2023 IEEE/ACM 23rd International Symposium on Cluster, Cloud and Internet Computing Workshops (CCGridW) | 979-8-3503-0208-0/23/$31.00 ©2023 IEEE | DOI: 10.1109/CCGridW59191.2023.00041

Disease Prediction using Chest X-ray Images in Serverless Data pipeline Framework

Vikas Singh
Computer Science
IIIT Lucknow
Lucknow, India, 226002
Email id: mcs21018@iiitl.ac.in

Neha Singh
Computer Science
IIIT Lucknow
Lucknow, India, 226002
Email id: neha08hs@gmail.com

Mainak Adhikari
Computer Science
IIIT Lucknow
Lucknow, India, 226002
Email id: mainak@iiitl.ac.in

***Abstract*—Serverless architecture is a rapidly emerging trend in the field of cloud computing that promises increased flexibility, scalability, and cost-effectiveness compared to traditional server-based approaches. Leveraging machines to automatically analyze and predict the disease using image data such as chest X-ray images is becoming a challenging task for various contemporary applications. Serverless computing is a cloud computing execution model that provides and manages resources based on the requirements of the users/applications. Besides that, modern data-intensive applications require the power to manage the flow of data between different components in a serverless platform. Motivated by that, in this paper, we develop a new serverless data pipeline framework for predicting disease using chest X-ray images. The system utilizes Deep Learning (DL)-based image classification models hosted on Google serverless platform for COVID-19 diagnosis. For disease prediction, we incorporate a transfer learning technique over three popular DL models, namely VGG-16, DenseNet121, and ResNet50. The experimental analysis demonstrates that the proposed serverless data pipeline framework achieves high accuracy, reliability, and speed during COVID-19 disease diagnosis. As per the simulation results, the VGG-16 model outperforms the existing DL models and achieves 97.66% accuracy.**

***Index Terms*—Serverless Computing, Data Pipeline, Cloud Functions, Cloud Storage Bucket, Function-as-a-Service.**

I. Introduction

Diagnosis of diseases in early stages can help to prevent them from progressing to more advanced stages. In many cases, early treatment can be less invasive and less costly, which can help improve patient outcomes and reduce health-care costs. Traditional diagnostics methods are often limited by subjectivity and variability as different clinicians interpret the same results differently. Traditional approaches are also limited by their capacity to handle a large amount of data, detect shallow and complex patterns at times requiring extensive manual analysis and rely on the availability of experts who can interpret those diagnostic reports.

Artificial Intelligence (AI) and Machine Learning (ML) based approaches for disease diagnosis can provide many consistent and objective results while reducing human effort. AI techniques can be cost-effective, requiring less expensive equipment and less specialized expertise than traditional diagnostic methods. However, utilizing the existing AI-based diagnosis methods is challenging due to high training and running time. Besides, most of the existing AI techniques require a certain level of technical expertise, which medical professionals lack and often need to outsource it. Moreover, the need to manage the computing resources such as hardware and software can add to the already existing overheads during model training.

The traditional approach of image classification requires huge resource utilization and human intervention. As a solution, cloud computing is used for delivering efficient computing resources such as processing devices, storage, databases, networking, software, and analytics tools, over the Internet. In other words, instead of owning and maintaining physical servers and other infrastructure, individuals, and organizations can access these resources remotely through a third-party service provider like Amazon, Google, IBM, etc. There are several different types of cloud computing architecture like Infrastructure-as-a-Service, Platform-as-a-Service, Software-as-a-Service, and Function-as-a-Service (FaaS). In recent times, FaaS has been gaining immense popularity as a cloud computing model. FaaS allows developers to write and deploy small pieces of independent code (known as functions) that can be executed on demand without the need of managing and scaling any underlying infrastructure. The cloud provider manages the servers, networking, and other resources required to execute the code, and charges the users based on the actual usage of the function rather than a fixed fee for the entire infrastructure. FaaS is best suited for building event-driven applications and serverless architectures, where functions are triggered by specific events such as user requests or changes in data. All the cloud providers have their version of FaaS computing platforms like Amazon Web Services (AWS) Lambda, Azure Functions, Google Cloud Function (GCF), etc., referred to as cloud functions. The application platform may be different but the underlying concept and structure remain the same for different service providers.

Many individuals tend to think of serverless computing and FaaS as being the same, though they are related concepts but not the same thing. Serverless computing is a broader concept that encompasses a range of cloud computing services that allow developers to build and deploy applications without having to manage any underlying infrastructure. This can include FaaS as well as other serverless services such as databases, storage, and messaging. In other words, FaaS is a specific type of serverless computing that focuses on the execution of small code functions while serverless computing encompasses a wider range of services and functionality. As stated earlier, FaaS functions are independent chunks of code executed individually, so it is important to resolve the dependencies of each function, which makes the concept of containerization important from the perspective of FaaS and Serverless. Containerization refers to the process of packaging an application or a function with all its dependencies and resources into a container allowing developers to create and deploy functions as self-contained and isolated units. Using containerization, developers create complex workflows or applications that involve multiple functions and each function can be packaged into a separate container and deployed independently.

Further, the incorporation of a serverless data pipeline can greatly enhance the automation of disease detection and diagnosis processes by streamlining the movement of data through various stages of the prediction process. The data pipeline streamlines the movement of data through various stages of the prediction process such as data collection, pre-processing, and modeling. The serverless nature of the pipeline allows for efficient scaling and automatic allocation of resources as needed without requiring manual intervention. Serverless computing is cost-effective and highly scalable when compared to traditional centralized computing systems. Deep Learning (DL)-based serverless FaaS pipeline also allows for faster development speed and higher availability of the system as opposed to traditional distributed ML/DL approaches while being easier to manage and highly cost-effective.

A. *Motivation*

The emergence of new and potentially fatal diseases that can rapidly spread across borders poses a significant threat to public health. Consequently, there is a pressing need for innovative and efficient methods of illness diagnosis that can aid in mitigating the risks associated with these diseases. AI has been identified as a potential solution to this problem due to its ability to enhance the accuracy and effectiveness of medical diagnostics. Utilizing techniques such as image processing and genetics-based diagnostics, AI has the potential to provide better care for patients and reduce the workload of healthcare professionals. However, implementing AI in healthcare requires significant computational power and resources, which can result in high development and maintenance costs. To overcome these challenges, we incorporate serverless computing that allows the on-demand allocation of computational resources and easy accessibility. By leveraging serverless computing, medical diagnostics can be performed more cost-effectively without sacrificing performance or accuracy. This makes the implementation of AI in healthcare more accessible, particularly in resource-limited settings where traditional computing infrastructures may not be available. Despite the potential benefits of AI and serverless computing in medical diagnostics, there are challenges to be addressed. Another important challenge in the serverless computing platform is managing the flow of data between different components by incorporating data pipeline technology. Nevertheless, the use of AI and serverless computing with data pipeline technology in medical diagnostics remains a promising area of research with significant potential for improving public health outcomes.

B. *Novelty and Contributions*

Motivated by the above-mentioned challenges, in this paper, we develop a new serverless data pipeline framework for predicting COVID-19 disease [1] using chest X-ray images. The contributions of the paper are summarized as follows.

- We develop a serverless data pipeline framework for disease prediction to automate the movement of data in cloud infrastructure. The proposed serverless data pipeline is created on the Google Cloud Platform (GCP), where a function is created using a standard DL model and is triggered by a cloud bucket whenever a chest X-ray image is uploaded for diagnosis. The proposed framework automates the process of fetching, pre-processing, and storing data while eliminating manual errors and reducing the overall deployment time of the infrastructure.
- Incorporate various standard DL models such as VGG-16, DenseNet121, and ResNet50 in GCP as a function for disease prediction. Initially, the model is trained using the publicly available dataset related to chest X-ray images in the cloud server. Once the model is trained, a transfer learning mechanism is applied over the trained DL model to prepare a lightweight model and deploy it on GCF during disease prediction.

Extensive simulation results demonstrate the efficiency of the proposed serverless data pipeline framework and VGG-16 model over the existing ones using various performance metrics. The remaining sections of this research paper are organized as follows. Section II describes the state-of-the-art techniques in the fields of serverless computing and DL. Section III outlines the system model followed by the problem statement. Section IV describes the detailed methodology of the proposed serverless data pipeline framework. The experimental results of the proposed methodology are demonstrated in Section V. Finally, Section VI concludes the proposed methodology.

II. Literature Review

Cloud computing is a revolutionary advancement in the field of information technology, and it has become a primary business model for delivering computational resources. With the advent of cloud computing, both individuals and organizations can enjoy the benefits of on-demand access to a shared pool of managed and scalable resources including processing devices, applications, and storage [2]. Users/Customers depend heavily on cloud services in their daily lives for storing data, writing documents, managing businesses, and engaging in online activities such as gaming. The concept of the computing reference model and the development of cloud services have progressed to new levels. Serverless computing is a cutting-edge execution model, where the cloud service provider dynamically manages the allocation of compute resources without the intervention of the users. Instead of paying for pre-purchased units of computing capacity, the consumer is billed for the actual volume of resources consumed during data processing or analysis. This innovative model provides optimal cost-efficiency, reduces configuration overheads, and enhances the application's scalability in the cloud [3]. Cloud service providers like Amazon Web Services(AWS), Azure [4], and Google Cloud Platform (GCP) [5] have embraced the potential of the serverless compute model, evident by their adoption of the serverless computing paradigm. Some open-source serverless frameworks also exist like OpenFaaS [6], Knative [7], etc. FaaS is a serverless paradigm that allows developers to build and deploy small, independent functions that are triggered by an event [8]. These functions are designed to perform a specific task bound to an event and can be easily integrated into larger applications. Whenever an event occurs, it triggers the bound function through an Application Programming Interface (API) call, and the cloud platform executes the corresponding function parallelly. This computing model also has some drawbacks, which are described as follows.

- The function must be stateless, i.e., any intermediate data generated during execution must be stored in external cloud storage, provided by the platform.
- The functions are allocated limited resources such as CPU power, local storage, and run-time. For example, GCF [9] offers a maximum of 8 GB of local memory and 500 MB of function size, and each function has a maximum run-time of 540 seconds and a maximum event size of 10 MB.
- Most cloud providers limit the integration of functions to the other computing services available on their own platform.
- Serverless functions only support a limited set of popular programming languages and development tools. The debugging capabilities of serverless code are fairly limited.

Despite these limitations, the merits of serverless computing such as cost-effectiveness, scalability, and ease of deployment, outweigh its drawbacks [10]. On the other hand, containerization provides a lightweight, portable, and consistent environment to package and deploy serverless functions [11]. Additionally, they provide isolation, which increases the security of the application by preventing one function from accessing the data or resources of another function. Serverless computing has emerged as a promising platform for deploying ML models in the cloud [12]. The data pipeline is an important aspect of serverless applications [13]. Data pipeline tools allow for the automated movement of data between different stages or systems [14]. This can be particularly useful in serverless architectures, where the focus is on building and deploying small, independent functions that are triggered by events [15].

The medical industry is continuously adopting new technologies to improve disease detection and treatment methods. With the increasing availability of medical imaging data, there is a growing demand for automated and efficient methods for disease detection using image processing and DL algorithms [16]. In recent years, there has been a shift towards using AI and DL algorithms for the diagnosis of various diseases. One such application is the detection of COVID-19 disease using chest X-ray images [17], [18]. Lately, there have been several studies exploring the use of serverless architecture for deploying DL models in the medical industry [19].

TABLE I
COMPARATIVE INVESTIGATION OF THE STUDY GAP IN LITERATURE

Existing Work	Deep Neural Networks	High Scalability	Efficient Resource Utilization	Data Pipeline
[18]	✓	χ	χ	χ
[17]	✓	χ	χ	χ
[14]	χ	✓	✓	✓
[23]	χ	✓	✓	χ
[11]	χ	✓	χ	χ
[24]	✓	χ	χ	χ
[15]	χ	✓	✓	χ
Our Work	✓	✓	✓	✓

In summary, the literature indicates that the use of AI and ML models in disease diagnosis is a growing field with many successful applications in the medical industry. Furthermore, the use of serverless architecture with data pipelining provides a promising platform for the deployment of end-to-end ML model workflow with benefits such as reduced costs [20], simplified deployment, and automation [21], [22]. The comparative analysis of the proposed serverless data pipeline framework and existing works for disease prediction over four key attributes is depicted in Table I.

III. SYSTEM MODEL AND PROBLEM STATEMENT

This section outlines the components and their interactions in the system model and provides the problem statement that the proposed solution aims to address.

A. System Model

In this work, we develop a serverless data pipeline framework for running DL-based image classification models that leverages cloud-based services to automate the process of disease prediction using chest X-ray images and improves the reliability and scalability of the application. Fig. 1 illustrates the proposed serverless data pipeline framework. The system relies on cloud-based storage services for storing the images and the trained DL model for disease prediction using function. The images are uploaded to a designated cloud bucket while the model is stored in another bucket.

The system triggers the GCF through Pub/Sub whenever the designated trigger event occurs, which fetches the image and pre-processes it. The pre-processing step involves resizing the image to a specific size and normalizing the pixel values. Next, the processed image is passed to the data pipeline that manages the flow of the images and analyzed the images using a DL model. During model training, we used VGG-19 for training the model using publicly available datasets in the cloud server [25]. Initially, the trained model is stored in a

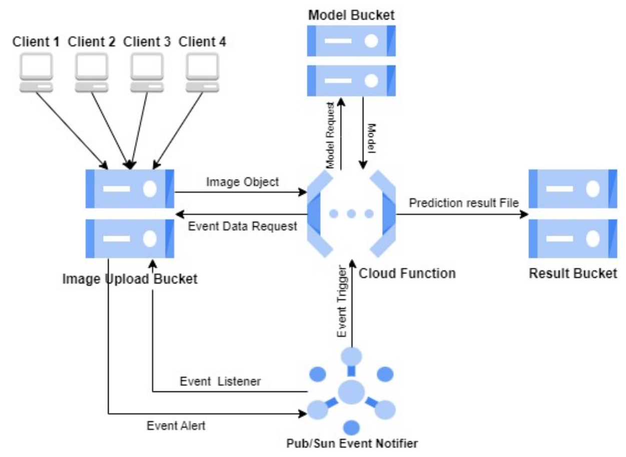

Fig. 1. Serverless Framework for Disease Prediction

separate cloud bucket, which is further loaded by the GCF during run-time for disease prediction.

The system is designed to be fully serverless, which means that it does not require any servers to be provisioned or managed by the user. This causes lower infrastructure costs and faster development and deployment times, and the system can be ready more quickly for data analysis. The system leverages cloud-based services such as GCF, Pub/Sub, and cloud storage to handle the various components of the data pipeline. The pipelining enables the system to be fault-tolerant and scalable.

B. Problem Description

The main challenge in traditional server-based architectures is the need for manual server provisioning and management. This can be a time-consuming and costly mechanism as it requires dedicated computational resources to handle server maintenance and upgradation. Additionally, server-based architectures may struggle with scalability issues as they may not handle sudden spikes in traffic without manual intervention. Serverless architectures address these challenges by providing a fully managed environment, where users only pay for the computing resources they use.

In many data-driven applications, data needs to be processed and transformed as it moves through the system. This can involve complex workflows and data transformations that are difficult to manage manually. Traditional data processing architectures may require custom scripts or manual interventions to handle these workflows, which can be time-consuming and error-prone. Data pipeline technologies address these challenges by providing a structured and automated way to move data through the system. Data pipelines can handle complex data transformations and workflows automatically, allowing users to focus on higher-level application logic. Additionally, data pipelines can be designed to be fault-tolerant and scalable. To address the above-mentioned challenges, in this work we have developed a new serverless data pipeline framework in GCP for predicting the COVID-19 disease using the standard DL models. A detailed description of the proposed methodology is elaborated in the following section.

IV. Deep Learning-enabled Serverless Data Pipeline Mechanism

In this section, we meticulously examine the problem statement by presenting a comprehensive review of the in-hand research challenges followed by the steps taken to resolve those challenges. We have provided a thorough system framework that has been created to address the reported issue. Various factors such as scalability, performance, and cost-effectiveness are taken into consideration for developing the proposed methodology to design an optimal solution to solve the problems efficiently and effectively. The proposed DL-enabled serverless data pipeline methodology is applied in disease prediction with chest X-ray images, which is a difficult challenge owing to the complexity of the disease and the limited image dataset available for DL model training.

A. Diesease Prediction Model

In the first phase of the proposed methodology, we built an image classification model to predict the disease using chest X-ray images. Two publicly available datasets of COVID-19 chest X-ray images are merged to develop a new dataset. The new dataset suffers from population invariance as the number of images of non-infected people is more in comparison to people who were infected by COVID-19. To handle this issue, we have applied data synthesis to select random images from the COVID-19 pool and applied data augmentation to create new images. Further, various pre-processing techniques such as the resolution of images are used to modify images to 224×224 pixels and then normalization is incorporated to manage image consistency.

Once the dataset is ready, we proceed to train the proposed DL-based image classification model for disease prediction. Due to the limited size of the newly created dataset, training an image classification model from scratch could not achieve a desirable result. As a solution, we used a transfer learning technique, which is an ML technique where a model trained on one task is adapted for another related but not the same task, often with much less data with minimum training time. To incorporate transfer learning, we used three pre-trained DL models, namely VGG-16, DenseNet121, and ResNet50, which are already trained on ImageNet dataset on the cloud server.

Once the weights and biases are downloaded from the trained DL models, we proceed to fine-tune them by setting the learning rate to 0.001 using categorical cross-entropy as a loss function and Adam as an optimizer. Here, the learning rate reduction factor is set to 0.5 with patience 3 and for the output layer, we have used softmax as the activation function. Further, we used our previously created dataset to train the image classification models in the serverless platform for disease prediction. The workflow of model training for disease prediction is depicted in Fig 2. Next. to deploy this model on a cloud server and to automate the whole function, we create a serverless data pipeline, described in the following sub-section.

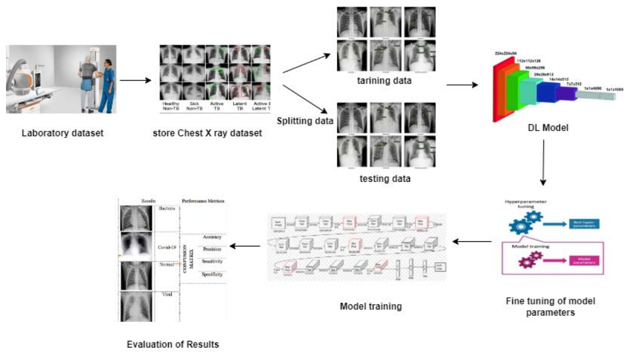

Fig. 2. Training and Evaluation of DL Image Classification Model

B. Serverless Data Pipeline Framework

In the second phase of the proposed work, we have created a serverless data Pipeline framework, illustrated in Fig. 3 that obtains the best performance on limited resources by providing easy scalability and reliability. The proposed serverless data pipeline framework implemented on GCP involves the creation of a cloud function along with a container, which is triggered by a cloud bucket whenever a chest X-ray image is uploaded to it. Whenever an object is created, an event is occurred, which is then published to a Google Pub/Sub topic. The subscribed cloud function receives the event and reads the image from the cloud bucket. To access the cloud function for disease prediction, proper permission policies needed to be attached to the function using Identity and Access Management (IAM) service. IAM allows users to manage access control and permissions for their GCP resources. IAM also helps to implement security policies by providing fine-grained access control over resources. After pre-processing the image according to the trained DL model, the cloud function fetches the trained DL model stored in a separate cloud bucket. Then, the processed image is passed to the DL model for disease detection. The obtained results are saved to a text file with the same name as the image and saved to a separate cloud bucket as a form of result. The use of a data pipeline is crucial to design the proposed system model for efficient functioning. It enables the automatic transfer of data between different stages of the system. The proposed data pipeline model is responsible for transferring data between the cloud bucket and the cloud function for processing and analysis. By using the data pipeline, we can streamline the flow of data between different stages of the serverless platform and ensure smooth functioning. This ensures that the proposed serverless data pipeline mechanism performs efficiently and effectively and can handle sudden spikes in traffic without any downtime or performance issues.

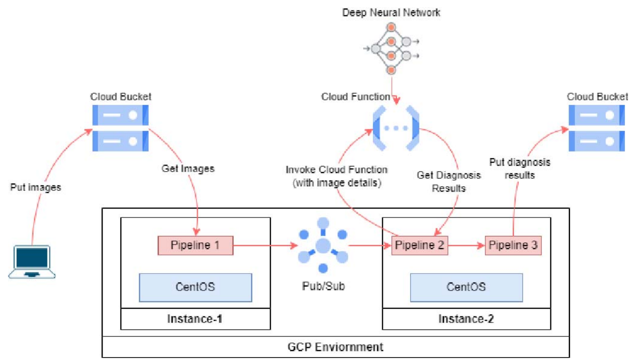

Fig. 3. Serverless Data Pipeline Framework

V. Experimental Analysis

In this section, we conduct an extensive set of simulations and demonstrate the efficiency of the proposed serverless data pipeline framework with the standard DL model over the benchmark COVID-19 dataset. It is important to thoroughly analyze and test the DL models and serverless framework to ensure that it meets the required specifications and performs as expected. To this end, we have conducted a number of tests on both the trained DL model and the serverless framework. This includes testing the model's accuracy, precision, recall, and F1-score on multiple benchmark datasets. We have also measured the model's inference time and the time taken for the serverless function to analyze the data. Furthermore, we have analyzed the model's performance on different subsets of the dataset such as testing the model's performance on images from patients of different age groups or with different levels of disease severity. This helps to identify any potential areas of improvement and make necessary adjustments to the model. Overall, the results of these tests provide valuable insights into the performance of the proposed DL-enabled serverless data pipeline methodology and guide future developments and improvements.

A. Dataset

To show the working of the proposed methodology, we have trained the model on the publicly available images of the chest X-rays of COVID-19. Due to limited quality images availability in a single dataset, we have merged two different publicly available datasets [1] and [2]. The newly created dataset suffered from class imbalance as the number of NORMAL case images is more than the COVID-19 case images. Further, data augmentation was used to handle this class imbalance, which resulted in the creation of the final dataset, consisted of 2746 total images that were divided into two classes: COVID-19 and NORMAL. To evaluate the proposed serverless data pipeline framework, we have created multiple test plans of varying sizes by randomly selecting images from the dataset.

[1]https://github.com/ieee8023/covid-chestxray-dataset

[2]https://www.kaggle.com/datasets/tawsifurrahman/covid19-radiography-database

B. Simulation Setup

We start by reshaping the images to the fixed dimensions of 224×224 and normalizing them to reduce inconsistency. The image classification models are trained on Google Colab where the used configurations are 12.7GB of RAM, Nvidia K80 GPU with 12GB of RAM, and 78GB of disk space. To evaluate the proposed serverless disease prediction system, we run a simulation on GCP with a test plan generated from publicly available datasets of chest X-ray images. Moreover, the resources utilized by the cloud function instances Intel Xeon CPU, and 4GB of function memory. Further, the minimum and maximum instance count for the GCF was set to 0 and 100 respectively. Then timeout for the cloud function is set to 180sec. The performance metrics of the conducted tests were obtained through Google Cloud Metrics directly.

C. Performance Analysis

In this section, we evaluate the performance of the proposed methodology for disease prediction using the DL model, deployed on a serverless framework. We first present the results of testing the trained models on a test dataset, followed by an evaluation of the performance of the serverless platform in terms of its latency and scalability. The performance analysis provides insights into the effectiveness and efficiency of the proposed serverless data pipeline framework for disease prediction, highlighting its potential as a practical tool for healthcare professionals. The performance of the trained DL models is analyzed in terms of accuracy, precision, recall, and F1-score, defined in [26].

1) Prediction Accuracy and Loss : We evaluated the performance of our trained DL models for image classification based on different performance metrics. Table II presents a comparison of the accuracy, precision, recall, and F-score metrics for the standard DL models. The VGG-16 model performs better due to the high number of trainable parameters when compared to other DL models as it was trained on a large dataset (ImageNet). Thus, VGG-16 has better pre-trained weights, which enables it to extract more meaningful features.

TABLE II
COMPARATIVE ANALYSIS OF DIFFERENT DL MODELS

Method	Train	Test	Accuracy	Precision	Recall	F-score
CNN-2d	2196	550	80.91	0.69	0.72	0.70
	2334	412	76.02	0.65	0.71	0.68
ResNet-50	2196	550	87.23	0.78	0.71	0.74
	2334	412	84.71	0.77	0.68	0.72
VGG-16	**2196**	**550**	**97.66**	**0.85**	**0.80**	**0.82**
	2334	412	75.47	0.81	0.72	0.76
DenseNet-121	2196	550	92.86	0.82	0.76	0.79
	2334	412	89.41	0.78	0.75	0.76

The pictorial presentation of the accuracy and loss curves for all the trained DL image classification models are shown in Fig. 4 and Fig. 5, respectively, which accuracy and loss depicted on the y-axis and number of epochs on the x-axis. From the figures, we observe how the various models performed throughout the training and validation process on the COVID-19 X-ray image dataset. The number of epochs was set to 30 while training the DL models to maintain uniformity across them and to avoid overfitting due to the limited size of the dataset before applying the transfer learning technique over the standard DL model. As per the figure and Table II, the VGG-16 model outperforms the existing DL model followed by DenseNet-121.

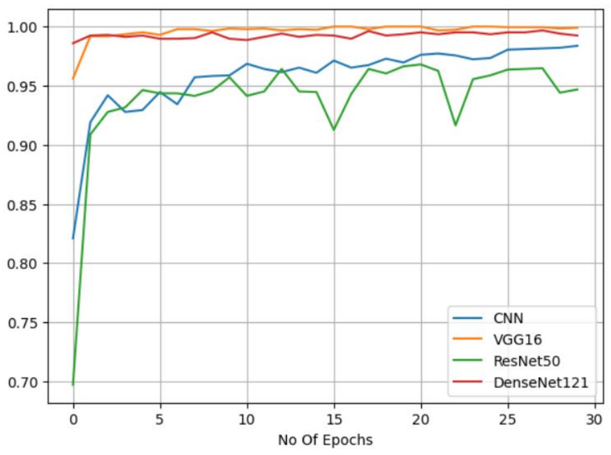

Fig. 4. Comparison of Accuracy for different DL models

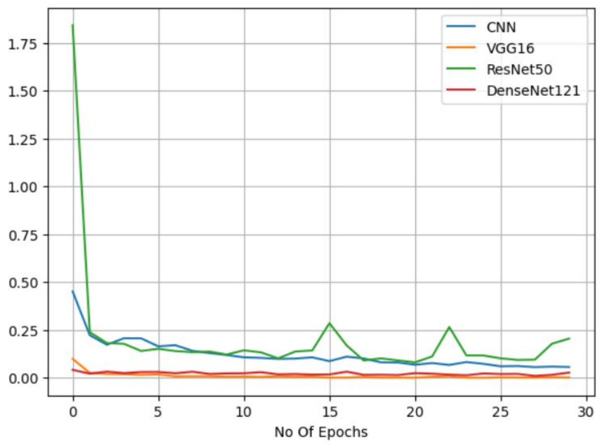

Fig. 5. Comparison of Loss for different DL models

On analyzing Fig. 4 and Fig. 5, we observe that almost every trained model is exhibiting an outlier behavior at 2 instances near about epoch 15 and 22, however, the VGG-16 model shows minimal outlier behaviour as visible in Fig. 6 and Fig. 7. The figures show the accuracy and loss curves for the VGG-16 image classification model. This outlier behavior may be caused by neutral deviation in the population or the limited size of the dataset.

2) CPU Utilization and Memory Usage: To evaluate the performance of the serverless data pipeline framework for disease prediction, we conducted several experiments using multiple test datasets of variable sizes. We have used the cloud metrics provided by GCP to monitor the results. We

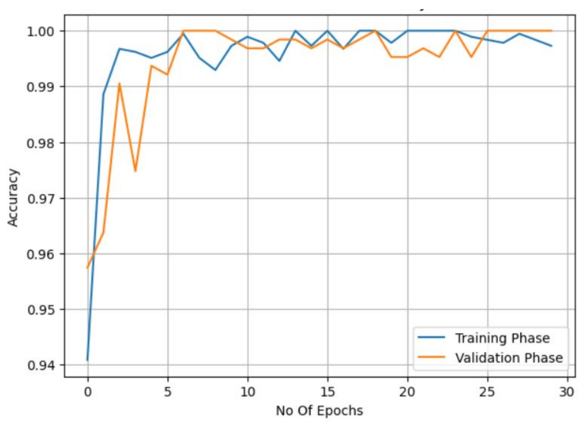

Fig. 6. Training & Validation Accuracy for VGG-16 model

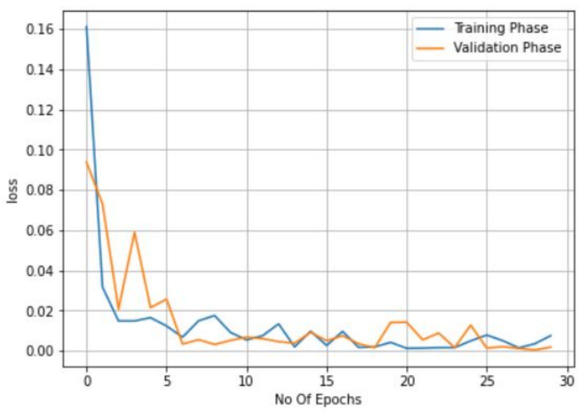

Fig. 7. Training & Validation Loss for VGG-16 model

evaluated the application's performance using metrics like CPU utilization and memory usage.

Fig. 8 shows the memory usage pattern when 100 concurrent requests are passed to the serverless function. The graph represents three regions, where the first bottommost region represents the memory range that half of the requests fall within, and the second middle region shows the limit under which 75% of the recommendations fall in. The top region represents the maximum memory utilization. Similarly, Fig. 9 shows the distribution of execution time per call. The bottom region of the figure represents the fastest 50% calls to function that is treated as the best-case scenario, the middle region represents the slowest 50% calls that are the median value and the topmost line represents the slowest calls or the worst-case scenario of serverless function execution.

To know how the proposed serverless function memory usage changes with the number of concurrent requests, we have made multiple invocations to the proposed function with distinct numbers of concurrent requests being 1, 3, 7, and 15. Fig. 10 shows the increase in memory utilization, which is not linear when increasing the number of concurrent requests. The memory needed for a single request is about 800MB while memory usage is about 1900MB and 2100MB for 7 and 15 concurrent requests, respectively.

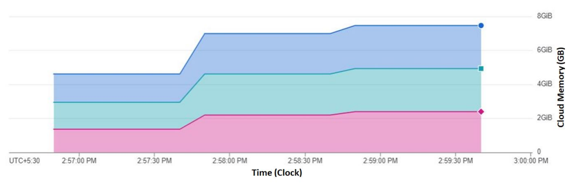

Fig. 8. Memory usage Statistics of Cloud Function

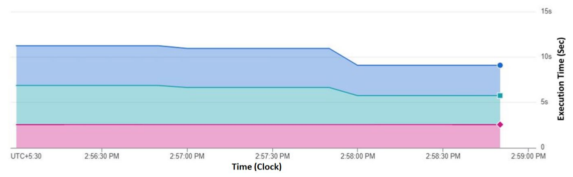

Fig. 9. Distribution of Execution Time per Call

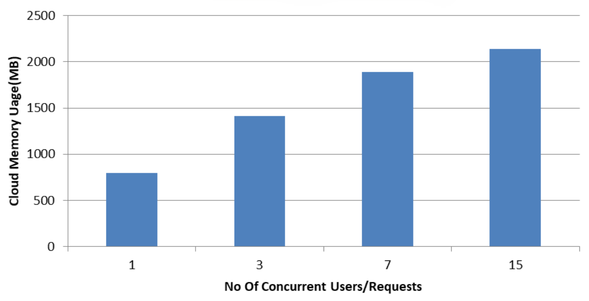

Fig. 10. Cloud Memory Usage for Concurrent requests/users

In order to evaluate the load capacity and response time of the proposed serverless data pipeline framework, we executed three different test plans in JMeter, with sizes of 100, 200, and 300 requests, respectively. Each test plan simulated a certain number of users, accessing the application simultaneously and measured the response time and throughput of the application under the above-mentioned loads. The test results are shown in Table III, which represent the reliability of the proposed serverless framework as all the requests were executed successfully without any error. The performance of the serverless framework was consistent. However, the response time for the first test plan showed an outlier behavior where the max response time for higher when it had the least number of concurrent requests, compared to the remaining test plans. This behavior can be observed in Fig. 11. This type of behavior can be caused due to spawning of the containerized dependencies and environment needed with the initial call. However, the results are within the limits, there were no errors and the system performed better at higher levels of the concurrent requests.

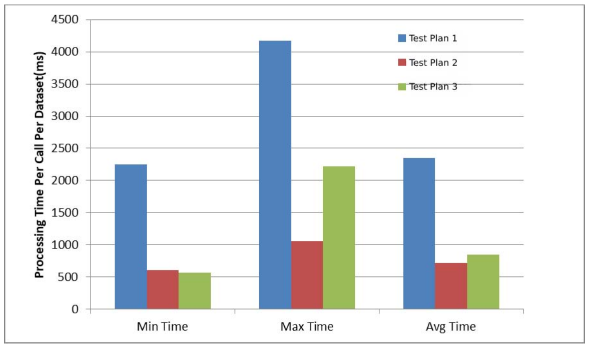

Fig. 11. Processing time per request in Threads

TABLE III
THROUGHPUT OF DL-ENABLED SERVERLESS DATA PIPELINE FRAMEWORK OVER DIFFERENT SAMPLE DATASETS

Label	Size	Success(%)	Throughput(request/sec)	Avg Time(ms)
Test Plan-1	100	100.0	1.924	2530
Test Plan-2	200	100.0	3.966	719
Test Plan-3	300	100.0	5.904	846

VI. CONCLUSION

In this paper, we have presented the design, implementation, and evaluation of the proposed DL-enabled serverless data pipeline framework. We demonstrated that serverless computing provides better scalability, flexibility, and resource utilization compared to the traditional cloud-based infrastructure. In accordance with this approach, we have created DL models that can precisely predict COVID-19 disease from chest X-ray images. We have performed a simulation-based assessment of various DL image classification techniques out of which the VGG-16 image classification model performed the best in terms of accuracy, i.e., 97.66%, whereas the simple Convolution Neural Network 2D approach achieved a lower accuracy of 80.91%. The future research plan involves extending our application to diagnose multiple diseases using a larger and more diverse dataset. Additionally, we plan to explore the feasibility of performing real-time diagnosis on streaming X-ray images using an edge computing framework with advanced ML models. These efforts will build upon our current work and further advance the capabilities and impact of the serverless data pipeline framework.

REFERENCES

[1] K. Yuki, M. Fujiogi, and S. Koutsogiannaki, "Covid-19 pathophysiology: A review," *Clinical Immunology*, vol. 215, p. 108427, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S152166162030262X

[2] A. Sunyaev, *Cloud Computing*. Cham: Springer International Publishing, 2020, pp. 195–236.

[3] R. A. P. Rajan, "Serverless architecture - a revolution in cloud computing," in *2018 Tenth International Conference on Advanced Computing (ICoAC)*, 2018, pp. 88–93.

[4] "Azure function serverless compute," https://azure.microsoft.com/en-us/products/functions/.

[5] "Google cloud functions," https://cloud.google.com/functions.

[6] "Openfaas- serverless framework made simple," https://www.openfaas.com/.

[7] "Knative-serverless and event driven applications," https://knative.dev/docs/.

[8] M. Shahrad, J. Balkind, and D. Wentzlaff, "Architectural implications of function-as-a-service computing," in *Proceedings of the 52nd annual IEEE/ACM international symposium on microarchitecture*, 2019, pp. 1063–1075.

[9] "Getting started with google serverless functions," https://cloud.google.com/functions/docs.

[10] Y. Li, Y. Lin, Y. Wang, K. Ye, and C.-Z. Xu, "Serverless computing: State-of-the-art, challenges and opportunities," *IEEE Transactions on Services Computing*, pp. 1–1, 2022.

[11] A. Pérez, G. Moltó, M. Caballer, and A. Calatrava, "Serverless computing for container-based architectures," *Future Generation Computer Systems*, vol. 83, pp. 50–59, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X17316485

[12] V. Ishakian, V. Muthusamy, and A. Slominski, "Serving deep learning models in a serverless platform," in *2018 IEEE International Conference on Cloud Engineering (IC2E)*, 2018, pp. 257–262.

[13] C. Dehury, P. Jakovits, S. N. Srirama, V. Tountopoulos, and G. Giotis, "Data pipeline architecture for serverless platform," in *European Conference on Software Architecture*. Springer, 2020, pp. 241–246.

[14] S. R. Poojara, C. K. Dehury, P. Jakovits, and S. N. Srirama, "Serverless data pipeline approaches for iot data in fog and cloud computing," *Future Generation Computer Systems*, vol. 130, pp. 91–105, 2022.

[15] J. Cho and Y. Kim, "A design of serverless computing service for edge clouds," in *2021 International Conference on Information and Communication Technology Convergence (ICTC)*. IEEE, 2021, pp. 1889–1891.

[16] K. Suganyadevi S., Seethalakshmi V. & Balasamy, "A review on deep learning in medical image analysis," *International Journal of Multimedia Information Retrieval*, pp. 1–1, 2021.

[17] S. Roy, W. Menapace, S. Oei, B. Luijten, E. Fini, C. Saltori, I. Huijben, N. Chennakeshava, F. Mento, A. Sentelli, E. Peschiera, R. Trevisan, G. Maschietto, E. Torri, R. Inchingolo, A. Smargiassi, G. Soldati, P. Rota, A. Passerini, R. J. G. van Sloun, E. Ricci, and L. Demi, "Deep learning for classification and localization of covid-19 markers in point-of-care lung ultrasound," *IEEE Transactions on Medical Imaging*, vol. 39, no. 8, pp. 2676–2687, 2020.

[18] T. Apostolopoulos I.D., Mpesiana, "Covid-19: automatic detection from x-ray images utilizing transfer learning with convolutional neural networks," *Physical and Engineering Sciences in Medicine*, pp. 635–640, 2020.

[19] G. Yang, J. Liu, M. Qu, S. Wang, D. Ye, and H. Zhong, "Faasrs: Remote sensing image processing system on serverless platform," in *2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC)*. IEEE, 2021, pp. 258–267.

[20] C. Wu, A. N. Toosi, R. Buyya, and K. Ramamohanarao, "Hedonic pricing of cloud computing services," *IEEE Transactions on Cloud Computing*, vol. 9, no. 1, pp. 182–196, 2021.

[21] J. Carreira, P. Fonseca, A. Tumanov, A. Zhang, and R. Katz, "Cirrus: A serverless framework for end-to-end ml workflows," in *Proceedings of the ACM Symposium on Cloud Computing*, 2019, pp. 13–24.

[22] M. Malawski, A. Gajek, A. Zima, B. Balis, and K. Figiela, "Serverless execution of scientific workflows: Experiments with hyperflow, aws lambda and google cloud functions," *Future Generation Computer Systems*, vol. 110, pp. 502–514, 2020.

[23] C. Lin, N. Mahmoudi, C. Fan, and H. Khazaei, "Fine-grained performance and cost modeling and optimization for faas applications," *IEEE Transactions on Parallel and Distributed Systems*, vol. 34, no. 1, pp. 180–194, 2023.

[24] I. F. Jassam, S. M. Elkaffas, and A. A. El-Zoghabi, "Chest x-ray pneumonia detection by dense-net," in *2021 31st International Conference on Computer Theory and Applications (ICCTA)*, 2021, pp. 176–179.

[25] J. Tao, Y. Gu, J. Sun, Y. Bie, and H. Wang, "Research on vgg16 convolutional neural network feature classification algorithm based on transfer learning," in *2021 2nd China International SAR Symposium (CISS)*. IEEE, 2021, pp. 1–3.

[26] S. Nandy, M. Adhikari, M. A. Khan, V. G. Menon, and S. Verma, "An intrusion detection mechanism for secured iomt framework based on swarm-neural network," *IEEE Journal of Biomedical and Health Informatics*, vol. 26, no. 5, pp. 1969–1976, 2021.