FastDDS-Based Middleware System for Remote X-Ray Image Classification Using Raspberry Pi

Omar H. Khater, Basem Almadani, Farouq Aliyu *Senior Member, IEEE*

***Abstract*—Internet of Things (IoT) based healthcare systems offer significant potential for improving the delivery of healthcare services in humanitarian engineering, providing essential healthcare services to millions of underserved people in remote areas worldwide. However, these areas have poor network infrastructure, making communications difficult for traditional IoT. This paper presents a real-time chest X-ray classification system for hospitals in remote areas using FastDDS real-time middleware, offering reliable real-time communication. We fine-tuned a ResNet50 neural network to an accuracy of 88.61%, a precision of 88.76%, and a recall of 88.49%. Our system results mark an average throughput of 3.2 KB/s and an average latency of 65 ms. The proposed system demonstrates how middleware-based systems can assist doctors in remote locations.**

***Index Terms*—IoT, Fast-DDS, ResNet50, Real-time Systems, DDS, X-Ray, Healthcare.**

I. Introduction

The global population is rising rapidly, but the number of medical staff has not increased proportionally to accommodate this growth. It causes strain on the healthcare system. A few years ago, the COVID-19 pandemic strained the world's healthcare systems. The world at that time realized the importance of increasing the number of medical staff, especially chest doctors. Also, what happened highlights the importance of developing a fully automated system that can help chest doctors in the diagnosis process and accelerate the process.

Recent vision-based deep learning models have proved their efficiency in analyzing medical images, capturing the critical features from X-ray images, and predicting chest diseases such as COVID-19 [1]. Convolutional Neural Networks (CNNs) have proven reliable in the diagnosis process [2]. ResNet50 proved its ability to classify the X-ray images. It can learn complex features and capture patterns due to its skip connections that allow training of deep network layers [3]. Additionally, avoid the vanishing gradient through the residual connections.

Middleware is a crucial component of modern distributed computing and data management, serving as an intermediary layer that connects the operating systems and the applications. It provides a standard interface that smoothly integrates the various system components. We chose FastDDS in this paper because of its scalability and high performance [4]. Additionally, FastDDS offers reliable, low-latency data transfer, which is very important in real-time communication. Moreover, it is open-source, compliant with the DDS standard, and applicable across different platforms [5].

O. H. Khater and B. Almadani are with the Computer Engineering Department. F. Aliyu is with the Center of Excellence in Development of Nonprofit Organizations, KFUPM, Dhahran, Saudi Arabia

Fig. 1 illustrates the proposed system. It integrates the FastDDS publish/subscribe middleware with a vision-based deep-learning model implemented on Raspberry Pi to assist the doctor and speed up the diagnosis process. Our system consists of two nodes with the FastDDS middleware. The first node is the doctor node, which publishes the X-ray image to the middleware and subscribes to the results. The second node is the inference node. It subscribes to the X-ray images and runs the trained ResNet50 model. Then, the result is published on the middleware, and the doctor node subscribes to it. The FastDDS will deliver the X-ray image from the doctor node to the inference node, and then the code script will run automatically. Then, the result will be waited for and passed back to the doctor node.

The rest of the paper is as follows. Section II reviews the approaches regarding middleware technologies, including state-of-the-art vision-based deep learning models diagnosing chest diseases. Section III explains the proposed methodology, while Section IV illustrates the experimental setup. Section V shows the results and discussions, while Section VI discusses the conclusion and future work.

II. Literature Review

In order to handle the growing complexity of real-time communication and data processing across a variety of fields, including distributed systems, robotics, and healthcare, middleware technologies have developed. Depending on their design and intended purpose, these technologies provide different trade-offs in terms of performance, scalability, simplicity, and real-time assurances. We can determine the advantages and disadvantages of each strategy by contrasting different middleware solutions to see how they each handle these issues differently.

Snout, created to make interacting with software-defined radios (SDR) easier, is one of the most straightforward yet effective middleware systems [6]. Snout was created with the intention of simplifying the process of dealing with SDRs by offering a simplified framework that needs little technical knowledge. Because of its architecture, which reduces memory utilization and CPU overhead, it is the perfect choice for real-time applications in settings with limited resources, such as embedded systems or tiny networked devices. However, Snout's simplicity has drawbacks: it lacks scalability in bigger, dispersed contexts where traffic needs may exceed its capacity and concentrate mostly on local management. More reliable middleware solutions, such as DDS (Data Distribution Service), which were created from the bottom up to manage

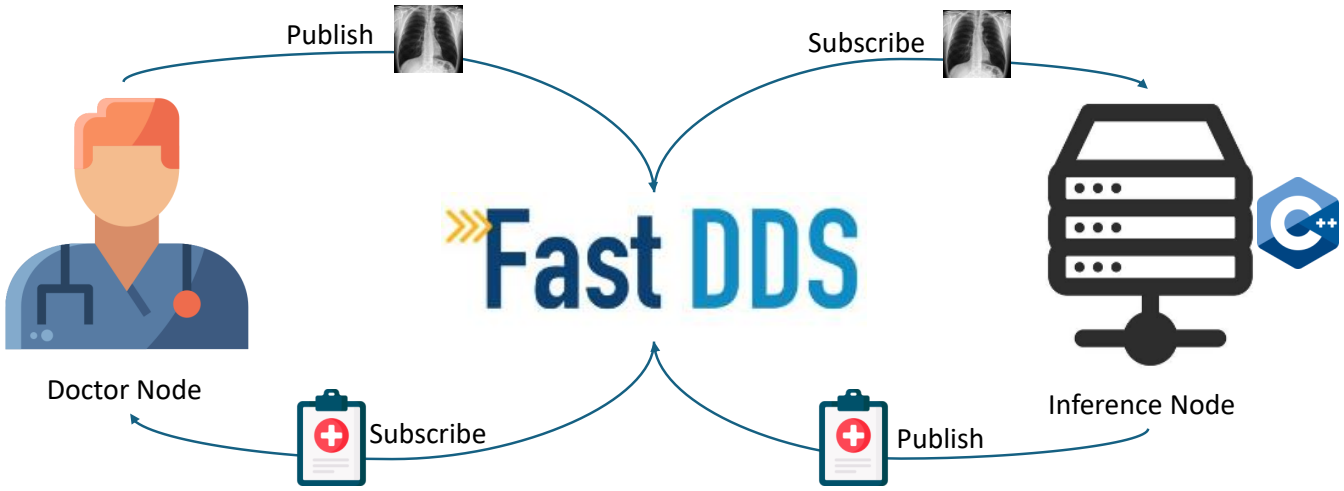

Fig. 1: The proposed system. i) The doctor node publishes the X-ray image. ii) The inference node subscribes to the X-ray image. iii) The inference node publishes the result. iv) The doctor node subscribes to the result.

the challenges of real-time data transmission across dispersed systems, contrast sharply with this lack of scalability.

FastDDS, CycloneDDS, RTI Connext, and OpenDDS are examples of DDS middleware designed for situations where distributed real-time communication is essential. These systems are better suited for large-scale applications like industrial control systems or driverless cars since they are excellent at handling data across several nodes. According to Bode et al., [4], FastDDS and CycloneDDS provide a solid basis for real-time distributed systems by offering reduced latency and packet delay variation in comparison to previous DDS implementations. In contrast to Snout, which is best suited for localized, low-overhead operations, DDS middleware can scale over big networks. However, because they rely on components like the Linux network stack, which is not tuned for real-time performance, even the best-performing DDS middlewares have trouble providing firm real-time assurances. This highlights a significant trade-off: DDS middlewares like FastDDS and CycloneDDS add more complexity but are more appropriate for demanding distributed applications. Snout offers simplicity and efficiency for small-scale use.

Researchers have looked into integrating user-space networking technologies like XDP (Express Data Path) and DPDK (Data Plane Development Kit) to improve the performance of DDS middleware in order to get beyond the real-time performance limits. Bode et al. [7] showed in their study that CycloneDDS, in conjunction with DPDK, greatly lowers latency and boosts throughput, resolving some of the real-time issues with conventional DDS implementations. CycloneDDS and DPDK operate together to reduce mean latency by up to 31%, which makes it an attractive option for high-performance, networked real-time systems like large-scale robotic systems or high-frequency trading. However, DPDK's technical complexity, which includes setup difficulties and driver installations, reduces its applicability in settings that call for simple deployments. This is a noticeable difference from Snout, which provides a reduced barrier to entry and ease of setup despite its scaling constraints, especially for smaller applications that do not need the intensive performance improvements offered by DPDK.

Systems that require connecting real-time and non-real-time components, such as military applications where Combat Management Systems (CMS) must interface with Command and Control (C2) systems, exhibit an additional layer of middleware complexity. Dalkıran et al. [8] examined the JMS-DDS hybrid technique, which combines DDS with Java Message Service (JMS) to provide smooth communication between real-time and non-real-time systems. This combination offers a special benefit: JMS enables the flexibility of asynchronous messaging for jobs that don't require real-time communication, while DDS guarantees real-time communication. This makes it possible to integrate complicated systems without sacrificing real-time guarantees, which is crucial for military applications where coordination and speed are vital. However, there are certain drawbacks to this integration, most notably its dependence on the Java environment, which may limit the solution's versatility in more general contexts. JMS-DDS is more suited for hybrid real-time systems than FastDDS and CycloneDDS, which are more platform-neutral but do not provide the same degree of integration with non-real-time components.

Middleware is essential to robotics because it allows systems to manage many hardware platforms and procedures while operating precisely in real-time. Laurenzi et al. [9] characterize XBot2 as being particularly made for multi-threaded robotic systems. Compared to traditional middleware like ROS (Robot Operating System), which has weak real-time guarantees and has trouble with intra-process communication, it offers real-time performance and modularity, making it more appropriate for real-time robotic applications. In robotics, XBot2 has a distinct advantage over more straightforward options like Snout thanks to its hardware abstraction layer (HAL) and ability to control many devices dynamically and in real-time. While DDS alternatives like FastDDS offer cross-platform compatibility, making them more adaptable for usage

across a variety of distributed real-time applications, XBot2's reliance on Linux-based platforms presents a difficulty for wider applicability. Despite its superiority in high-performance robotics, XBot2's limited mobility outside of Linux contexts restricts its application in dispersed or heterogeneous systems.

Another important role of middleware, especially in wireless communication systems, is to control real-time data transfer over unstable networks. Applications like automated valet parking frequently run in settings where network instability and packet loss are major issues. In order to overcome this, Peeck et al. [10] improved DDS middleware by implementing a retransmission strategy that ranks crucial data according to its temporal significance. This offers a benefit that conventional protocols like TCP and UDP cannot match: it guarantees that time-sensitive data is transmitted consistently, even in lossy network conditions. Although FastDDS and other DDS methods provide dependable network communication, in extremely unstable networks, their performance may still decrease, requiring such improvements. The trade-off between simplicity and the capacity to manage complicated network settings is further highlighted by the fact that simpler systems, such as Snout, are unequipped to meet the demands of large-scale, dispersed communication or the retransmission of crucial data.

Another area where progress is being made is middleware's cloud interoperability. Bharany et al. [11] introduced .NET Core-based middleware, which was created to enhance application portability across cloud platforms such as Amazon Web Services (AWS), Microsoft Azure, and Google Cloud. Applications may more easily move across cloud providers without requiring major rewrites thanks to this middleware, which isolates cloud-specific APIs. However, middleware such as FastDDS lacks the integrated cloud-specific capabilities necessary for smooth interoperability across various cloud environments. Instead, it concentrates on offering scalable, low-latency options for distributed systems. In contrast, Snout does not support cloud-based systems and is mostly concerned with local control. The different scopes of these middleware solutions are reflected in this distinction; DDS middlewares prioritize performance in distributed, high-performance applications, whereas cloud-focused middlewares provide more flexibility in multi-cloud situations. When the operational scope includes different cloud environments, where ease of migration and integration is crucial, the comparison demonstrates how middleware solutions vary.

In the healthcare industry, middleware is extremely essential, particularly for enabling real-time medical diagnostics. Research by Karaddi and Sharma [12], and Alshmrani et al. [13] demonstrated the application of deep learning models for the classification of lung diseases using chest X-ray (CXR) images. Middleware that can handle the quick and dependable flow of medical data between devices and processing systems is necessary for these real-time classification jobs. Real-time decision-making in medical settings is made possible by FastDDS's low-latency communication characteristics, which make it ideal for these kinds of applications. While Snout works well for real-time, small-scale operations, it lacks the scalability and data-handling capabilities needed for big healthcare applications where accuracy and speed are crucial. The significance of real-time middleware in the healthcare industry is shown by middleware such as FastDDS, which guarantees prompt and accurate diagnosis by enabling rapid data flow between medical imaging devices and neural networks. This comparison also demonstrates how middleware such as FastDDS performs very well in dispersed settings where patient outcomes depend on the prompt transfer of large amounts of medical data.

It is evident from an in-depth investigation of prior research on middleware solutions that several platforms, including Snout, DDS (FastDDS, CycloneDDS), and specialized systems like JMS-DDS and XBot2, each have unique benefits and drawbacks. Snout is really good at being simple. However, it isn't scalable enough for big dispersed systems. DDS middleware solutions, on the other hand, struggle with latency optimization and real-time assurances, especially when depending on non-real-time system components like the Linux network stack. Still, they provide the scalability and speed required for high-demand systems. Furthermore, middleware solutions like JMS-DDS facilitate the integration of real-time and non-real-time systems; nevertheless, their wider flexibility is limited by platform requirements, such as their dependence on Java.

Four major restrictions and research gaps so become apparent: i) Scalability problems, especially with smaller, more straightforward systems like Snout; ii) Real-time Guarantees, since even high-performance middleware like DDS finds it difficult to meet stringent real-time requirements; iii) Latency Optimization, which necessitates extra improvements like DPDK to improve performance in networked environments; and iv) the requirement for middleware that effectively supports high-performance deep learning model classifiers, especially in the healthcare industry, where precise and quick data transfer between medical devices and AI models is essential. For middleware technology to advance and better serve high-performance, scalable, and real-time applications, these gaps must be filled.

In this work, we are focusing on addressing the reliability and scalability issues from the middleware concern. Regarding the vision-based deep learning model, we are interested in high precision because if the model incorrectly identifies a diseased patient as healthy, it may lead to serious consequences.

III. Methodology

A. The Scenario

The deployed scenario for the proposed system consists of the doctor node, which will be responsible for publishing the X-ray image to the FastDDS and subscribing to the result. On the other side, the inference node is responsible for subscribing to the X-ray image from FastDDS, running the C++ inference script, then generating the result and publishing it to the FastDDS. Finally, the FastDDS middleware is committed to transmitting the data between the publisher and the subscriber in the proposed system.

B. Proposed system

The proposed system comprises two main components: hardware and software. Firstly, the hardware is Raspberry Pi, which we use to deploy our system. On the other hand, the software components contain a C++ script, which is called from the inference node, which is the subscriber, in that case, to generate the diagnosis result from the received X-ray image from the doctor node, which is the publisher. The Fast DDS middleware is responsible for data transfer between the doctor node and the inference node.

IV. Experimental Setup

The experiment setup for the proposed system contains several hardware configuration details used in the FastDDS setup on Raspberry Pi with the Linux operating system, and the RAM size is 4 GB. The dataset [14] [15] was used to fine-tune the ResNet50 for chest disease classification problems. The hardware utilized compromises NVIDIA GeForce RTX 3080 Ti with 16 GB memory, and the RAM size is 32 GB. The dataset used involves 21,269 X-ray images with four classes.

The chosen dataset in this work contains X-ray images of cases of COVID-19, lung opacity, viral Pneumonia, and pneumonia chest. It was acquired from Kaggle, an open-source online resource. The datasets consist of a total of 21,269 labeled X-ray images, of which 3,617 are COVID-19, 10,193 are normal chest, 6,013 are lung opacity, and 1,346 are Viral Pneumonia. The selected dataset was split into 80% for training, 10% for validation, and 10% for testing.

V. Results and Discussions

After the fine-tuning of the ResNet50 on the X-ray images dataset, the ResNet50 showed superior performance, achieving an accuracy of 88.61%, a precision of 88.76%, and a recall of 88.49%, which is expected to be very beneficial for the medical staff and accelerate the diagnosis process. The ResNet50 performance is illustrated in Fig. 2[1]

The total latency was determined based on precise calculations. We analyzed the transmission latency starting from publishing the X-ray image from the doctor's node through the FastDDS reaching the inference node, also the inference time, and ending with the inference node publishing the result through the FastDDS and reaching back the doctor's node, with an average latency of 65 ms. The whole transmission data was around 185 packets. Fig. 4 shows the visual representation of the transmitted packets' latency. The latency pattern provides an extremely valuable insight into the operational characteristics of the communication between the doctor node and the inference node. The throughput computations assess the publishing and subscribing processes, which is shown in Fig. 3 is around 3,200 bytes/second.

Based on our results, our system proved the effectiveness of ResNet50 in X-ray image classification and FastDDS, which offers real-time data transmission with very high reliability and scalability.

[1]The code used in this study can be accessed through this link: omarkhater98.github.io/FastDDS_chest_classification

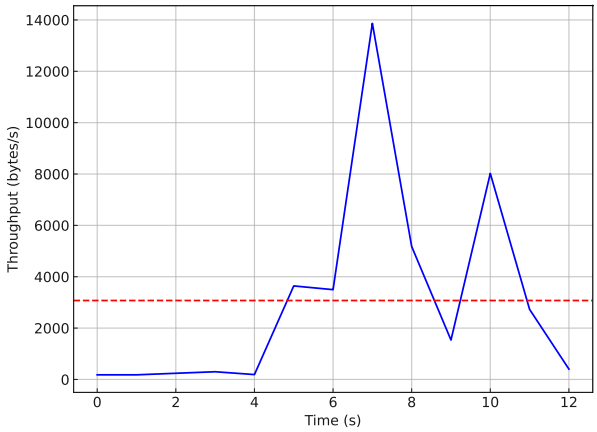

Fig. 3: Throughput profile

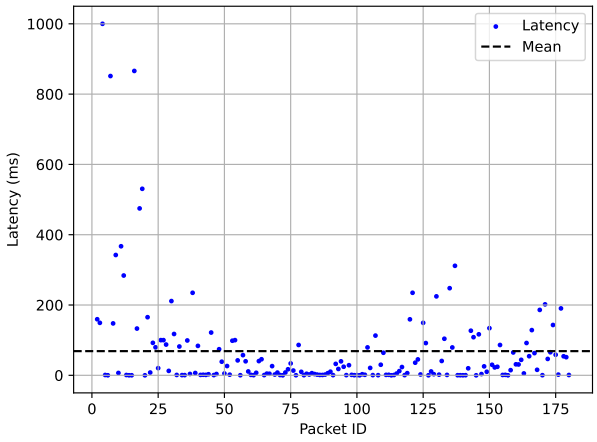

Fig. 4: Latency profile

VI. Conclusion and Future Work

This work investigated the integration between the vision-based deep learning models and middleware technology to generate a system capable of assisting the medical staff in chest disease classification. The ResNet50 showed superior performance with the X-ray images because it contained residual connections to prevent vanishing gradient, and its efficiency in feature extraction made it the suitable choice for the classification task. Additionally, the FastDDS middleware is presenting outstanding performance in terms of reliability and scalability concerns. The ResNet50 model achieved an accuracy of 88.61%, a precision of 88.76%, and a recall of 88.49%. Moreover, the results mark an average throughput of 3,200 bytes/second and an average latency of 65 ms.

In the future, the research should focus on enhancing scalability and handling different sizes of images while maintaining real-time performance. Additionally, with scalability, the system must be secured against attacks, so we suggest adding encryption techniques and authentication in the data transfer process. Also, extend and add more nodes to offer

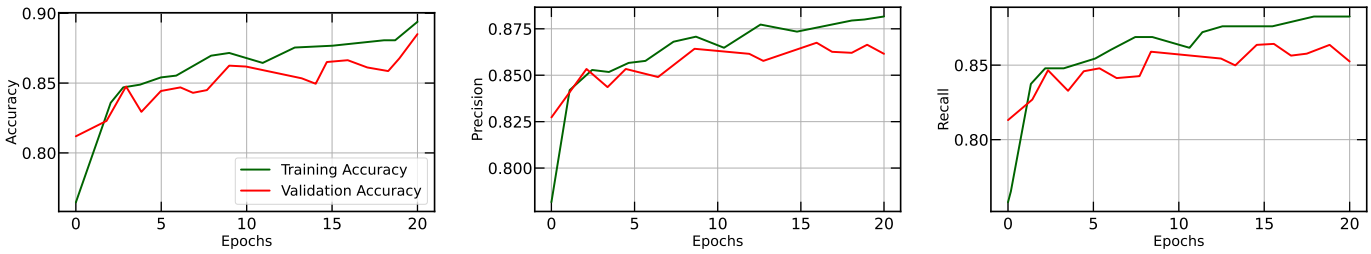

Fig. 2: Performance of ResNet50

fault tolerance, allow load-balancing, and make sure that the system will perform efficiently under different scenarios.

ACKNOWLEDGE

The authors would like to acknowledge all support provided by King Fahd University of Petroleum & Minerals (KFUPM).

REFERENCES

[1] R. M. Pereira, D. Bertolini, L. O. Teixeira, C. N. Silla Jr, and Y. M. Costa, "Covid-19 identification in chest x-ray images on flat and hierarchical classification scenarios," *Computer methods and programs in biomedicine*, vol. 194, p. 105532, 2020.

[2] K. Santosh, D. Das, and U. Pal, "Truncated inception net: Covid-19 outbreak screening using chest x-rays," *PREPRINT (Version 1) available at Research Square*, vol. 3, 2020.

[3] B. Koonce and B. Koonce, "Resnet 50," *Convolutional neural networks with swift for tensorflow: image recognition and dataset categorization*, pp. 63–72, 2021.

[4] V. Bode, C. Trinitis, M. Schulz, D. Buettner, and T. Preclik, "Dds implementations as real-time middleware – a systematic evaluation," in *2023 IEEE 29th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, pp. 186–195, 2023.

[5] J. Zhang, M. Ma, P. Wang, and X.-d. Sun, "Middleware for the internet of things: A survey on requirements, enabling technologies, and solutions," *Journal of Systems Architecture*, vol. 117, p. 102098, 2021.

[6] J. K. Becker and D. Starobinski, "Snout: A middleware platform for software-defined radios," *IEEE Transactions on Network and Service Management*, vol. 20, no. 1, pp. 644–657, 2022.

[7] V. Bode, C. Trinitis, M. Schulz, D. Buettner, and T. Preclik, "Adopting user-space networking for dds message-oriented middleware," in *2024 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pp. 36–46, IEEE, 2024.

[8] E. Dalkıran, T. Önel, O. Topcu, and K. A. Demir, "Automated integration of real-time and non-real-time defense systems," *Defence Technology*, vol. 17, no. 2, pp. 657–670, 2021.

[9] A. Laurenzi, D. Antonucci, N. G. Tsagarakis, and L. Muratore, "The xbot2 real-time middleware for robotics," *Robotics and Autonomous Systems*, vol. 163, p. 104379, 2023.

[10] J. Peeck, M. Möstl, T. Ishigooka, and R. Ernst, "A middleware protocol for time-critical wireless communication of large data samples," in *2021 IEEE Real-Time Systems Symposium (RTSS)*, pp. 1–13, IEEE, 2021.

[11] S. Bharany, K. Kaur, S. Badotra, S. Rani, Kavita, M. Wozniak, J. Shafi, and M. F. Ijaz, "Efficient middleware for the portability of paas services consuming applications among heterogeneous clouds," *Sensors*, vol. 22, no. 13, p. 5013, 2022.

[12] S. H. Karaddi and L. D. Sharma, "Automated multi-class classification of lung diseases from cxr-images using pre-trained convolutional neural networks," *Expert Systems with Applications*, vol. 211, p. 118650, 2023.

[13] G. M. M. Alshmrani, Q. Ni, R. Jiang, H. Pervaiz, and N. M. Elshennawy, "A deep learning architecture for multi-class lung diseases classification using chest x-ray (cxr) images," *Alexandria Engineering Journal*, vol. 64, pp. 923–935, 2023.

[14] T. Rahman, A. Khandakar, Y. Qiblawey, A. Tahir, S. Kiranyaz, S. B. A. Kashem, M. T. Islam, S. Al Maadeed, S. M. Zughaier, M. S. Khan, *et al.*, "Exploring the effect of image enhancement techniques on covid-19 detection using chest x-ray images," *Computers in biology and medicine*, vol. 132, p. 104319, 2021.

[15] M. E. Chowdhury, T. Rahman, A. Khandakar, R. Mazhar, M. A. Kadir, Z. B. Mahbub, K. R. Islam, M. S. Khan, A. Iqbal, N. Al Emadi, *et al.*, "Can ai help in screening viral and covid-19 pneumonia?," *Ieee Access*, vol. 8, pp. 132665–132676, 2020.