Tracing for Radio Propagation Modeling," *arXiv preprint*, Mar. 2023.
- [24] Blender Online Community, *Blender - a 3d modelling and rendering package*, Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. [Online]. Available: <http://www.blender.org>.
- [25] Recommendation ITU-R, *Effects of building materials and structures on radiowave propagation above about 100 mhz p series radiowave propagation*, 2015. [Online]. Available: https://www.itu.int/dms_pubrec/itu-r/rec/p/R-REC-P.2040-1-201507-S!!PDF-E.pdf.
- [26] ETSI TR 138 901, *Study on channel model for frequencies from 0.5 to 100 ghz (3gpp tr 38.901 version 16.1.0 release 16)*, 2020. [Online]. Available: https://www.etsi.org/deliver/etsi_tr/138900_138999/138901/16.01.00_60/tr_138901v160100p.pdf.

A Lightweight Hybrid AI Framework for Cataract Detection Using Fundus Images: Real-World Evaluation on Clinical Data

Ishaan Kunwar

STEM Program

Edison High School

Edison, United States

e-mail: ishaankunwar19@gmail.com

Abstract—Cataract is one of the most prevalent eye diseases affecting the elderly population. In underserved regions, a low ophthalmologist-to-patient ratio and a scarcity of specialized medical devices pose challenges for early detection. This study aims to harness recent advancements in Deep Learning (DL) to automate cataract detection. Although numerous studies have been conducted in this area, improving model accuracy and minimizing overfitting, all while maintaining a simple architecture that requires fewer computational resources, remains challenging. This research proposes a hybrid method that merges featureization achieved by a Convolutional Neural Network (CNN) with classification techniques to improve prediction accuracy. The model's predictive performance is evaluated not only on the original test dataset but also on a newly acquired image set collected independently from a hospital. Experiments are conducted across different model architectures, such as CNNs and hierarchical Vision Transformers (ViTs) in combination with classifiers, such as multi-layer perceptron (MLP), K-nearest neighbors, and RandomForest. The highest accuracy is achieved using a combination of the ConvNeXtXLarge architecture for feature extraction coupled with a MLP classifier, reaching 92.3% on the original test dataset and improving to 94% on the new hospital-based dataset.

Keywords- cataract, convolutional neural network, vision transformer, multilayer perceptron.

I. INTRODUCTION

Cataracts are a predominant cause of visual impairment and blindness, accounting for approximately 33% of cases of impaired vision and 51% of first causes of blindness worldwide [1]. This eye ailment occurs due to the clumping of proteins in the lens, which significantly reduces its transparency. The ophthalmologists detect it by performing a manual retinal exam. They administer eye drops to dilate the pupil and then use the slit lamp, which is a specialized microscope with bright light, to clearly examine the retina for opacity.

Early detection of cataracts is crucial to preventing progressive blindness or avoiding costly surgical interventions, particularly in underserved regions where the ophthalmologist-to-patient ratio can be alarmingly low, often around 1 to 10,000.

Currently, cataract detection and diagnosis in hospitals are primarily based on clinical examinations by ophthalmologists using devices, such as slit lamps, ophthalmoscopes, and biomicroscopy. These methods involve direct visualization of the eyes' lens to assess opacity levels, and in some cases, specialized imaging techniques like ultrasound biomicroscopy and Scheimpflug imaging may also be employed. However, these procedures depend heavily on the availability of trained

medical professionals and advanced equipment, often resulting in significant delays in diagnosis and treatment initiation, particularly in underserved areas.

Fundus imaging, however, presents a simpler and more accessible alternative, particularly suitable for underserved regions. Fundus cameras are relatively portable, cost-effective, and easy to operate, requiring less specialized training compared to traditional ophthalmic diagnostic methods. These characteristics enable broader deployment, even in remote or resource-constrained settings, facilitating early detection and continuous monitoring of cataracts. Thus, leveraging fundus imaging could substantially improve the scalability and reach of cataract screening programs, particularly benefiting communities with limited healthcare infrastructure.

Recent advances in Artificial Intelligence (AI), particularly Deep Learning (DL), have shown promising potential for automating medical diagnostics across various domains, including ophthalmology. Deep learning-based systems, especially Convolutional Neural Networks (CNNs) and Vision Transformers (ViTs), have successfully demonstrated high performance in recognizing pathological conditions in ophthalmic imaging [2]. However, despite these successes, existing models often face critical challenges, including overfitting, excessive computational complexity and costs, and limited generalizability when exposed to datasets collected from diverse and independent clinical settings [3][4].

To address these issues, it is crucial to develop streamlined yet highly accurate models that are computationally efficient, generalizable, and robust to varied imaging conditions and demographic differences encountered in different regions. This paper proposes a hybrid approach that integrates deep feature extraction through advanced CNN architectures, specifically ConvXtnet-large, with traditional classification methods, including multi-layer perceptron (MLP), K-nearest neighbors (KNN), and RandomForest classifiers, with MLP being the chosen classification method. Evaluating the model performance not only on standard benchmarking datasets but also on independently acquired hospital datasets provides a more rigorous and realistic assessment of its generalization capabilities.

Such comprehensive validation across diverse datasets is essential for ensuring reliability and clinical applicability in underserved regions where disparities in healthcare accessibility demand robust, efficient, and accurate automated diagnostic tools. The developed approach aims to bridge gaps in oph-

thalmic care by providing a scalable, accessible, and precise cataract detection system.

The remainder of the paper is structured as follows: In Section II, the paper discusses prior work in the area of automating cataract detection using machine learning, including prior successes and limitations. Section III discusses the design of the experiment, including data collection, data preprocessing, feature extraction, architecture of classification models and of the proposed approach, and experimental procedures. In Section IV, the validation and testing results of the experiments are disclosed. In Section V, the results are discussed and explained and the limitations of the study are revealed. In Section VI, future prospects of the proposed method are detailed, and in Section VII, the paper is concluded and final thoughts are summarized.

II. RELATED WORK | METHODS

Several studies have leveraged Machine Learning (ML) and Deep Learning (DL) approaches to automate cataract detection. Typically, these methodologies involve three primary stages: data preprocessing, feature extraction, and classification. Traditionally, CNNs have dominated this domain due to their strength in image-based feature extraction. However, recently ViTs have gained significant traction, demonstrating promising results in ophthalmic disease diagnosis.

Multiple researchers have employed ViT-based methods with considerable success. Ali et al. [5] introduced a hyperparameter-optimized ViT model combined with Explainable AI techniques to diagnose various eye diseases from a diverse medical image dataset, achieving an accuracy of 91.40%. Similarly, Purba et al. [6] utilized a ViT architecture tailored for human eye disease classification, optimizing hyperparameters to attain an accuracy of 92.86% and recall of 85.72%. Another pertinent work by Gummadi et al. [7] implemented ViT for ocular disease classification, achieving an F1-score of 83.49%. Complementing these findings, Kumar et al. [8] conducted a comparative evaluation between traditional CNNs, specifically Visual Geometry Group-16 (VGG16) and ResNet50, and ViT on a consistent dataset, concluding that ViT demonstrated superior performance with an accuracy score of 70%.

Further advancements have been achieved through hybrid transformer models and specialized feature engineering approaches. Wang et al. [9] proposed a Transformer-based Knowledge Distillation Network (TKDNet) specifically tailored for cortical cataract grading. Their innovative methodology includes a zone decomposition strategy for extracting precise features and introduces specialized sub-scores addressing key clinical indicators, such as opacity location, area, and density. Their multi-modal mix-attention Transformer efficiently fused these sub-scores with image modalities, achieving a notable accuracy of 95.1% and recall of 81.6%.

Despite the growing popularity of ViTs, CNN-based methods remain highly relevant due to their computational efficiency and high accuracy. Khan et al. [10] successfully utilized a pre-trained VGG19 CNN model to detect cataracts from color fundus images, reaching accuracy and precision scores of 97.47%.

Lai et al. [11] developed a custom CNN architecture comprising seven layers—including convolutional, max-pooling, flatten, and dense layers for cataract detection from digital camera images, achieving outstanding accuracy and recall scores of 98.5% and 97.9%, respectively. Weni et al. [12] introduced a CNN-based method incorporating dropout regularization to mitigate overfitting, obtaining an accuracy of 88%. Further, Ganokratanaa et al. [13] compared a LeNet-based CNN to a traditional Support Vector Machine (SVM) classifier, with their LeNet-CNN approach yielding an impressive 96% accuracy.

While significant progress has been made in automating cataract detection, several challenges persist. Critical areas for future research include enhancing prediction accuracy, minimizing model overfitting and computational costs, and improving generalizability by testing the model on different geographical locations.

This paper proposes a hybrid, computationally efficient approach that integrates deep feature extraction through advanced CNN architectures, specifically ConvNeXtXLarge, with traditional classification methods, including Multi-Layer Perceptron (MLP) [14], K-Nearest Neighbors (KNN) [15], and RandomForest classifier [16]. Evaluating the model performance not only on standard benchmarking datasets but also on independently acquired hospital datasets provides a more rigorous and realistic assessment of its generalization capabilities.

III. METHODS AND MATERIALS

A. Dataset

This study uses the Ocular Disease Intelligent Recognition (ODIR) dataset [17] from Shangong Medical Technology Co., Ltd. It is a structured ophthalmic database of 5,000 patients with age, color fundus photographs from left and right eyes and doctors' diagnostic keywords from doctors. This dataset represents a real life set of patient information collected by Shangong Medical Technology Co., Ltd. from different hospitals and medical centers in China. It has images of normal eyes and images of eyes with cataract. It is then randomly divided into three parts, with 80% of the images being used for training, 10% for validation and the remaining 10% for testing the accuracy of the model. These three datasets contain nearly equivalent numbers of normal and cataract eye images. This study also utilizes additional fundus images of normal (40 patients) and cataract (10 patients) eyes obtained from GSVM Medical College, which is a public medical college in Kanpur, India. This dataset represents a real life set of fundus images of Indian patients. It is used only as a testing dataset to assess the generalizability of the model trained on the ODIR dataset. Figure 1 shows the retinal fundus images of normal and cataract eyes. Both the ODIR and GSVM datasets used in this work are publicly available and fully comply with Health Insurance Portability and Accountability Act (HIPAA) in protecting patients' health information.

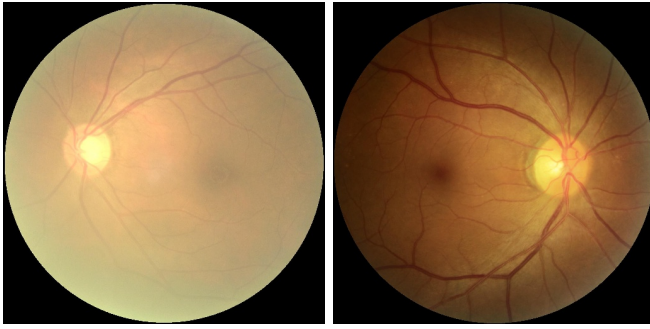


Figure 1. Eye Fundus Images. (a) Cataract. (b) Normal.

B. Data Preprocessing

A pre-processing stage is crucial in standardizing the input of fundus images obtained from multiple sources, as they may have different characteristics. The ODIR dataset is from several medical institutions in China where fundus images are captured by various cameras in the market, such as Canon, Zeiss and Kowa, resulting in varied image resolutions. The GSVM dataset of raw images contained irrelevant visual components like computer monitors and medical devices, which are cropped out to have only retinal fundus images. Each fundus image is then resized to 224x224 pixels for uniformity and then converted to RGB color space resulting in a three-dimensional (3D) array of 224x224x3. The three 2D arrays represent red, green and blue channels respectively. It is then converted from RGB to BGR and each color channel is zero-centered with respect to the ImageNet dataset, without scaling, as required by ConvNeXtXLarge.

C. Feature Extraction

ConvNets have always been popular for computer vision related tasks due to their inherent inductive biases like translation equivariance and sliding window strategy. Translation equivariance is important for object detection and sliding window strategy allows neighbors to share computations, which is essential for visual processing. Recently, ViTs have entered this space with better accuracy rate than traditional ConvNets and are getting increasingly dominant. The primary reason for their superiority is their global attention design, which has quadratic complexity with respect to the input image size and can quickly become unmanageable with higher resolution images. To address this limitation, hierarchical Transformers like Swin Transformer have been developed, which incorporates some of the inductive biases of ConvNets like sliding window strategy. But the resulting design is still complex, requiring significantly more computational resources than ConvNets. ConvNeXtXLarge model [18] is an enhanced traditional ConvNets, which retains the design simplicity of convolution and then incorporates features like depthwise convolution, inverted bottleneck and large kernel sizes taken from the architecture of hierarchical Transformers. It outperforms vanilla ViTs, while maintaining the simplicity and efficiency of standard ConvNets and for these reasons is used here to extract image features.

The weights used are from the pretraining of this model on the ImageNet-21k dataset and then fine-tuned on the ImageNet-1k dataset. The top layer of the ConvNeXtXLarge model is replaced with a global average pooling layer to avoid overfitting. It also ensures that some spatial information is retained by averaging each feature map, which allows for higher versatility across different input variations evident in datasets of this study. It also helps in keeping the architecture simple, which leads to faster featurization and less computational resource consumption. Essentially, the idea was that initially featurizing the images and then classifying them based on these numerical features would provide better accuracies than solely applying a CNN-variant. By stripping off the classifier head of the ConvNeXtXLarge model, the model extracts 2048 features from the images. As opposed to the classifier head making the prediction, additional predictive models were added on the ConvNeXtXLarge model to improve accuracy. The set of features extracted from the dataset are randomly reshuffled to avoid subsequent classification models from learning the patterns based on the order of the images in the dataset.

In order to determine the importance of the ConvNeXtXLarge featurizer in the proposed pipeline, an ablation study was conducted. One of the variants of the pipeline tested in the ablation study involved substituting the ConvNeXtXLarge model for DenseNet-201, another featurizer model, pairing DenseNet-201 with MLP. A DenseNet-201 model, pretrained on the ImageNet-1k dataset, is a CNN that has dense connectivity, meaning that each layer takes input from all the layers that were before that particular layer and provides output to all layer subsequent to that particular layer. The model has several dense blocks, with each block containing a certain amount of layers. Each block is separated from other blocks by transition layers, which are composed of a 1 by 1 convolution layer followed by a pooling layer, with the goal being to compress the feature maps. Due to these characteristics of the DenseNet-201 model, it has several advantages, such as reducing the occurrence of vanishing gradients and cutting down on parameter redundancy. The DenseNet-201 model extracts 1920 numerical features from the fundus images and also has its top layer stripped away, replaced with a global average pooling layer for similar reasons as the ConvNeXtXLarge model. The reason for choosing DenseNet-201 as the substitute for ConvNeXtXLarge in this ablation study lies in the fact that DenseNet-201, being a classic CNN, lacks the ViT-like enhancements that ConvNeXtXLarge possesses, such as inverted bottleneck, depthwise convolution, and large kernels. The ablation study, in part, aims to determine the effect of removing ViT-like enhancements on the performance of the model.

D. Classification

The extracted features are then used to train MLP, KNN, and Random Forest classifiers. The MLP Classifier is a feedforward neural network having at least three layers, an input layer, one or more hidden layers, and an output layer. Each node in the input layer corresponds to a feature in the feature map.

There can be any number of hidden layers and each can have any number of nodes. They calculate a weighted sum of the inputs followed by an activation function, which adds bias and introduces nonlinearity. The output layer generates the final prediction, so in this study acts as the binary classifier having two nodes for "Normal" and "Cataract" prediction. The predicted output is compared to the actual label using a loss function and to minimize its value, weights of the nodes and activation function are adjusted during backpropagation.

The KNeighborsClassifier is a simple yet powerful lazy learning algorithm. It preserves the entire training data from the training phase and uses it to classify based on similarity measures. The class of a data point is determined by the majority or average of its K neighbors, which are found based on a distance metric.

The RandomForest Classifier is an ensemble tree learning algorithm. During the training phase, it creates a number of decision trees using random subsets of the features from the feature map. Each individual tree makes its prediction and the final prediction is determined by voting where the most frequently predicted result is chosen. These three different classifiers are paired with ConvNextXLarge and compared in an ablation study to isolate the contributions of the MLP classifier in the proposed pipeline and to assess its individual importance to the performance of the proposed pipeline.

Figure 2 demonstrates the architecture of the proposed method, including preprocessing, featurization, and the different classification models that ConvNeXtXLarge is paired with.

In order to evaluate the performance of the proposed methodology, the results have been compared with traditional CNN models like ResNet50 [19], EfficientNetb2 [20], and MobileNetv2 [21] as well as computationally costly ViTs, such as Swin transformer [22] and vanilla ViT [23].

ResNet50 is a type of Deep Convolutional Neural Network (DCNN) with 50 layers, a part of a group called Residual Networks. It uses special connections, known as residual connections, to help gradients flow effectively and overcome issues like vanishing gradients, making it reliable for tasks like image classification. EfficientNetB2 is a CNN that enhances width, depth, and resolution of images through a calculated scaling method. As the third model in the EfficientNet series, it expands upon previous models (B0 and B1) with more layers, wider channels, and higher resolution. This allows EfficientNets to achieve high accuracy with fewer parameters and less computational demand compared to older models like ResNet. ViTs break down images into small, equal sized patches, flatten these patches, and feed them into a global attention module, using positional embeddings for each patch. This approach allows ViTs to focus on broad patterns in images, often surpassing traditional CNNs, especially in large-scale datasets. SwinTransformers are a specialized version of ViTs that apply attention mechanism within local windows that are shifted across the image. By incorporating convolutional layers, they create hierarchical feature maps similar to those in CNNs, which helps them be effective in classification tasks.

E. Experimental Procedures

For training MLP, the hyperparameter `learning_rate_init`, which controls the step size in updating the weights during backpropagation to minimize loss function, is evaluated for values 0.01, 0.05, 0.001, 0.0001, 0.00001, 0.1 and 0.000001. For each value of `learning_rate_init`, the hyperparameter `max_iter`, which is the epoch value, is evaluated for values ranging from 10 to 110 increasing in intervals of 10.

For KNN, the hyperparameter `n_neighbors`, which is the number of nearest neighbors to consider in deciding the class of a data point, is evaluated for values ranging from 1 to 15.

For RandomForest, the hyperparameter `max_depth`, which is the depth of the decision tree, is evaluated for values in the range 1 to 7. For each value of `max_depth`, the hyperparameter `n_estimators`, which is the number of the decision trees in the forest, is evaluated for values ranging from 10 to 110.

The performance of the proposed methodology is also benchmarked against the MobileNetV2, ResNet50, EfficientNetB2, SwinTransformer, ViT model, which has a similar architecture, to assess improvements in detection accuracy, highlighting the effectiveness of the featurization over classification methods in enhancing cataract prediction performance. The hyperparameter epochs values ranging from 10 through 50 with the `learning_rate` hyperparameter ranging from 0.000001 to 0.05, depending on the model, are used to fine-tune the models.

Performance metrics like accuracy, precision and recall are used to identify the most effective model. Accuracy is the percentage of true prediction out of the total prediction. Precision is the percentage of true positive prediction (i.e., cataract eye) out of total positive prediction. Recall is the percentage of true positive prediction out of the total positive samples. For tasks like medical diagnosis, the cost of false negative prediction (i.e., cataract eye predicted as normal) is the highest, so a higher recall value is given the highest precedence followed by precision and then accuracy.

Finally, the computational efficiency and speed of the proposed pipeline was quantified by measuring the wall time (s) as well as CPU time (s) of the entire pipeline, including both featurization and classification, during training, validation, and testing. Wall time is the elapsed time from when the task began to when it ended, taking into account computation of the models, waiting for inputs and outputs, network delays, and several other real world factors to represent the real world time that a user must wait for the result. On the other hand, CPU time is the amount of time that the computer processing spent executing the pipeline. For the validation, testing, and collected testing dataset, two additional performance metrics were measured: inference latency (s) and throughput (samples/s). Inference latency is the total elapsed time it takes a trained ML model to take in a singular input, in this case a fundus image, and make a prediction. Inference latency is critical in this context as it measures how long a user may have to wait for a diagnosis for cataract, with speed being essential to lessen the impact of cataract. Throughput, which is mathematically the inverse of inference latency, measures the amount of predictions

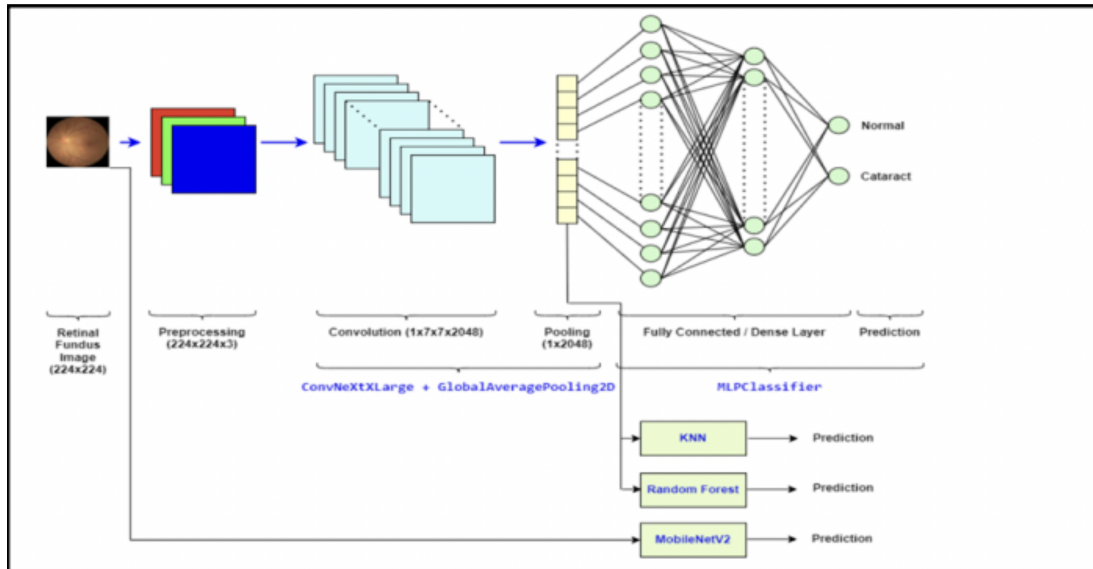


Figure 2. Architecture of proposed hybrid approach.

a trained ML model can make within a certain unit of time. Throughput is essential in this context as it measures the efficiency of the pipeline, which is essential when processing batches of patients, a common occurrence in overwhelmed and understaffed rural clinics.

IV. RESULTS

In the case of MLP, the highest accuracy of 98.28% is achieved, with minimum number of epochs, when epochs are 40 and learning rate is 0.01. In Figure 3, the performance of MLP based on different pairs of epochs and learning rates is shown. Other combinations of epochs and learning rate result in this accuracy, including 50 epochs and 0.01 learning rate, 60 epochs and 0.01 learning rate, 80 epochs and 0.01 learning rate, and 70 epochs and 0.05 learning rate. This is the highest validation accuracy, higher than every model except RandomForest.

In the case of KNN, the highest accuracy of 96.55% is achieved with 2 neighbors and the lowest accuracy of 89.50% is achieved when the number of neighbors increases to 5, 6, 7, and 13. This performance is demonstrated by Figure 4a based on different numbers of neighbors. For RandomForest, the highest accuracy of 98.28% is achieved with 20 trees and a depth of 2. This performance is demonstrated by Figure 4b based on different pairs of trees and depths.

In the ablation study, which is documented in Table I, the combination of ConvNextXLarge and MLP yielded an accuracy of 98.28% in the validation dataset, 92% in the testing dataset, and 94% in the collected GSVM testing dataset. These were the highest recorded accuracies out of all combinations of components tested in the ablation study. The combination of ConvNextXLarge and KNN yielded a testing accuracy of 90.40% and a GSVM testing accuracy of 75.50%. The combination of ConvNextXLarge and RandomForest yielded a testing accuracy of 88.50% and a GSVM testing accuracy of

79.60%. The combination of DenseNet201 and MLP yielded a validation accuracy of 93.10%, a testing accuracy of 86.30%, and a GSVM testing accuracy of 91.80%.

TABLE I. ABLATION STUDY ACCURACY RESULTS

Variant	Val.(%)	Test(%)	GSVM(%)
ConvNextXLarge+MLP	98	92	94
ConvNextXLarge+KNN	97	90	76
ConvNextXLarge+RF	98	89	80
DenseNet-201+MLP	93	86	92

For each classification model paired with ConvNeXtXLarge, each of the graphs depicting their validation accuracy has learning rate on the horizontal axis, validation accuracy on the vertical axis, and each line represents a different epoch number in the sequence 10, 20, 30, 40, and 50. The performance metrics of each model on the testing portion of the ODIR dataset as well as on the collected GSVM dataset are depicted in Table II and Table III, respectively.

For EfficientNetB2, according to Figure 5a, the highest validation accuracy was 96.55% at 30 epochs and 0.01 learning rate. For ResNet50, according to Figure 5b, the highest validation accuracy was 98.28% at 10 epochs and 0.001 learning rate. For MobileNetv2, according to Figure 5c, the highest validation accuracy was 94.83% at 30 epochs and 0.005 learning rate.

Moving on to ViTs, for Swin Transformer, the highest validation accuracy, according to Figure 6a was 94.83% at 30 epochs and 0.0001 learning rate. Finally, for the vanilla ViT, according to Figure 6b the highest validation accuracy was 98.04% at 50 epochs and 0.000001 learning rate.

In Table II and Table III, the performance of the classification models on ODIR and GSVM test dataset is summarized, respectively. The testing of the ODIR test dataset on the MLP model resulted in an accuracy score of 92.30%. The same

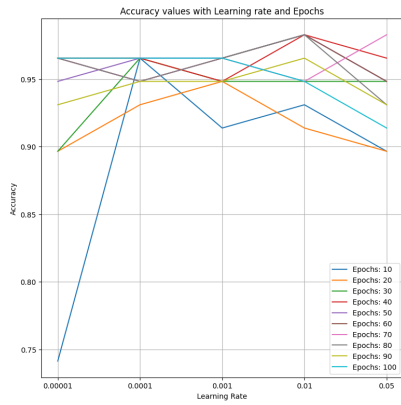
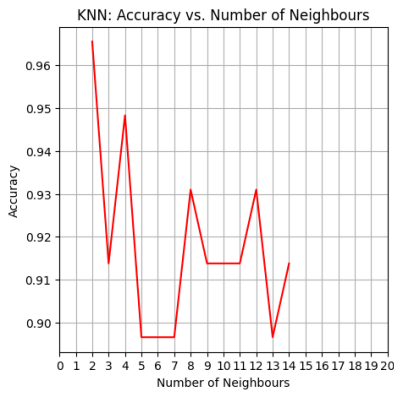
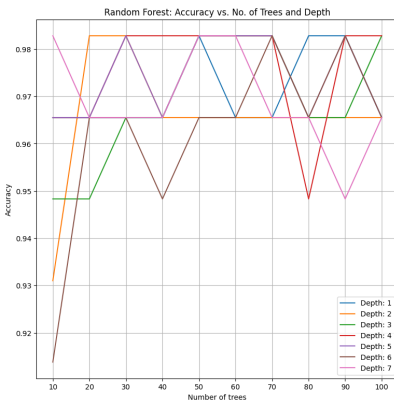


Figure 3. Hyperparameter tuning of the MLP model.

model is also tested using the GSVM test dataset resulting in an accuracy of 94%. Each of these is depicted by Figures 7a and 7b, respectively.



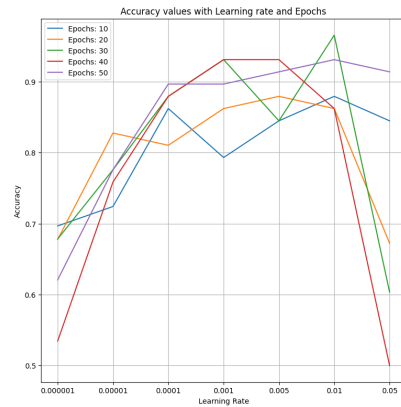
(a) K-Nearest Neighbors



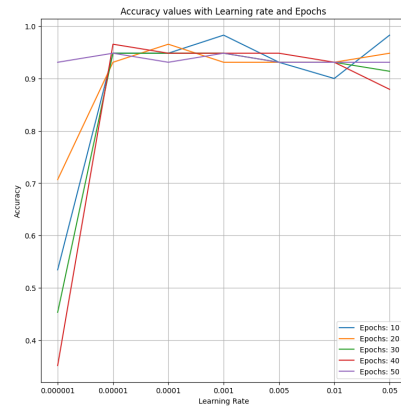
(b) RandomForest

Figure 4. Validation accuracy of ML models across their parameters.

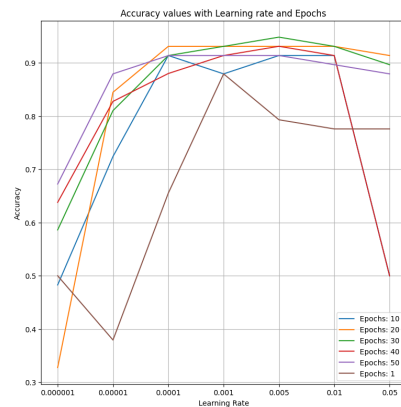
The computer system performance metrics for the proposed pipeline when applied to each dataset were measured to quantify its computational efficiency. For the training dataset, the recorded CPU time was 2 hours, 43 minutes, and 43 seconds. The recorded wall time was 2 hours, 17 minutes, and 30 seconds. The wall time of the validation dataset was 21



(a) EfficientNetB2 model

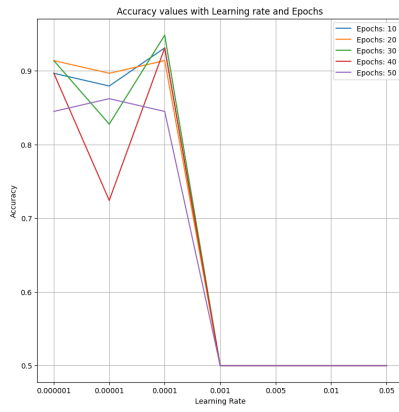


(b) ResNet50 model

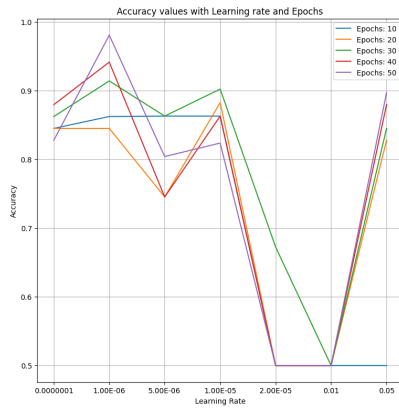


(c) MobileNetV2 model

Figure 5. Validation accuracy of CNN models across learning rates and epochs.



(a) Swin Transformer



(b) Vanilla ViT

Figure 6. Validation accuracy of ViT models across their parameters.



Figure 7. Performance of proposed model on testing datasets.

minutes and 5 seconds, yielding an average inference latency, for the validation dataset, of 21.8 seconds per fundus image and a throughput of 0.046 fundus images per second. For the testing dataset, the wall time was 18 minutes and 9 seconds, yielding an average inference latency of 21.4 seconds per fundus image and a throughput of 0.047 fundus images per second. For the collected GSVM testing dataset, the wall time was 15 minutes and 4 seconds, yielding an inference latency of 18.4 seconds per fundus image and a throughput of 0.054 fundus images per second. All these measurements are referenced from Table

TABLE II. PERFORMANCE OF CLASSIFICATION MODELS ON ODIR TEST DATASET

Model	Acc.(%)	Prec.(%)	Rec.(%)	F1.(%)
MLP	92	92	92	92
ResNet50	90	90	90	90
EfficientNetB2	92	92	92	92
ViT	90	92	90	90
Swin Trans.	92	92	92	92
MobileNetV2	86	86	86	86

TABLE III. PERFORMANCE OF CLASSIFICATION MODELS ON GSVM TEST DATASET

Model	Acc.(%)	Prec.(%)	Rec.(%)	F1.(%)
MLP	94	97	83	88
ResNet50	92	93	92	92
EfficientNetB2	94	95	94	94
ViT	86	93	61	64
Swin Trans.	96	93	93	93
MobileNetV2	90	94	72	78

IV below.

TABLE IV. COMPUTER SYSTEM PERFORMANCE METRICS OF PROPOSED PIPELINE

Dataset	Wall time(s)	CPU time(s)	Inference Latency(s)	Throughput
Train	8250	9823	-	0.058
Val.	1079	1275	21.8	0.046
Test	850	1089	21.4	0.047
GSVM	904	1059	18.4	0.054

V. DISCUSSION | EVALUATION

The combination of ConvNeXtXLarge and GlobalAveragePooling2D for featurization with MLP Classifier as the classifier resulted in the highest validation accuracy, which was higher than similar prior studies, albeit with a different and smaller dataset. Convolution does not have high computational costs like global attention design of ViTs, which requires computationally expensive global attention modules. Additionally, just one hidden layer with 100 nodes in MLP classifier also helps in keeping the architecture simple, requiring less resources. KNN and RandomForest also had decent validation accuracies, with RandomForest actually matching MLP in validation accuracy. However, KNN and RandomForest require comparatively more resources than the proposed method.

For KNN, more neighbors after K=2 results in a drop in accuracy. This is likely due to the bias-variance tradeoff involved with involving more neighbors. In the case of RandomForest, there is not much variation in accuracy for different combinations of depth and number of trees. Every evaluated combination results in the accuracy within the range of 91.38% to 98.28%. It is also evident during tuning that computational requirements of this classifier are directly proportional to the number of trees.

In the case of MLP, the highest accuracy of 98.28% is achieved when epochs are 40 and learning rate is 0.01. The accuracy increases with the increase in epochs for every single value of learning rate. After an optimal value of epochs is reached, the accuracy plateaus for each learning rate. The lowest learning rate of 0.00001 has the maximum deviation in accuracy ranging from around 74% to 97% as epochs increase. The highest learning rate of 0.05 has second highest deviation in accuracy ranging from around 89% to 98% with increase in epochs. It demonstrates that too low or too high learning rate during backpropagation can adversely impact accuracy. For higher learning rates, such as 0.05, weights are updated too quickly, resulting in oscillations around convergence and thus making number of epochs cause large variation in validation accuracies. However, since 0.05 as a learning rate is not too large, the deviation in accuracy based on epochs is not too drastic and maximum accuracy of 98.28% is still achievable with this learning rate when paired with enough epochs, in this case 70.

In the ablation study, given that the combination of ConvNextXLarge as the featurization model and MLP as the classifier model yielded the highest accuracies in classifying the fundus images among all three of the datasets (validation, testing, and GSVM testing), clearly, both of these models contribute heavily to the performance of the pipeline. Likely, ConvNextXLarge and MLP outperformed the pairing of ConvNextXLarge and KNN because when there is a large number of features involved, such as 2048 features, the Euclidean distances that KNN calculates between points to classify a point tend to concentrate as close neighbors and far away neighbors appear to be roughly equidistant from the datapoint currently being classified, thus making KNN's prediction inaccurate. MLP likely trumped RandomForest's performance as well since RandomForest uses the aggregate prediction of axis-aligned decision trees, which perform optimally on tabular data but often stumble when trying to classify an image based on the smooth, high-dimensional feature maps produced by CNNs such as ConvNextXLarge. The ConvNextXLarge model likely outperformed the DenseNet-201 model due to its ViT-like enhancements, such as depthwise convolutions and inverted bottleneck, which boosted its accuracy without requiring computationally expensive global attention modules.

The evaluation of various models, including MobileNetv2, ResNet50, EfficientNetB2, ViT, and SWINTransformer, across different learning rates and epochs shows the impact of hyperparameter tuning on model performance. Lower learning rates frequently demonstrate slower convergence, leading to suboptimal accuracy, as observed with MobileNetv2 and similarly noted in other models like EfficientNetB2 and ViT. In contrast, moderate learning rates, particularly around 0.0001 to 0.005, consistently yield higher stability and precision, resulting in efficient convergence without the risk of overshooting the global minimum, a pattern evident across both ResNet50 and MobileNetv2. High learning rates, such as 0.05, introduce significant volatility, destabilizing the training process as illustrated by diminished results in MobileNetv2, ViT, and

EfficientNetB2.

For the ODIR testing performance of each model, the proposed hybrid approach of pairing ConvNeXtXLarge to featurize fundus images and using MLP to classify each image based on these features got the highest accuracy of 92% and highest recall of 92%. These performance metrics were matched by Swin Transformer as well as EfficientNetB2. The proposed approach likely performed one of the best because ConvNeXtXLarge leverages the strengths of both the CNNs (ConvNet / CNN) and hierarchical ViTs for featurization. The inherent inductive biases of CNN, like translation equivariance and sliding window strategy, work together with the depthwise convolution and inverted bottleneck of ViTs to extract image features. Thus, strong spatial representations are fed into MLP, which allows MLP to make accurate predictions.

EfficientNetB2 also shared the same high performance metrics due to its compound scaling of fundus images, which effectively scales the width, depth, and input resolution of inputted images using a user-specified scaling coefficient. This scaling allows it to capture finer details and improve representation of images. For Swin Transformer, it first splits the images into patches that it then flattens into feature vectors. By applying self-attention to small local windows that are then shifted across the image to ensure cross-window communication, the model is capable of paying attention to local features as well as maintain global awareness, thus allowing it to notice small features in the fundus images and generalize better on new datasets.

On the GSVM dataset, the differences between each of these three respective model were more. While the hybrid proposed approach, EfficientNetB2, and Swin transformer had similar accuracies despite the proposed approach's simplified architecture, Swin transformer and EfficientNetB2 had higher recall values than the proposed approach did. This is likely because the MLP head, with only 100 nodes in one hidden layer, was unable to properly detect all positives, hence its lower recall. The model likely requires more training on noisy real world hospital data to be able to properly generalize to real world datasets and their quality issues. The hierarchical window-based self-attention of Swin Transformer and the compound scaling of EfficientNetB2 likely allowed each model to notice small details in fundus images of the training dataset, thus allowing them to generalize to hospital data even with its flaws.

The vanilla ViT likely performed much worse than the others because ViTs, due to their global attention modules, require large training datasets to properly generalize to other datasets and often miss small localized details. MobileNetv2 is a lightweight model that does not translate well to datasets that have a lot of noise. While the proposed approach has a lower recall than Swin Transformer and EfficientNetB2 on real-world hospital data due to the relatively low quality of the data, the proposed approach still performs on par with, and sometimes better than (in the case of validation dataset) state of the art models on quality datasets that resemble its training dataset, and further training on more real world hospital data will likely allow it to generalize to imperfect hospital data.

Additionally, since MLP has only one layer with 100 nodes and ConvNeXtXLarge lacks computationally expensive attention-based modules, the proposed approach is computationally more efficient than other models, including vanilla ViTs with their global attention modules. Finally, the proposed approach, due to the replacement of ConvNeXtXLarge's FTC with a Global Average Pooling Layer, is able to reduce overfitting of the model on the training dataset.

Considering the computer system performance metrics for the proposed pipeline, the inference latency times, computed for each of the three non-training datasets, are roughly similar to each other. Taking into account the relative number of images in each of the three datasets, the average inference latency is 20.6 seconds, which is much faster than the standard 30-60 minutes a standard cataract examination may take involving a specialist and advanced equipment. For each dataset, the CPU time exceeded the wall time, regardless of the other delays that the wall time considers. This is because of the use of parallel processing when running the pipeline, using multiple cores at the same time as opposed to a single-threaded process.

The limitations of this study include a lack of large datasets to train classifier models and the local nature of the datasets. The study uses ODIR dataset for training, which has just a few hundred fundus images for cataract as opposed to thousands of images typically required to train models efficiently and avoid overfitting. Both the ODIR and GSVN fundus image datasets are of patients from south Asia. It is not clear if the accuracy of the model will be the same if tested on fundus image datasets from other parts of the world. Further study and more diverse sources of datasets are required to address these aspects.

VI. CONCLUSION AND FUTURE WORK

This study establishes a robust framework for cataract detection using deep learning and traditional classifiers, showcasing strong performance on both benchmark and hospital-based datasets. The few seconds that it takes the web app to predict the presence of cataract from a fundus image is much faster than skilled medical personnel, using advanced detection equipment, would be able to without even considering the fact that these personnel can only visit a clinic once every few weeks or even months due to understaffing. Nonetheless, there remain several promising avenues for future research. Expanding the dataset size with diverse fundus images from various geographic locations will help improve the generalizability and robustness of the model across different populations. Moreover, incorporating techniques like transfer learning from larger ophthalmologic datasets or integrating advanced data augmentation methods could further mitigate overfitting and improve performance.

Additionally, incorporating explainability methods and visualization tools to interpret model predictions could provide clinicians greater confidence in AI-assisted diagnoses, promoting better clinical acceptance and decision-making.

Integration of multimodal data, combining fundus imaging with other diagnostic modalities, such as Optical Coherence

Tomography (OCT) or clinical patient histories, could further enhance diagnostic accuracy and reliability. Furthermore, longitudinal studies assessing the real-world clinical impact and economic feasibility of deploying this AI-based cataract detection system will be crucial to translating research advancements into practical healthcare improvements.

With regards to the use of this web app by clinicians, a possible improvement to the web app could be "Clinician-in-the-Loop Testing," where a clinician could participate in the predictions made by the web app by having the web app identify certain fundus images that it is unsure of and thus passing them off to the clinician for a more detailed review. The rate at which clinicians accept or reject the predictions of the web app could also be recorded as another metric for performance. Finally, efficiency benchmarks can be used to demonstrate the efficiency and speed of the model on different hardware. For instance, a possible benchmark is throughput, which is the number of fundus images that can be classified within a certain amount of time. Other measures, such as CPU usage, memory consumption, and power draw of the web app when predicting can also be measured. These benchmarks are heavily affected by the quality of hardware used. Since rural clinics will often be limited to hardware with limited computing power, these metrics help further quantify the computing resources that the pipeline may require to ensure that it does not exceed computational limits.

REFERENCES

- [1] D. Pascolini and S. Mariotti, "Global estimates of visual impairment: 2010", *The British journal of ophthalmology*, vol. 96, pp. 614–8, Dec. 2011. DOI: 10.1136/bjophthalmol-2011-300539.
- [2] V. Gulshan *et al.*, "Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs", *jama*, vol. 316, no. 22, pp. 2402–2410, 2016.
- [3] Y. Bao *et al.*, "Self-adaptive transfer learning for multicenter glaucoma classification in fundus retina images", in *Ophthalmic Medical Image Analysis: 8th International Workshop, OMIA 2021, Held in Conjunction with MICCAI 2021, Strasbourg, France, September 27, 2021, Proceedings 8*, Springer, 2021, pp. 129–138.
- [4] S. Lu *et al.*, "Deep learning-driven approach for cataract management: Towards precise identification and predictive analytics", *Frontiers in Cell and Developmental Biology*, vol. 13, p. 1611216, 2025.
- [5] M. S. Ali and M. Islam, "A hyper-tuned vision transformer model with explainable ai for eye disease detection and classification from medical images", *BS thesis, Faculty of Engineering and Technology Islamic University*, 2023.
- [6] S. O. Purba *et al.*, "Classification of eye diseases in humans using vision transformer architecture model", in *2024 International Conference on Information Technology Research and Innovation (ICITRI)*, IEEE, 2024, pp. 71–75.
- [7] S. D. Gummadi and A. Ghosh, "Classification of ocular diseases: A vision transformer-based approach", in *International Conference on Innovations in Computational Intelligence and Computer Vision*, Springer, 2022, pp. 325–337.

- [8] D. Kumar, B. Bakariya, C. Verma, and Z. Illes, "Cataract disease identification using transformer and convolution neural network: A novel framework", in *2023 3rd International Conference on Technological Advancements in Computational Sciences (ICTACS)*, IEEE, 2023, pp. 1230–1235.
- [9] J. Wang *et al.*, "A transformer-based knowledge distillation network for cortical cataract grading", *IEEE Transactions on Medical Imaging*, vol. 43, no. 3, pp. 1089–1101, 2023.
- [10] M. S. M. Khan, M. Ahmed, R. Z. Rasel, and M. M. Khan, "Cataract detection using convolutional neural network with vgg-19 model", in *2021 IEEE World AI IoT Congress (AIoT)*, IEEE, 2021, pp. 0209–0212.
- [11] C.-J. Lai *et al.*, "The use of convolutional neural networks and digital camera images in cataract detection", *Electronics*, vol. 11, no. 6, p. 887, 2022.
- [12] I. Weni, P. E. P. Utomo, B. F. Hutabarat, and M. Alfalah, "Detection of cataract based on image features using convolutional neural networks", *Indonesian Journal of Computing and Cybernetics Systems*, vol. 15, no. 1, pp. 75–86, 2021.
- [13] T. Ganokratanaa, M. Ketcham, and P. Pramkeaw, "Advancements in cataract detection: The systematic development of lenet-convolutional neural network models", *Journal of Imaging*, vol. 9, no. 10, p. 197, 2023.
- [14] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors", *nature*, vol. 323, no. 6088, pp. 533–536, 1986.
- [15] T. Cover and P. Hart, "Nearest neighbor pattern classification", *IEEE transactions on information theory*, vol. 13, no. 1, pp. 21–27, 1967.
- [16] L. Breiman, "Random forests", *Machine learning*, vol. 45, pp. 5–32, 2001.
- [17] G. Challenge, *Peking university international competition on ocular disease intelligent recognition (odir-2019)*, 2019.
- [18] Z. Liu *et al.*, "A convnet for the 2020s", in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 11 976–11 986.
- [19] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
- [20] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks", in *International conference on machine learning*, PMLR, 2019, pp. 6105–6114.
- [21] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.
- [22] Z. Liu *et al.*, "Swin transformer: Hierarchical vision transformer using shifted windows", in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 10 012–10 022.
- [23] A. Dosovitskiy *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale", *arXiv preprint arXiv:2010.11929*, 2020.