Hindawi
Journal of Sensors
Volume 2023, Article ID 3717035, 19 pages
https://doi.org/10.1155/2023/3717035

Research Article

Real-Time Medical Image Classification with ML Framework and Dedicated CNN–LSTM Architecture

Imrus Salehin,[1,2] **Md. Shamiul Islam**,[3] **Nazrul Amin**,[4] **Md. Abu Baten**,[4] **S. M. Noman**,[2,5] **Mohd Saifuzzaman**,[2] **and Serdar Yazmyradov**[1]

[1]*Department of Computer Engineering, Dongseo University, 47 Jurye-ro Sasang-gu, Busan 47011, Republic of Korea*
[2]*Department of Computer Science and Engineering, Daffodil International University, Dhaka, Bangladesh*
[3]*Department of Computer Science and Engineering, Bangladesh University of Business and Technology, Dhaka, Bangladesh*
[4]*Department of Computer Science and Engineering, Northern University Bangladesh, Dhaka, Bangladesh*
[5]*Faculty of Computer Science and Engineering, Frankfurt University of Applied Sciences, Frankfurt, Germany*

Correspondence should be addressed to S. M. Noman; noman33-340@diu.edu.bd

Received 26 May 2023; Revised 19 October 2023; Accepted 25 November 2023; Published 19 December 2023

Academic Editor: Sangsoon Lim

Copyright © 2023 Imrus Salehin et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

In the domain of modern deep learning and classification techniques, the convolutional neural network (CNN) stands out as a highly successful and preferred method for image classification in artificial intelligence. Especially in the medical field, CNN has proven to be an ideal approach for analyzing medical data and accurately identifying diseases. Over the recent years, CNN has demonstrated significant potential and success in various computer vision tasks, with medical image classification being one of the prominent applications. In our study, we introduce a novel custom CNN model called MedvCNN, designed for classifying different types of classes. We conduct experiments with various image sizes to explore their versatility. In addition, long short-term memory (LSTM), a type of recurrent neural network (RNN), is incorporated into our approach. LSTM is specifically tailored to handle sequential data, making it ideal for time series analysis. However, its capabilities extend beyond time series data and are effectively applied to various sequential data types, including sequential vectors derived from image data. One of the key advantages of utilizing LSTM for image classification is its ability to effectively memorize and capture important features in the image data. This feature is particularly advantageous in medical image processing, where precise and accurate identification of key attributes is crucial for successful diagnosis and analysis. Furthermore, our experiments reveal that the hybrid custom LSTM model, MedvLSTM, a RNN algorithm, surpasses other methods in the domain of medical image classification. Our study places significant emphasis on attaining robust classification performance for medical image data through a sophisticated, parameter free approach, complemented by an ablation study, and comprehensive statistical analysis. This comprehensive analysis and evaluation allow us to gain a deeper understanding of the model's effectiveness and its potential impact in the field of medical image analysis. We compare these two approaches to a baseline CNN architecture, aiming to streamline the classification process, reduce time consumption, and improve cost efficiency. Additionally, we present a real-time web-based AutoML framework along with a practical demonstration. Ultimately, our research provides a thorough investigation of the current state-of-the-art in medical image analysis accuracy, focusing on the utilization of neural networks and LSTM.

1. Introduction

Medical data are one of the most sensitive and vital in our modern automation era. The classification of medical images [1] and tabular data [2] is one of the most challenging tasks with regard to accuracy, model performance analysis, and model selection. These research categories are rapidly advancing in the domain of deep learning and AutoML. Furthermore, user-friendly and easy access to machine learning trends have also highlighted this sector. In this era, neural networks have proven to be very useful in the field of image analysis due to their ability to extract important features while disregarding irrelevant ones. This has led to significant improvements in accuracy and performance in the medical sector and data analysis [3]. To address the image classification and analysis tasks, many deep learning methods have

been applied, including decision trees [4], hidden Markov models [5], and deep neural networks (DNN), which are very popular. However, deep learning approaches have proven to be impactful in various applications, including natural language processing [6] and object recognition [7]. On the other hand, model evaluation is crucial for justifying research experiment results with mathematical calculations and selecting the best-fitted model for any experiment based on mathematical observations [8]. There are different model selection approaches that use various algorithms to optimize models, but these consecutive evaluation processes can be inconvenient and have some issues during the training and testing phases.

In 2019, Yadav and Jadhav [9] conducted a study focusing on medical image data classification using various CNN architectures, including VGG-16 and Inception-V3 models. Their research specifically targeted chest X-ray datasets. They applied transfer learning, utilizing pretrained models for their study. In the paper by Zhou et al. [10], they propose an intelligent recommendation system based on patient–physician-generated data using a CNN–RNN integrated framework. In this study, a pretrained CNN model is utilized and fine-tuned using augmented data for brain tumor grade classification [11]. This approach is to achieve desired performance in medical image classification even with a less strongly trained CNN structure. They accomplish this by incorporating an efficient long short-term memory (LSTM) structure at the end of the CNN. LSTM helps capture temporal dependencies in sequential data and enhances the overall classification performance [12]. In recent research, many studies have employed medical image data for classification tasks, utilizing pretrained CNN architectures, and fine-tuning approaches. However, there appears to be limited emphasis on addressing issues related to time consumption and cost efficiency in these studies. To prevent such issues, special methods focusing only on evaluation are used. However, model selection needs customization depending on the settings, evaluation approach, and environment. In our experiments, we applied different custom settings for different datasets due to their diversity, wide acceptance, and availability, which are common characteristics in medical data applications and classification approaches.

This paper aims to demonstrate the impact of deep learning model architectures and the autocustom CNN technique on the accuracy of medical data, which has previously been unexplored. In this study, we work with two types of deep learning models, LSTM [13] and convolutional neural network (ConvNet) [14], which belong to the class of artificial neural networks (ANN). Our experiments show significant improvement in recognition accuracy.

The main focus of our research is as follows:

(1) Developing a low time-consuming and less parameter image classification CNN and LSTM architecture with a particular emphasis on medical image.

(2) Developing a framework designed to surpass benchmark classification methods by utilizing fewer parameters and conducting operations in significantly less time.

(3) Proposing a programing code-free framework that can be used by anyone without any programing knowledge.

In detail, we have discussed this in our further section below. Introduction (Section 1): This section briefly introduces the research topic and its significance. Literature Review (Section 2): A comprehensive review of the existing literature on the topic to establish the current state of knowledge and identify research gaps. Proposed Method (Section 3): Explanation and presentation of the methodology or approach proposed by the study. Background of the Main Method's Algorithm (Section 4): Detailed information about the algorithms or methods used in the proposed approach, along with their theoretical background. Materials and Methods (Section 5): This section describes the materials used in the study (such as datasets, software, and tools) and outlines the specific methods employed. Experiments and Results (Section 6): Presentation of the experiments conducted to validate the proposed method and the results obtained from these experiments. New Framework of AutoML (Section 7): An overview of the new framework developed for AutoML and its key features. Discussion (Section 8): Further discussion on the implications of the results and how they relate to the existing literature. Conclusions (Section 9): A summary of the findings, contributions, and possible future directions for research.

2. Overview of Medical Image Classification Work

Medical data recognition has become a prominent research area in recent years with the rise of deep learning methods. These models have shown significant potential in accurately classifying medical data by leveraging complex architectures and advanced algorithms. In this section, we discuss the application of CNN and LSTM networks for medical image classification. Additionally, we explore the importance of statistical evaluation and proper review in developing sensitive and complex models for medical data recognition. We have conducted a review of recent research focused on medical image classification, with an emphasis on advanced techniques for image classification.

Mateen et al. [15] utilized a widely recognized open-source dataset to investigate the effectiveness of a DNN for detecting diabetic retinopathy in medical images. Their study proposed using the VGG19 DNN architecture, which was found to achieve higher accuracy than other comparable models [15]. Li et al. [16] presented a method for classifying remote sensing images using CNN, stacked autoencoders (SAE), and deep belief networks (DBN). Their approach was evaluated on benchmark datasets for hyperspectral image (HSI) classification [16]. An investigation was conducted to compare the efficacy of the recommended LSTM hyperparameter-based time series forecasting technique against other statistical and computational intelligence methodologies. The study found that the LSTM approach outperformed other tested methods, including ARIMA, ETS, ANN, KNN, RNN, and support vector machine (SVM), in terms of outcome measures [17]. Zhao and Du [18] conducted a comparative

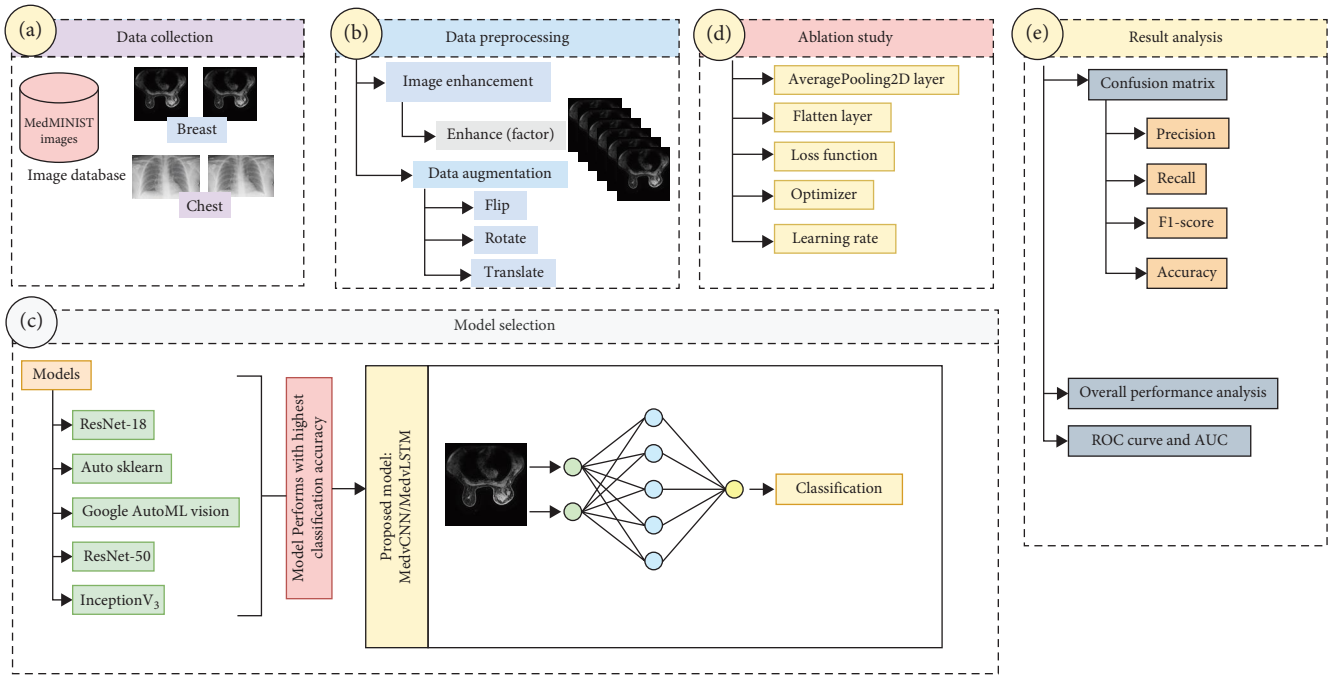

FIGURE 1: A proposed entire methodology for multiclass medical image classification. (a) Data collection, (b) data preprocessing, (c) model selection, (d) ablation study, and (e) result analysis.

study of deep learning techniques for image classification using two distinct datasets and multiple classification methods, such as RAW, principal component analysis (PCA), LDA, SSFC, and LFDA. The results showed that the SSFC method achieved the highest accuracy among all the techniques examined [18]. Gholinejad et al. [19] proposed a K-fold cross-validation (KFCV)-based particle swarm optimization (PSO) model that efficiently operates with a smaller Galactic Co-Prosperity Sphere (GCPS). The proposed approach was tested on four high-resolution satellite images, and the results demonstrated that PSO–KFCV was robust against initial values and dispersion of GCP. Additionally, the proposed method outperformed other state-of-the art metaheuristic algorithms [19]. Nguyen et al. [20] suggested a feature-based and attribute-based (FEAT) approach for human activity recognition (HAR) using the MHealth, Daily and Sport, and RealDisp datasets. The FEAT system was developed by integrating SVM, KNN, and random forest (RF) classifiers, which were trained to recognize new activities with only a few samples from the target activity. The study found that the RF classifier with FEAT outperformed earlier techniques in identifying new activities [20].

In their study, Mehmood et al. [21] conducted an analysis of HAR in the ambulation, transportation, and exercise/fitness categories of MHealth data. They utilized several multiclassification techniques to test their results and evaluated them using seven supervised classifier models. The results of their study indicated that the fuzzy rule model achieved the highest accuracy score at 99.79%, followed by RF at 99.7%, multilayer perceptron neural network at 98.96%, decision tree at 98.58%, K-NN at 95.95%, SVM at 89.1%, and Naive Bayes at 53.18% [21].

Zdravevski et al. [22] proposed five time-series strategies to improve classifier model performance and reduce the operating costs of ambient assisted living. The authors employed five classification methods, including K-NN, logistic regression, Naive Bayes, RF, extremely randomized trees, and SVM, to evaluate the datasets from DaLiAc, MHEALTH, FSP, SBHAR, and SBHARPT. The MHEALTH dataset, which provided 3,232 features after image extraction using multiple methods, achieved the highest accuracy score at 99.8% [22]. In their study, Subasi et al. [23] developed an intelligent m-healthcare the system that utilizes IoT technologies and data mining techniques to accurately recognize human activities with a score of 99.89%. They employed various validation methods, including ANN, SVM, K-NN, C4.5, CART, RF, and rotation forest, on the MHEALTH dataset. The results showed that both SVM and RF achieved the highest accuracy score of 99.89% [23]. The Internet of Medical Things incorporates cutting-edge deep learning techniques to classify health technology. In their study, Raj et al. [24] utilized the OCS method to improve the accuracy of the DL classifier for detecting lung cancer, Alzheimer's disease, and brain images. Their results demonstrated a significant enhancement in the classifier's accuracy [24]. Uddin and Hassan [25] developed a smart healthcare system that utilizes deep convolutional neural network (DCNN). They utilized a classification approach that involved Z-score normalization and Gaussian kernel-based PCA. Their study utilized body sensors and the MHEALTH dataset to classify various human activities, with accuracy scores of 87.99% for ANN, 90.01% for DBN, and 93.9% for the proposed DCNN [25].

3. Proposed Method

In this study, we propose a sequential model for medical multiclass image classification, as depicted in Figure 1. Our

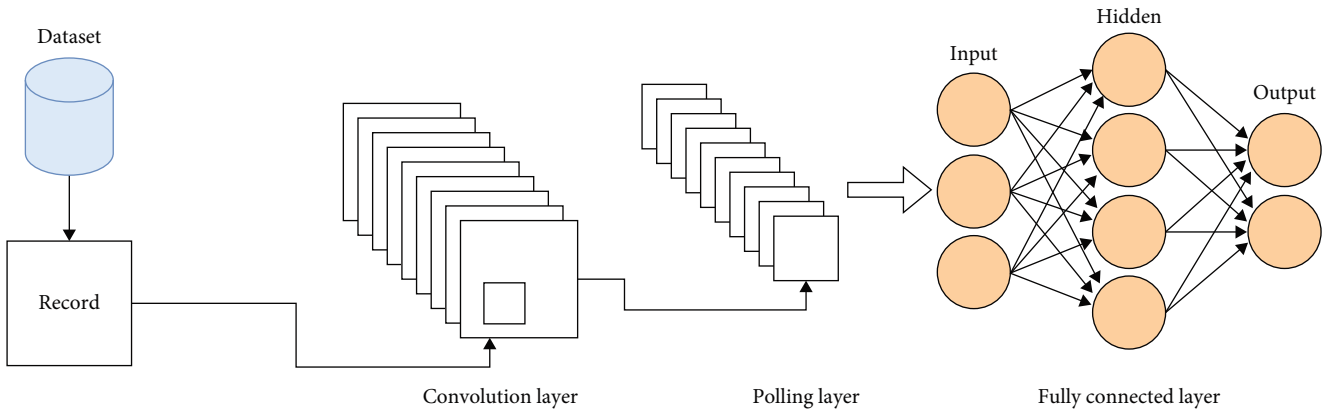

Figure 2: A visualization entire methodology for multiclass medical image classification.

approach begins by focusing on dataset quality and benchmarking, adhering to state-of-the-art (SOTA) guidelines. We carefully select seven outstanding datasets to ensure robustness. Subsequently, data preprocessing is conducted, involving image class definition, data augmentation, and appropriate sampling to prepare the datasets for experimentation. The core of our method lies in the design of a custom model tailored for medical image classification, based on a CNN. We thoroughly explain the architecture and rationale behind this model. Additionally, we perform an ablation study to dissect the CNN strategy and evaluate individual component performances. Finally, we present the statistical outcomes of our experiments, elucidating model accuracy, and key contributing factors while comparing them to relevant parameters. Our sequential model allows for a comprehensive analysis of the proposed approach, providing valuable insights for medical image classification tasks. In our research, we put a significant emphasis on the design of a custom CNN that does not rely on any pretrained models. This allows us to have full control over the architecture and parameters, tailoring them precisely to suit the characteristics of the given problem. We carefully experiment with different configurations of convolutional layers and employ diverse pooling techniques based on the specific demands of the images we are classifying. Furthermore, we address the crucial aspect of image processing and consider ways to optimize the time consumption during the training and inference stages. By fine-tuning the model architecture and hyperparameters, we aim to strike a balance between computational efficiency and accuracy. In addition to solely utilizing CNN, we recognize the potential of incorporating LSTM networks into our approach. By employing a hybrid CNN–LSTM architecture, we leverage the strengths of both CNN and LSTM for our classification task. This enables us to better capture temporal dependencies in sequential data and enhances our model's ability to handle time series or sequential information. The LSTM computation and memory utilization strategy we adopt have shown promising results, leading to improved accuracy in our experiments. We believe that this holistic approach, encompassing the custom CNN design, time efficiency optimization, and hybrid CNN–LSTM utilization, contributes to the overall success of our model and makes it a powerful tool for medical image classification tasks.

4. Algorithm Background

This section provides a detailed description and overview of ANN, specifically CNN and LSTM. We have provided mathematical explanations, training approaches, and architectures for image classification.

4.1. Convolutional Neural Networks. There are a lot of reasons why CNN are one of the most robust DNN architectures applied for solving various problems, especially in the field of computer vision. The primary reason is that these networks are made up of multiple layers of CNN, which leverage convolution operations to extract and learn features from images. CNN has been used extensively in image classification and object detection approaches [26]. The architecture of a typical ConvNet consists of different layers, including input, convolution, pooling, fully connected (FC), and output. Figure 2 illustrates an example of a ConvNet architecture that is based on our medical MedMNIST v2 dataset, which contains different class levels. The convolution layer acquires feature maps that work through elementwise multiplication of kernels from the previous input or output layer. After the pooling layer, the output is passed through a FC layer. This layer combines the features extracted by the convolutional and pooling layers and then produces the final output. The output layer is usually activated by an activation function, such as the SoftMax function, which is commonly used for multiclass classification problems [27]. During the training phase, the error is calculated for the weights of the FC layers and learnable filters in convolutional layers. These weights are then updated by backpropagation methods and the gradient descent algorithm is applied for optimization [28, 29].

4.2. Long Short-Term Memory. LSTM is a type of recurrent neural network (RNN) architecture that is well-suited for modeling sequential data with long-range dependencies [30]. In medical applications, LSTM networks have been used for diagnosing complex conditions and identifying physiological traits in patients, which is often referred to as phenotyping [31]. In our study, we reframed the multilabel classification problem as a phenotyping challenge in clinical time series data [32].

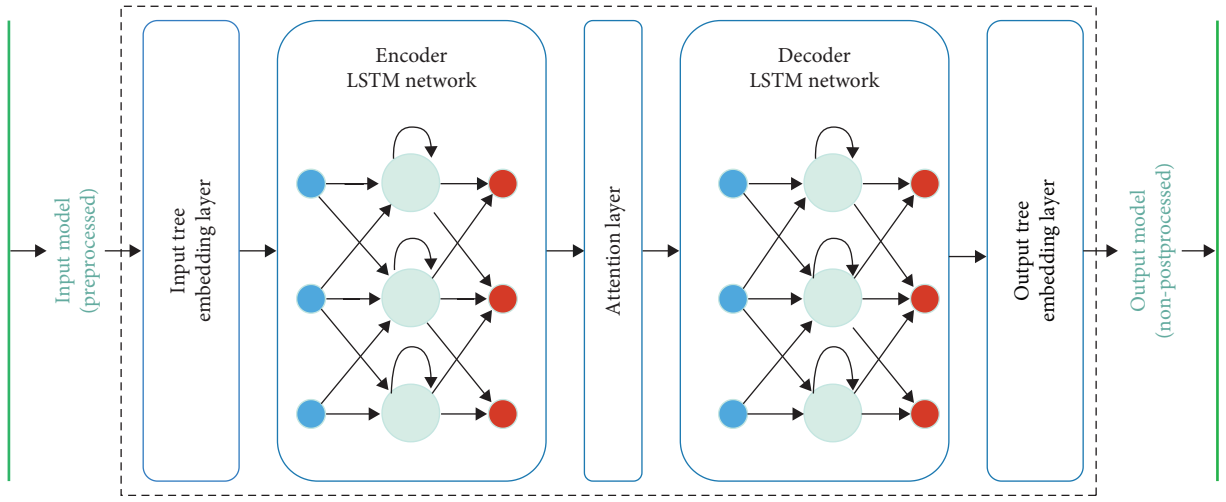

Figure 3: Representation of LSTM gates mechanism.

We develop a classifier using a set of observations $(x^{(1)}, \ldots, x^{(t)})$ to provide hypotheses about the actual labels y in Equations (1) and (2):

$$g_l^{(t)} = \Phi\left(W_l^{gx} h_{(l-1)}^{(t)} + W_l^{gh} h_l^{(t-1)} + b_l^g\right), \tag{1}$$

$$i_l^{(t)} = \sigma\left(W_l^{ix} h_{(l-1)}^{(t)} + W_l^{ih} h_l^{(t-1)} + b_l^i\right). \tag{2}$$

In this case, t denotes the index of a sequence step and t denotes the sequence length for any sample. Memory cells in our proposed LSTM, RNN have forgotten gates but no peephole connections. Because our problem is multilevel, we employ a FC layer on top of the top LSTM layer as the output, followed by an elementwise sigmoid activation function (Equation (3)):

$$f_l^{(t)} = \sigma\left(W_l fx h_{(l-1)}^{(t)} + W_l fh h_l^{(t-1)} + b_l^f\right). \tag{3}$$

The update for a layer of memory cells can be calculated using the below equations $h_l^{(t)}$, where $h_{(l-1)}^{(t)}$ represents the prior layer at the same sequence step either an earlier LSTM layer or the input $x^{(t)}$, and $h_l^{(t-1)}$ represents the same layer (Equation (4)):

$$o_l^{(t)} = \sigma\left(W_l ox h_{(l-1)}^{(t)} + W_l^{oh} h_l^{(t-1)} + b_l^o\right). \tag{4}$$

In these equations, the sigmoid (logistic) function is applied elementwise as denoted by σ, the tanh function is applied elementwise as denoted by, and the Hadamard product is denoted by the symbol $\odot$. Equations (5) and (6) are as follows:

$$s_l^{(t)} = g_l^{(t)} \odot i_l^{(i)} + s_l^{(t-1)} \odot f_l^{(t)}, \tag{5}$$

$$h_l^{(t)} = \gamma\left(s_l^{(t)}\right) \odot o_l^{(t)}, \tag{6}$$

where g is the input node and has a tanh activation, the words "i," "o," and "f" are used to represent the input, output, and forget gates. LSTM gates mechanism is shown in Figure 3.

5. Methodology

CNN and other neural network architectures play a crucial role in learning from medical image data and developing prediction models in medical AI and smart health technology. In this research, we introduce a custom deep learning CNN architecture tailored for medical data analysis, along with a custom LSTM designed for critical image data analysis. These specialized architectures aim to enhance the accuracy and efficiency of medical data interpretation, contributing to the advancement of medical artificial intelligence and smart health technologies.

5.1. Data Processing. DNN are feed-forward neural networks with numerous layers, whereas LSTM are RNN that can process sequential data. In image classification, data processing is as important as learning models. Without a proper dataset and data level we have not trained our model properly way. In our experiment, we preprocess the dataset. Because in MedMNIST v2, some datasets do not contain a good amount of data for a proper experiment. So, we also have applied dataset preprocessing techniques using some well-known preprocess methods along with data augmentation (Flipping, Translating, Rotating) shown in Table 1, which is illustrated in Figure 4.

After the data augmentation, we enhanced some dataset data with sampling. For some datasets, we use sampling because of the imbalance of class and instance. With proper dataset information, we showed in Table 2.

Table 1: Data augmentation parameters setting.

Augmentation techniques ranges	
$\text{Rotate}_{\text{aug}}$	45°
$\text{RotateHorizonta}_{\text{aug}}$	45°
$\text{RotateLeft}_{\text{aug}}$	90°
$\text{RotateRight}_{\text{aug}}$	90°
$\text{Translate}_{\text{aug}}$	y, z (16.2, 5.5)
$\text{HorizontalFlip}_{\text{aug}}$	True
$\text{VerticalFlip}_{\text{aug}}$	True

The data snippet presented here defines a transformation pipeline for a PyTorch dataset. The "transforms.Compose" function creates a sequence of data transforms to be applied to the input data. The pipeline consists of two transforms: "transforms.ToTensor()" converts the input data to a PyTorch tensor and "transforms.Normalize(mean = (0.5), std = (0.5))" standardizes the input data by subtracting the mean and dividing by the standard deviation. Data preparation is a crucial stage in machine learning model construction. It involves cleaning the data, handling missing values, and normalizing or standardizing the data. Our data snippet defines a data transformation pipeline for a PyTorch dataset. The "transforms.Compose" function creates a series of data transforms to be applied to the input data in sequence. The pipeline consists of two transforms: "transforms.ToTensor ()" converts the input data to a PyTorch tensor, and "transforms.Normalize(mean = (0.5), std = (0.5))" standardizes the input data by subtracting the mean and dividing by the standard deviation. The mean and standard deviation values used here are both set to 0.5, which scales each element of the input data to the range [−1, 1]. Overall, this transformation pipeline converts input data to PyTorch tensors and standardizes the data, which can improve the performance and convergence of deep learning models. The "DataLoader" class provides an efficient way to load data in batches, shuffle the data during training, and manage data loading for model evaluation.

5.1.1. Normalization. This is necessary to employ normalization in order to put all of the features within a comparable range and to prevent large-scale features from taking precedence. We use MedMNIST v2 image data to construct our dataset. MedMNIST v2 is one of the most comprehensive free web resources for disease images. There are multiple applications for the dataset. Construct a powerful image classifier capable of classifying any image with several variation class levels. EDA across categories to comprehend visual distinctions and draw generalizations. Combining photos of numerous diseases under a larger umbrella group. Min–max normalization and Z-score normalization are two common methods for normalizing data. Min–max normalization scales the values in a dataset to a specific range, such as [0, 1] or [−1, 1]. This method is often used for datasets with a large range of values or when the data are not normally distributed. It works by subtracting the minimum value from each data point and then dividing it by the range of the data (i.e., the difference between the maximum and minimum values). The choice of normalization method depends on the specific characteristics of the dataset, the goals of the analysis, and the type of machine learning model that will be used. Equation (7) describes the transformation function:

$$Z_{\text{new}}^{*} = \left(\frac{Z_{\text{old}} - Z_{\min}}{Z_{\max} - Z_{\min}}\right). \tag{7}$$

Here, the original, maximum, and minimum values of the feature under consideration are denoted by Z_{old}, $Z_{\max}$, and $Z_{\min}$, respectively. The new normalized value of Z_{old} is denoted by Z^{*} new. Its values fall between [0, 1].

5.1.2. Parameterize CNN Architecture. ConvNet for the medical image classification, we have designed the medical variational convolutional neural network (MedvCNN). After some potential fine-tuning, we set up the customized architecture for medical multiclass image classification in Figure 5. There are five convolutional layers and three FC layers in this CNN model. The input to the network is a 4D tensor with the dimensions (batch size, in channels, height, and width) where "in channels" is the number of input channels and height and width are the image dimensions. The first layer consists of 16 Kernel along with 3×3 followed by batch normalization and the ReLU activation function. This layer's output is passed to the second layer, which is a convolutional layer with 16 filters of size 3×3, batch normalization, ReLU activation function, and max pooling with kernel size 2×2 and stride 2. The third layer consists of 64 convolutional filters of size 3×3, followed by batch normalization and the ReLU activation function. This layer's output is passed to the fourth layer, which is a convolutional layer with 64 sizes of 3×3 filters, batch normalization, and ReLU activation functions. The fifth layer is a convolutional layer with 64 sizes of 3×3 filters and one padding, followed by batch normalization, the ReLU activation function, and max pooling with a 2×2 kernel, and a stride of 2. The fifth layer's output is flattened and sent into the FC layers. The FC layers consist of two hidden layers with 128 units and the ReLU activation function, as well as an output layer with num class units, where num classes are the number of classes in the classification task. Class probability predictions are the output of the last FC layer. In Figure 5, we illustrated the full MedvCNN architecture. The loss function utilized is task dependent. If the classification problem involves many labels and binary classes, the binary cross-entropy with logits loss (BCE_with_log_its_loss) is utilized. If not, cross-entropy loss (Cross_Entropy_Loss) is utilized. The employed optimizer is stochastic gradient descent (SGD) with a learning rate of lr and a momentum of 0.90. Customs model setting for CNN stepwise explanation are given below:

Group 1: This group consists of a convolutional layer with one input channel and 16 output channels, followed by a ReLU activation function. The convolutional filter size is 3×3 and the stride is 1×1.

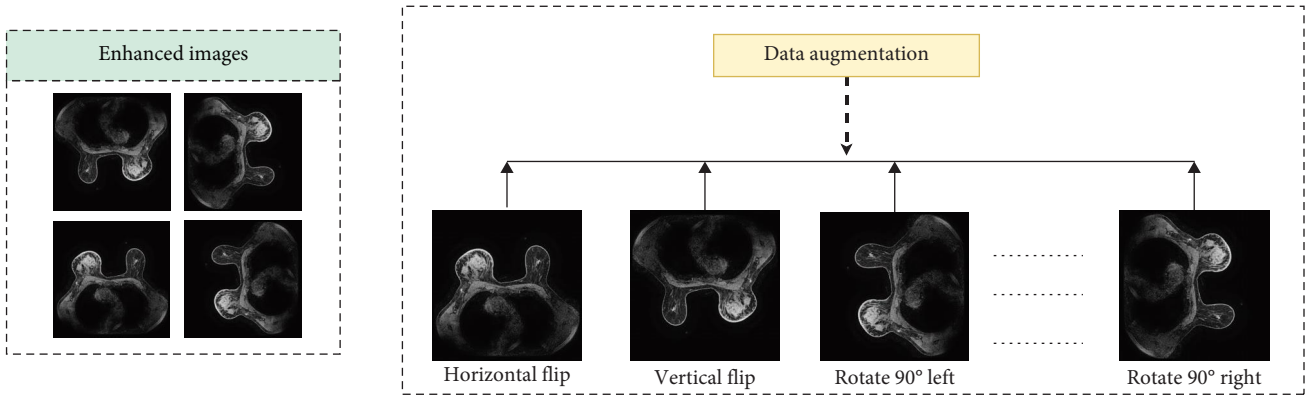

Figure 4: Data preprocessing for image enhancement (augmentation techniques: horizontal flip, rotate 45°, vertical flip, rotate 45° horizontal, rotate 90° left, rotate 90° right, translate).

Table 2: Image amount after doing data augmentation.

Dataset	Instances	Class level	Augmentation info	Training images	Validation images	Testing images
PathMNIST	107,180	9	No	89,996	10,004	7,180
ChestMNIST	112,120	14	No	78,468	11,219	22,433
OCTMNIST	109,309	4	No	97,477	10,832	1,000
PneumoniaMNIST	5,856 (36,506)	2	Yes	26,500	5,600	4,960
TissueMNIST	236,386	8	No	165,466	23,640	47,280
OrganAMNIST	58,850	11	No	34,581	6,491	17,778
BreastMNIST	780 (60,202)	2	Yes	35,600	14,300	10,302

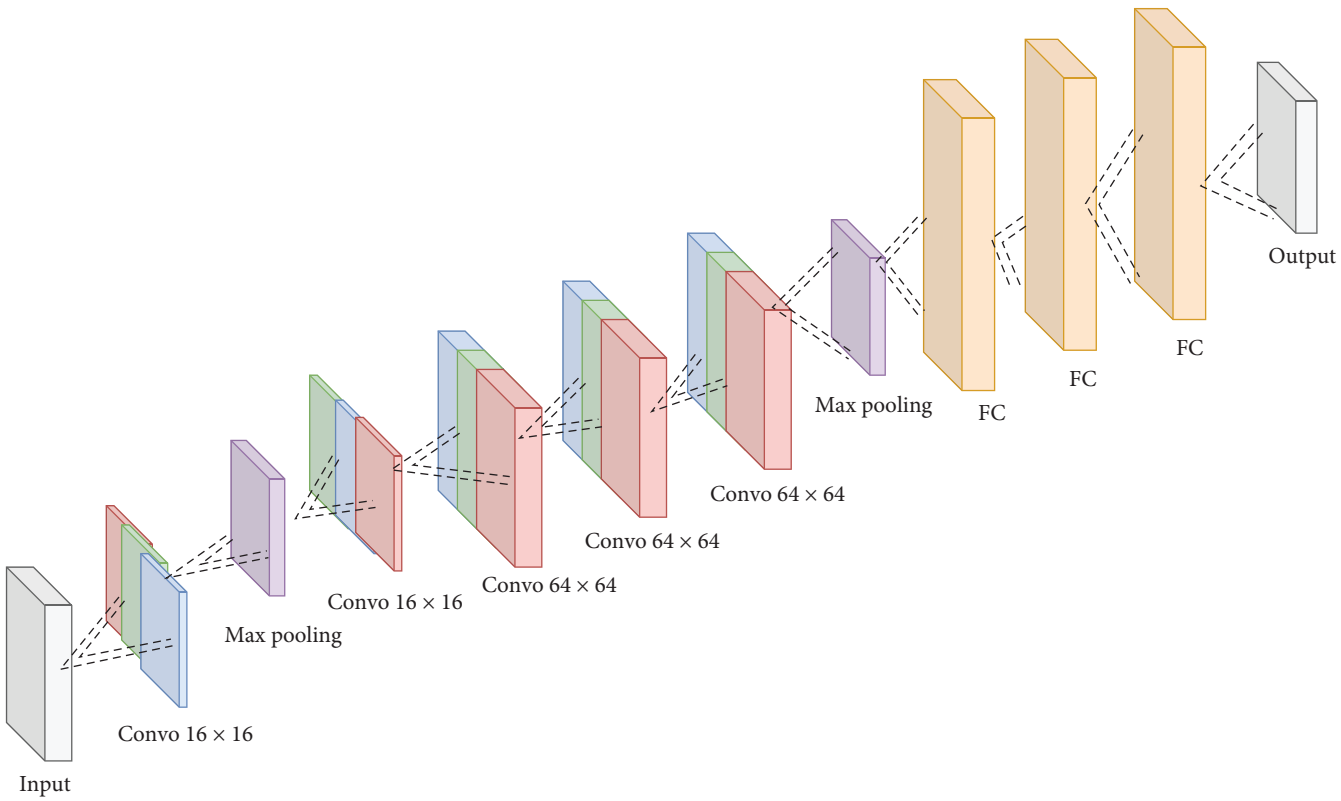

Figure 5: Convolutional neural network (MedvCNN) model architecture.

Table 3: MedvCNN model parameter setting.

Groups	Layer	Kernel size	Stride
Group 1	Conv2d(1,16) ReLU	(3, 3)	(1, 1)
Group 2	Conv2d(16,16) ReLU MaxPool2d	(3, 3)	(1, 1)
Group 3	Conv2d(16,64) ReLU	(3, 3)	(1, 1)
Group 4	Conv2d(64,64) ReLU	(3, 3)	(1, 1)
Group 5	Conv2d(64,64) MaxPool2d	(3, 3)	(1, 1)

Group 2: This group consists of a convolutional layer with 16 input channels and 16 output channels, followed by a ReLU activation function and a max pooling layer with a filter size of 3×3 and a stride of 1×1.

Group 3: This group consists of a convolutional layer with 16 input channels and 64 output channels, followed by a ReLU activation function. The convolutional filter size is 3×3 and the stride is 1×1.

Group 4: This group consists of a convolutional layer with 64 input channels and 64 output channels, followed by a ReLU activation function. The convolutional filter size is 3×3 and the stride is 1×1.

Group 5: This group consists of a convolutional layer with 64 input channels and 64 output channels, followed by a max pooling layer with a filter size of 3×3 and a stride of 1×1.

FC layer 1: Dense layer with 128 neurons and ReLU activation.

FC layer 2: Dense layer with 128 neurons and ReLU activation.

FC layer 3: Dense layer with num_classes neurons.

For more clarification, we show all parameters in Table 3.

Mathematical explanations for the choice of architecture in a CNN for medical data classification are given below:

(i) Convolutional Layers.

In a CNN, each convolutional layer applies a set of filters (also known as kernels) to the input image. Mathematically, this operation is represented as a convolution in Equation (8):

$$\text{FeatureMap} = \text{Convolution}(\text{InputImage}, \text{Filter}). \tag{8}$$

(1) Number of filters: The number of filters in each layer determines the depth of the feature maps produced. More filters lead to a greater capacity to learn diverse features.

(2) Kernel size: The size of the filter (usually 3×3 or 5×5) influences the size of the receptive field and the type of features that can be detected.

(3) Stride: The stride determines how the filter slides across the input image. A larger stride reduces the spatial dimensions of the feature map.

(4) Activation function: After convolution, an activation function (commonly ReLU—rectified linear unit) is applied elementwise. It introduces nonlinearity into the model Equation (9):

$$\text{Output} = \text{ReLU}(\text{Convolution}(\text{Input}, \text{Filter})). \tag{9}$$

(ii) Pooling Layers.

Pooling layers (often MaxPooling) reduce the spatial dimensions of feature maps while retaining the most important information. This is done using a pooling operation, usually max or average pooling. Mathematically, Equation (10) is given as follows:

$$\text{PooledFeatureMap} = \text{Pooling}(\text{FeatureMap}). \tag{10}$$

The size of the pooling window determines the downsampling factor. For example, a 2×2 pooling window reduces the dimensions by half.

(iii) FC Layers.

FC layers perform a weighted sum of the input features, followed by an activation function (often softmax for classification). Mathematically, Equation (11) is given as follows:

$$\text{Output} = \text{Activation}(\text{WeightedSum}(\text{Input})). \tag{11}$$

The number of neurons in each FC layer affects the model's capacity to learn complex decision boundaries. More neurons can capture intricate patterns but may lead to overfitting with insufficient data.

(iv) Regularization Techniques.

Regularization techniques like dropout and batch normalization can be mathematically represented as follows: dropout: In each training iteration, dropout randomly sets a fraction of neurons' outputs to zero. This can be expressed as shown in Equation (12):

$$\text{Output} = \text{Input} \times \text{mask}, \tag{12}$$

where the mask is a binary random variable. Batch normalization normalizes the activations of each layer. Mathematically, it involves subtracting the mean and dividing by the standard deviation of the batch. These mathematical explanations highlight the operations performed in each layer of a CNN and how they contribute to feature extraction, dimensionality reduction, and regularization. The choice of architecture, including the number of layers and their hyperparameters, is often guided by empirical experimentation and the need to strike a balance between model complexity and performance.

To assess the predictive accuracy of our MedvCNN and MedvLSTM parallel network model, we utilize the following evaluation metrics in Tables 4 and 5: mean square error

Table 4: The evaluation loss function of the MedvCNN model with different numbers of convolutional layers.

CNN layers	First layer	Second layer	Third layer	Fourth layer	Fifth layer	MSE/h^2	$RMSE/h$	MAE/h
1	CNN (16)	–	–	–	–	6.04	3.63	2.01
2	CNN (16)	CNN (16)	–	–	–	5.79	3.44	1.79
3	CNN (16)	CNN (16)	CNN (64)	–	–	5.90	3.05	1.56
4	CNN (16)	CNN (16)	CNN (64)	CNN (64)	CNN (64)	4.60	2.99	1.40

Table 5: The evaluation loss function of the MedvLSTM model with different numbers of convolutional layers.

CNN layers	First layer	Second layer	Third layer	Fourth layer	Fifth layer	MSE/h^2	$RMSE/h$
1	CNN (16)	–	–	–	–	6.04	3.63
2	CNN (16)	CNN (16)	–	–	–	5.79	3.44
3	CNN (16)	CNN (16)	CNN (64)	–	–	5.90	3.05
4	CNN (16)	CNN (16)	CNN (64)	CNN (64)	CNN (64)	4.60	2.99

(MSE), root mean square error (RMSE), and mean absolute error (MAE).

It is evident that with the incorporation of various convolutional techniques, the loss function consistently decreases, resulting in improved generalization. In Results section, our accuracy calculations are primarily based on the loss function. In most cases, we employ the MSE as our preferred metric for error calculation.

5.2. Parameterize LSTM Architecture. RNN a prominent subsection of the deep learning family is primarily concerned with sequential data. Processing long-term sequential data is another skill that may be consistently learned via practice. The fact that some of these results have been attained using the conventional RNN model is an intriguing finding. Vanishing gradient problems have already been addressed by newer recurrent network models like LSTM and GRU, therefore they are not a worry while training an RNN for our task. The faults that can be backpropagated through sequence and layers are preserved through the use of gates by LSTM and GRU [33]. They enable recurrent networks to carry on learning over numerous bands of hyperspectral pixels without the danger of overfitting by keeping a more constant error. The benefit of using LSTM over RNN is that they can store more data while processing it. This makes it possible for LSTM to more efficiently understand intricate patterns and connections in data, which is crucial for image classification jobs. Also, LSTM classifies images more accurately than RNN because they learn the critical features while disregarding the unimportant ones. This is due to the fact that LSTM outperforms RNN in capturing long-term dependencies [34]. Figure 6 represents the medical variational long short-term memory (MedvLSTM) entire hybrid architecture for multiclass image classification. The input image is maintained in the batch normalization layer. This batch normalization layer uses the transformation process and maintains the activation function's standard deviation.

The activation function near the range between (0 and 1) thus normalizing this process. The output size of the batch normalization is the same as the input size, making it unusable for LSTM cells. After the dimensional input, we passed the image through the LSTM cell. Tanh (i.e., the hyperbolic tangent) widely used activation function and LSTM cell dropout over fitted data to prevent overfitting because LSTM uses a dropout process [35]. LSTM especially remembers the long-term dependencies along with the regional size of the input image and the entire pattern. After that, the LSTM layer is connected with the convolutional layer. This convolutional layer creates a kernel that produces a tensor of output. As an activation function rectified linear unit (ReLU) has been used in this convolutional layer.

While LSTM are skilled at catching temporal patterns in sequential data, like time series, CNN excel at capturing spatial patterns in data, such as those seen in images [36]. Our hybrid model may recognize complicated patterns in both the spatial and temporal domains by integrating the two networks to make use of each architecture's advantages. The extraction of hierarchical characteristics from the data is made possible via a hybrid model. The CNN layers may learn edges, textures, and other low-level characteristics, and these features are then fed into LSTM layers so they can learn higher level temporal relationships based on the sequential nature of the input. A more efficient and effective feature-learning process may result from this hierarchical approach. CNN is susceptible to overfitting, particularly when working with little data [37].

On the other hand, LSTM can be generalized more effectively in certain circumstances. It may be possible to reduce overfitting by integrating the two models, which would improve performance on untested data. Variable-length sequence processing, which is a typical need in many real-world applications, is a task that LSTM is well suited for. Our hybrid model may handle sequential data of varied duration, such as text with different sentence lengths, by working with

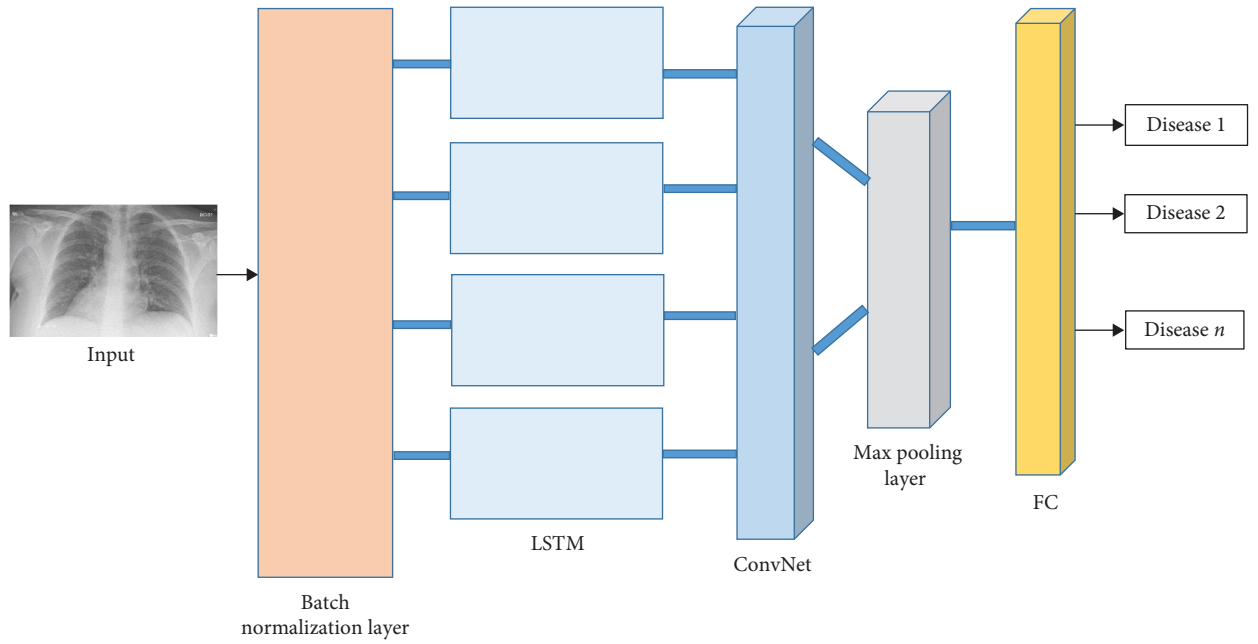

Figure 6: Long short-term memory (MedvLSTM) model architecture.

Table 6: Benchmark test accuracy on four datasets of MedMNIST v2 for improving CNN.

Methods	PathMNIST		ChestMNIST		OCTMNIST		PneumoniaMNIST	
	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC
ResNet-18 [40]	0.983	0.907	0.768	0.947	0.943	0.743	0.944	0.854
Autosklearn [40]	0.934	0.716	0.649	0.779	0.887	0.601	0.942	0.855
Google AutoML vision [40]	0.944	0.728	0.778	0.948	0.955	0.763	0.947	0.878
MedvCNN (our)	0.985	0.841 ± 0.23	0.574	$\mathbf{0.949 \pm 0.30}$	0.947	$\mathbf{0.847 \pm 0.20}$	0.986	$\mathbf{0.945 \pm 0.15}$

Bold values signify the high accuracy.

an LSTM in combination with CNN. In some circumstances, developing a hybrid model as opposed to developing distinct models for various tasks may be more computationally efficient. Sharing parameters can speed up training and save memory requirements if CNN and LSTM can use the same feature extraction layers. The foundation of our hybrid model might be pretrained CNN models for image recognition tasks, which are freely accessible [38].

Instead of having to train everything from scratch, we can concentrate on optimizing the LSTM component for our real-time medical image classification by using pretrained CNN layers. A classical model could still be more appropriate in some circumstances, while alternative cutting-edge designs like transformer-based models might be more effective for specific sequence-based activities [39].

5.3. Experimental Setup. In this section, we explained our CPU setup for multiclass medical image classification. All the models are implemented and computing platform with Intel(R) Core(TM) i7-8700K CPU @ 3.70 GHz and an NVIDIA GeForce GTX 1080i GPU (16 GB memory). During the experiment, for each epoch, we take 28 batch size mini-batch SGD to train the network for 50 epochs. The initial learning rate is 0.001 and we take it for every three epochs with dropout and without dropout.

6. Results

In this section, we describe our model accuracy performance with benchmark datasets. Beginning part, we also discussed the experimental setting and last, we explained the ablation study about the performance of our model.

6.1. Comparative Performance Analysis of Applied Models. In this section, we discuss our performance and compare it with other applied work which is done by other researchers. The terms "area under the ROC curve" (AUC) and "accuracy" are used to refer to the evaluation metrics that are utilized here (ACC). The area under the curve, or AUC, is a measure that does not depend on a threshold in order to evaluate continuous prediction scores, but the area under the curve (ACC) utilizes a threshold in order to evaluate discrete prediction labels (or argmax). In contrast, ACC and AUC are less likely to suffer from class imbalance. In Table 6, we summarize our excrement with MedMNIST v2 and compared with benchmarking SOTA data, and our MedvCNN outperforms several image datasets. Besides this MedMNIST v2, we also show the MedMNIST dataset accuracy for Table 7 along with the MedvCNN outcome. We run several same excrements and set a standard deviation with our main accuracy. Table 8 shows the performance of MedvLSTM with the

Table 7: Benchmark test accuracy on three datasets of MedMNIST v2 for improving CNN.

Methods	Breast_MNIST		Tissue_MNIST		OrganA_MNIST	
	AUC	ACC	AUC	ACC	AUC	ACC
Autosklearn [40]	0.836	0.803	0.828	0.532	0.963	0.762
MedvCNN (our)	0.898	0.831 ± 0.22	0.888	0.595 ± 0.58	0.990	0.878 ± 0.57

Table 8: Benchmark test accuracy on each dataset of MedMNIST v2 for improving LSTM.

Methods	Path_MNIST		Chest_MNIST		OCT_MNIST		Pneumonia_MNIST	
	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC
MedvLSTM (our)	0.962	0.841 ± 0.22	0.924	0.886 ± 0.58	0.908	0.917 ± 0.18	0.906	0.945 ± 0.06

Table 9: Comparison of the prediction performance loss value of different models for LSTM.

Models	MSE/h	RMSE/h	MAE/h	Main parameter settings
LSTM [42]	9.18	3.03	2.45	LSTM layer (one layer) Neurons (32)
Bi_LSTM [43]	8.46	2.90	2.34	Bi_LSTM layer (one layer) Neurons (32)
LSTM_attention [44]	7.66	2.77	2.23	LSTM layer (one layer) Neurons (128)
MedvLSTM (our)	5.53	2.23	2.20	LSTM layer (three layers) Neurons (128)

MedMNIST v2 dataset. Given that our datasets do not include any particularly extreme examples of class imbalance, the average class composition (ACC) has the potential to be another valuable indicator as well. Even though there are a great many extra measurements, we only utilize AUC and ACC since doing so enables us to examine everything in the simplest way that is possible. This is despite the fact that there are a great many additional metrics. Therefore, in order to make it easier to compare the efficacy of different methods, we present both the average area under the curve (AUC) and the average area under the curve (ACC) for the MedMNIST v2 dataset [41]. In Tables 6–8, we present our experiment results, comparing them with other benchmark methods on the MedMNIST dataset. In the case of MedvCNN with PathMNIST, our accuracy is slightly lower than ResNet-18, with a difference of approximately 0.6%. However, on the ChestMNIST dataset, our model performs exceptionally well, surpassing existing models with an accuracy of over 94%. Moreover, for OCTMNIST and PneumoniaMNIST, our approach outperforms Autosklearn, Google AutoML Vision, and ResNet-18, demonstrating the effectiveness and superiority of our proposed method on these datasets. In our experiments, we observed that BreastMNIST, TissueMNIST, and OrganAMNIST datasets all perform admirably with MedvCNN, achieving an accuracy of over 80%. These results demonstrate the efficacy of our proposed MedvCNN model in accurately classifying medical images from these datasets. In Table 8, we can observe that the hybrid CNN–LSTM model, which incorporates PathMNIST, ChestMNIST, OCTMNIST, and PneumoniaMNIST datasets, demonstrates superior performance in terms of accuracy. This accuracy is notably close to that of the improved CNN accuracy. The results highlight the effectiveness of our hybrid CNN–LSTM approach, showcasing its potential in accurately classifying medical images from these datasets. In Table 9, we present a comparison of loss values derived from related studies. We executed this model using the medical dataset (Pneumonia_MNIST) and generated Table 9 to showcase the results. Additionally, we provide insight into the loss values in conjunction with the specific neuron parameter settings employed in our research.

6.2. Ablation Study. Ablation study is a commonly used parameter tuning in grid search strategy used to choose the parameters (batch size, epochs, learning rate, optimizer, and dropout). Furthermore, the ablation study ensures the proposed method's model robustness. For all four case studies, we use the seven datasets and use the average result for accurate measurement.

6.2.1. CASE 1: MaxPooling2D. This case studied different pooling layers in CNN along with our model MaxPooling2D layer.Vld_Acc (%) and Vld_Ls (%) are better perform for our MaxPooling2D. The Tst_Acc (%) [Test Accuracy] and Tst_Ls (%) [Test Loss] results outperform then other polling layers. We calculate this accuracy for the overall dataset and input our result on average in Table 10.

6.2.2. CASE 2: Flatten Layer. In this examination, a flattened layer has been added after the feature maps of the previous FC layers. We also change this layer with some other layers and get the entire result (Table 11).

6.2.3. CASE 3: Loss Functions. To utilize the proper Loss function, we study several different types of loss functions in our experiment. After the experiment, we get a better result using the CrossEntropy Loss which shows the outperforming result in Table 12.

6.2.4. CASE 4: Optimizers and Learning Rates. To examine the best learning rate and optimizer, we use two types of optimizers. After the experiment, we take SGD and 0.001

Table 10: Experiment with different Pooling2D layer.

Layer names	Vld_Acc (%)	Vld_Acc (%)	Tst_Acc (%)	Tst_Ls (%)	Performance
GlobalAveragePooling2D	95.89	0.10	95.95	0.29	Accuracy_dropped
AveragePooling2D	94.69	0.34	95.01	0.35	Accuracy_dropped
MaxPooling2D	**96.89 ± 0.22**	**0.30**	**97.56 ± 0.32**	**0.20**	Identical
GlobalMaxPooling2D	96.01	0.36	96.50	0.26	Accuracy_dropped

Bold values signify the high accuracy and identical results.

Table 11: Experiment with the position setup of flatten layer.

Layer names	Vld_Acc (%)	Vld_Acc (%)	Tst_Acc (%)	Tst_Ls (%)	Performance
GlobalAveragePooling2D	80.89	0.20	88.95	0.69	Accuracy_dropped
AveragePooling2D	76.69	0.74	80.01	0.36	Accuracy_dropped
Flatten layer	**95.89 ± 0.18**	**0.10**	**98.56 ± 0.22**	**0.14**	Identical
GlobalMaxPooling2D	90.12	0.30	92.51	0.20	Accuracy_dropped

Bold values signify the high accuracy and identical results.

Table 12: Experiment with different loss functions.

Loss_functions	Vld_Acc (%)	Vld_Acc (%)	Tst_Acc (%)	Tst_Ls (%)	Performance
Cosine similarity	92.09	0.20	88.95	0.19	Accuracy_dropped
Mean squared error	93.86	0.12	96.11	0.11	Accuracy_dropped
Cross-entropy loss	**96.88 ± 0.06**	**0.11**	**97.56 ± 0.22**	**0.80**	Identical

Bold values signify the high accuracy and identical results.

Table 13: Experiment with altering the optimizers and learning rates.

Optimizers	Learning_Rates	Vld_Acc (%)	Vld_Acc (%)	Tst_Acc (%)	Tst_Ls (%)	Performance
	0.00001	92.09	0.20	93.95	0.13	Accuracy_dropped
Adam	0.0001	93.86	0.12	96.11	0.11	Accuracy_dropped
	0.001	96.88	0.11	96.98	0.12	Accuracy_dropped
	0.00001	93.09	.25	94.99	0.19	Accuracy_dropped
SGD	0.0001	90.66	0.15	95.13	0.16	Accuracy_dropped
	0.001	**97.56 ± 0.04**	**0.10**	**97.95 ± 0.22**	**0.12**	Identical

Bold values signify the high accuracy and identical results.

learning rate for better performance. Table 13 shows the experiment results.

6.3. Performance Evaluation. In this section, we discuss our experiment result evaluation process along with the confusion matric and accuracy metrics. Last, we show ROC for the model performance analysis. Resulting in the accuracy metric in Table 14, we apply the prominent machine learning evaluation formula. In Equations (13) and (14), TP refers to true positive and TN means true negative. Another FP is false positive and FN means false negative.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}, \tag{13}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}. \tag{14}$$

To determine the recall value, we applied Equation (15):

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FP}}. \tag{15}$$

A confusion matrix is frequently used to assess the model's performance. The confusion matrix is generated by computing the TP, TN, FP, and FN values. In this matrix, TP and TN represent the number of samples that are accurately identified as positive and negative, respectively. On the other hand, FP and FN refer to the number of samples that are incorrectly classified as positive and negative, respectively [45]. In our accuracy metrics (Table 14), we showed seven MedMNIST v2 datasets precision, recall and F1-score PathMNIST and ChestMNIST, TissueMNIST and OrganAMNIST, respectively, showed 88%, 94%, 59%, and 86% which outperformed other previous studies along with

Table 14: Accuracy metrics of the entire model for each dataset.

Scenario	Class	Precision	Recall	F1-score
PathMNIST	0	0.97	0.96	0.96
	1	0.88	0.87	0.87
	2	0.95	0.90	0.91
	3	0.70	0.72	0.73
	4	0.68	0.86	0.87
	5	0.92	0.90	0.91
	6	0.88	0.85	0.86
	7	0.78	0.80	0.81
	8	0.86	0.88	0.87
	Accuracy			0.88
	Macro avg	0.84	0.86	0.86
ChestMNIST	0	0.70	0.75	0.74
	1	0.89	0.88	0.89
	2	0.96	0.92	0.93
	3	0.94	0.95	0.94
	4	0.92	0.91	0.90
	5	0.84	0.86	0.87
	6	0.90	0.91	0.92
	7	0.95	0.92	0.93
	8	0.93	0.90	0.90
	9	0.92	0.93	0.92
	10	0.97	0.95	0.96
	11	0.91	0.89	0.90
	12	0.94	0.91	0.93
	13	0.87	0.78	0.90
	Accuracy			0.94
	Macro avg	0.88	0.95	0.94
TissueMNIST	0	0.62	0.85	0.71
	1	0.46	0.01	0.01
	2	0.45	0.28	0.34
	3	0.50	0.63	0.56
	4	0.43	0.35	0.39
	5	0.39	0.21	0.27
	6	0.76	0.53	0.63
	7	0.50	0.57	0.53
	Accuracy			0.59
	Macro avg	0.51	0.43	0.43
OrganAMNIST	0	0.64	0.85	0.73
	1	0.87	0.54	0.67
	2	0.71	0.79	0.75
	3	0.89	0.75	0.81
	4	0.59	0.69	0.63
	5	0.61	0.68	0.64
	6	0.96	0.85	0.90
	7	0.95	0.96	0.96
	8	0.98	0.98	0.98
	9	0.80	0.78	0.79
	10	0.66	0.58	0.62
	Accuracy			0.86
	Macro avg	0.79	0.77	0.78
PneumoniaMNIST	0	0.93	0.96	0.97
	1	0.90	0.92	0.93
	Accuracy			0.93
	Macro avg	0.91	0.94	0.95
BreastMNIST	0	0.80	0.82	0.81
	1	0.85	0.89	0.87
	Accuracy			0.82
	Macro avg	0.82	0.85	0.84
OctMNIST	0	0.54	0.94	0.69
	1	0.82	0.80	0.81
	2	0.78	0.16	0.26
	3	0.78	0.84	0.81
	Accuracy			0.82
	Macro avg	0.73	0.68	0.5

multiclass image classification. The other two datasets PneumoniaMNIST and BreastMNIST binary classifications results are quite good and four class OCTMNIST midlevel image classification accuracy is better than the other models. The seven dataset statistical analyses have been done based on our MedvCNN medical image classification model. After that, we represent a visualization of the performance with the confusion matrix in Figure 7. Among the seven datasets, we have randomly selected and analyzed two binary datasets and two multiclass datasets for confusion matric calculation. In this case, the test dataset contains different amounts of data along with different class levels. BreastMNIST dataset has 10,302 test data and our CNN model successfully classifies 10,156 image data and PneumoniaMNIST classifies 4,929 data from 4,960 test data. In the multiclass classification (OCTMNIST dataset and TissueMNIST), the successful classification rate outperforms then another multiclass classification experiment. In Figure 8, the ROC curve presents the proposed MedvCNN. The first image (a) shows a multiclass binary-class (OCTMNIST Dataset) and the second one (b) is a binary-class (for ChestMNIST dataset) performance curve. The ROC curve is constructed with two axes (x-axis and y-axis). The range of these curves' values is from 0 to 1. If the value is closer to 0 that means it performing poorly. In parallel, the closer the value is to 1, that means better the performance of the model is. For the performance analysis, we randomly select one binary classification and one multiclass dataset for ROC curve calculation. In our case, our model curves have quite high performance both in binary and multiclass.

6.4. Time and Complexity Evaluation. In this section, we delve into the analysis of time complexity and model parameters specific to the CNN architecture. Time complexity refers to the computational time required for a given algorithm or model to process data. In the context of CNN architectures, understanding the time complexity is crucial

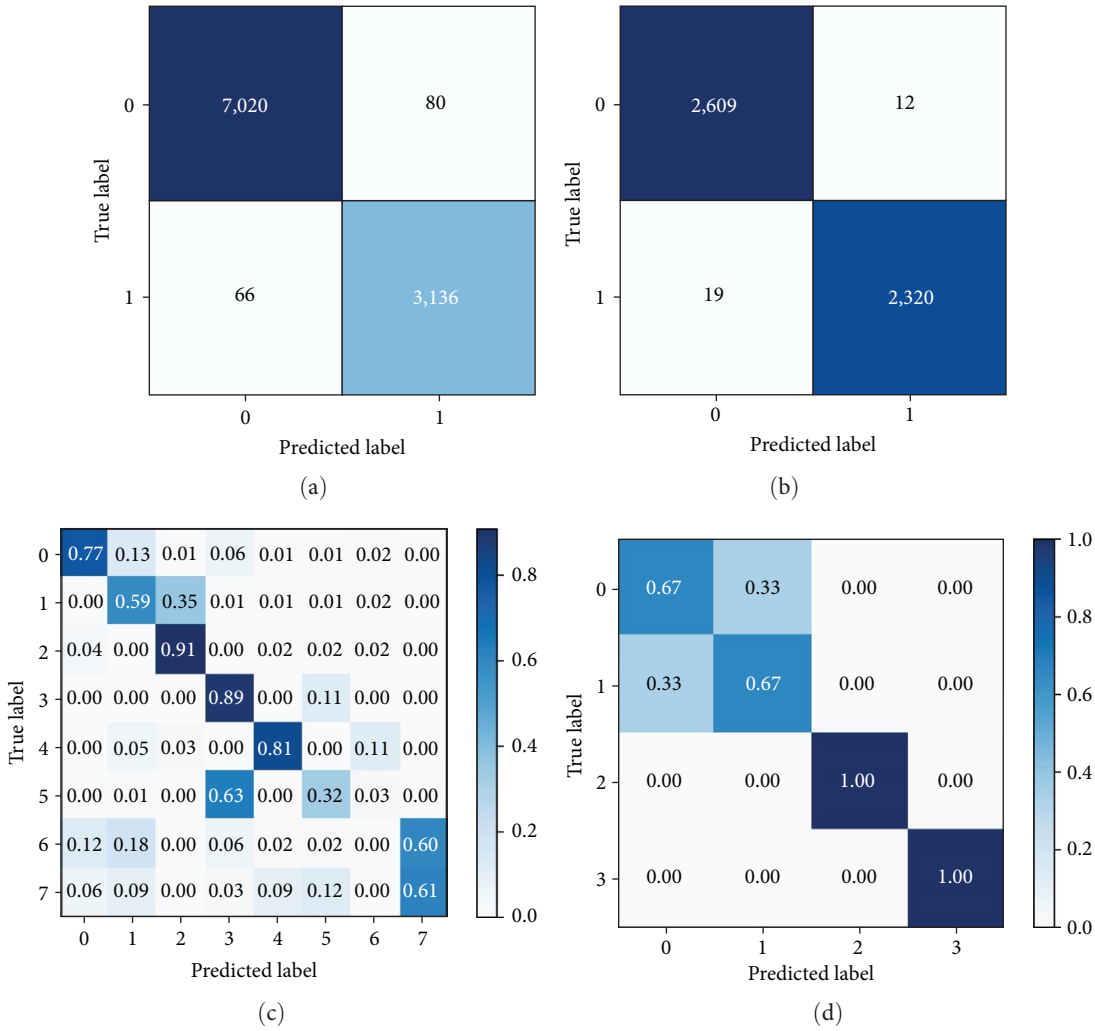

Figure 7: Confusion matrix of applied model. (a) For BreastMNIST dataset, (b, c) for TissueMNIST and for PneumoniaMNIST dataset, and (d) for OCTMNIST dataset.

for assessing the computational efficiency of the model. It involves calculating how the processing time scales with input size. The time complexity of a CNN depends on several factors, including the number of layers, the size of each layer, and the operations performed in each layer (e.g., convolutions, pooling, and FC layers). It is important to provide a breakdown of the time complexity at different stages of the CNN model, explaining which components contribute most to the computational cost. Model parameters are the weights and biases that the CNN learns during the training process. Analyzing model parameters is essential for assessing the model's capacity and potential for overfitting. Calculate the total number of parameters in your CNN model. This includes the weights and biases in each layer. In Table 15, we present the model parameters (Per (M)) and training time (in seconds). M is denoted million, and s is denoted second. We mention here every epoch training time average.

7. AutoML Framework Design

In this section, we explained the framework of AutoML with deep learning and website combination. In Figure 9, we show this process about the working process. Medical image classification website involves a user visiting the website and uploading an image for classification through the homepage. The image is received by the back-end of the website which includes a pretrained image classification model. The model analyzes the image and generates a classification label and probability. Medical image classification website involves a user visiting the website and uploading an image for classification through the homepage (Figure 10). The image is received by the back-end of the website which includes a pretrained image classification model. Algorithm 1 shows the coding explanation. Upon receiving the image, the back-end processes it through the pretrained model's inference pipeline. The model performs a series of sophisticated

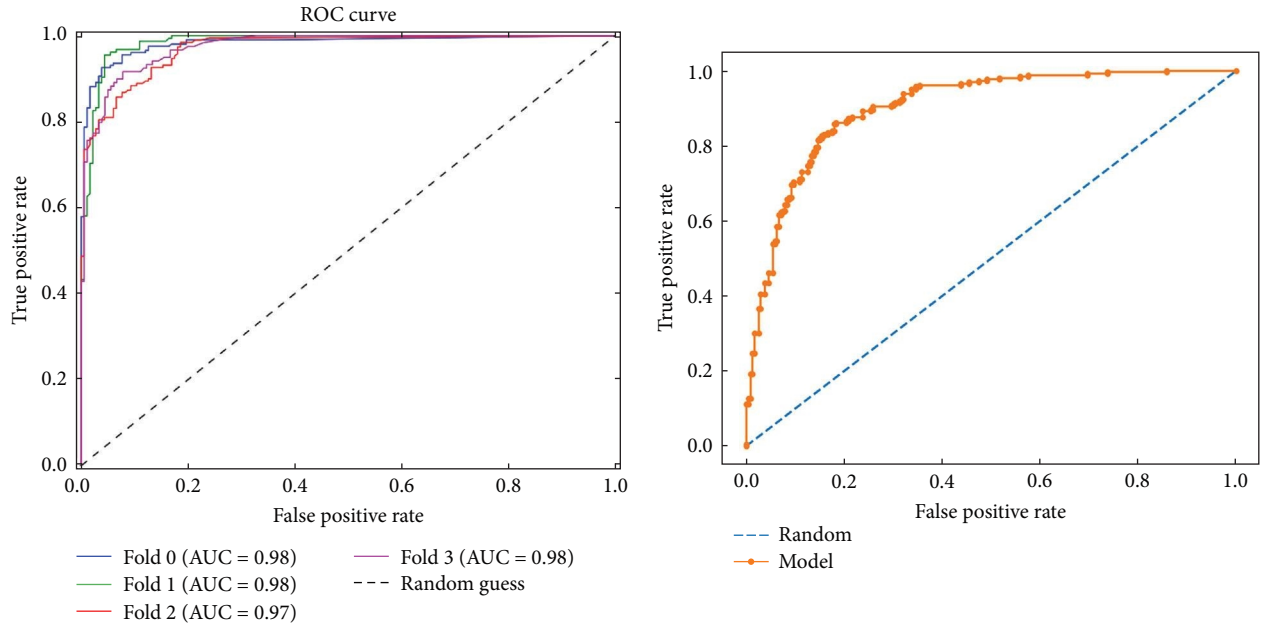

Figure 8: Illustration of receiver operating characteristics curve (ROC) for multiclass (a) and binary-class (b).

Table 15: Time and complexity evaluation metric.

Methods	PathMNIST		ChestMNIST		OCTMNIST		PneumoniaMNIST	
	Train time (s)	Per (M)	Train time (s)	Per (M)	Train time (s)	Per (M)	Train time (s)	Per (M)
ResNet-18	2,180.30	11.2	2,080.33	11.2	1,880.10	11.2	2,380.13	11.2
Autosklearn	1,589.02	9.4	1,429.25	9.4	1,389.26	9.4	1,601.20	9.4
Google AutoML vision	1,709.24	10.6	1,826.21	10.6	1,726.08	10.6	1,869.02	10.6
MedvCNN (our)	455.06	5.56	512.06	5.56	489.02	5.56	556.9	5.56
MedvLSTM (our)	689.03	7.23	786.02	7.23	689.23	7.23	786.23	7.23

mathematical operations, such as convolutions and activations, to analyze the image's pixel values and extract high-level features relevant to medical diagnosis. The model then generates a classification label representing the most likely medical category or condition depicted in the image. Additionally, the model produces a probability score that reflects its confidence in the assigned label. The model analyses the image and generates a classification label and probability. The result is displayed on a results page generated by the backend and served by the front end. The website's server handles incoming requests, processes the image classification, and serves the results to the user. The user can repeat the process by submitting another image for classification.

8. Discussion

Our research focuses on medical public image datasets, and we have developed our custom CNN and hybrid LSTM model. The reason for creating a new custom model is that existing CNN models like Res-Net, AlexNet, and VGG-Net, which are pretrained on generic image datasets, do not perform optimally when faced with different types of image sizes and high-resolution medical data. Medical images often have unique characteristics and higher resolution than the images used to pretrain standard models. Consequently, off-the-shelf CNN models may not capture the relevant features effectively, leading to suboptimal performance on medical data. To address this limitation, we have designed a custom CNN architecture tailored specifically to handle medical images with varying sizes and high levels of detail.

Additionally, to further enhance our model's performance, we have incorporated LSTM components into the architecture. LSTM are powerful sequential models that can effectively capture temporal dependencies in the data, which is particularly relevant in medical imaging where context and sequential patterns can play a vital role in diagnosis and analysis. By leveraging both custom CNN and LSTM components, our hybrid model aims to improve the accuracy and robustness of medical image analysis. We believe that this approach will yield more accurate results and potentially aid medical professionals in making better-informed decisions based on the visual information provided by the images. To ensure the robustness and classification performance of our model, we conduct a thorough evaluation using

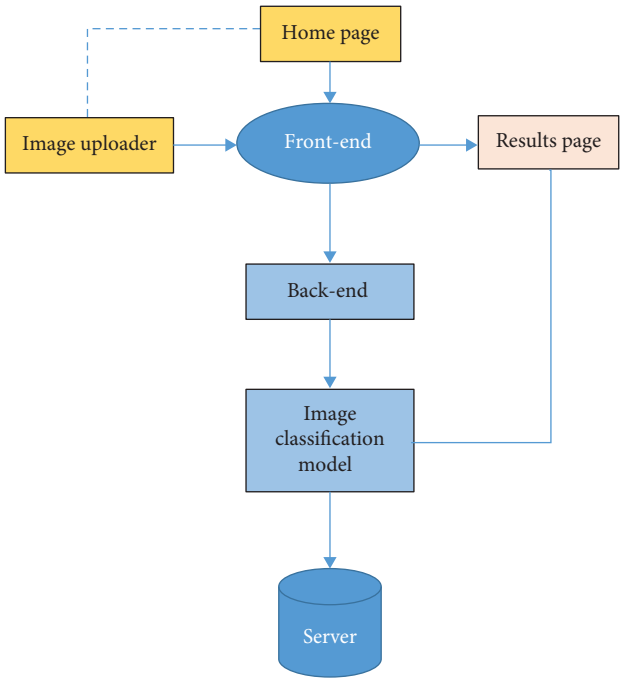

Figure 9: Framework architecture for AutoML medical image classification.

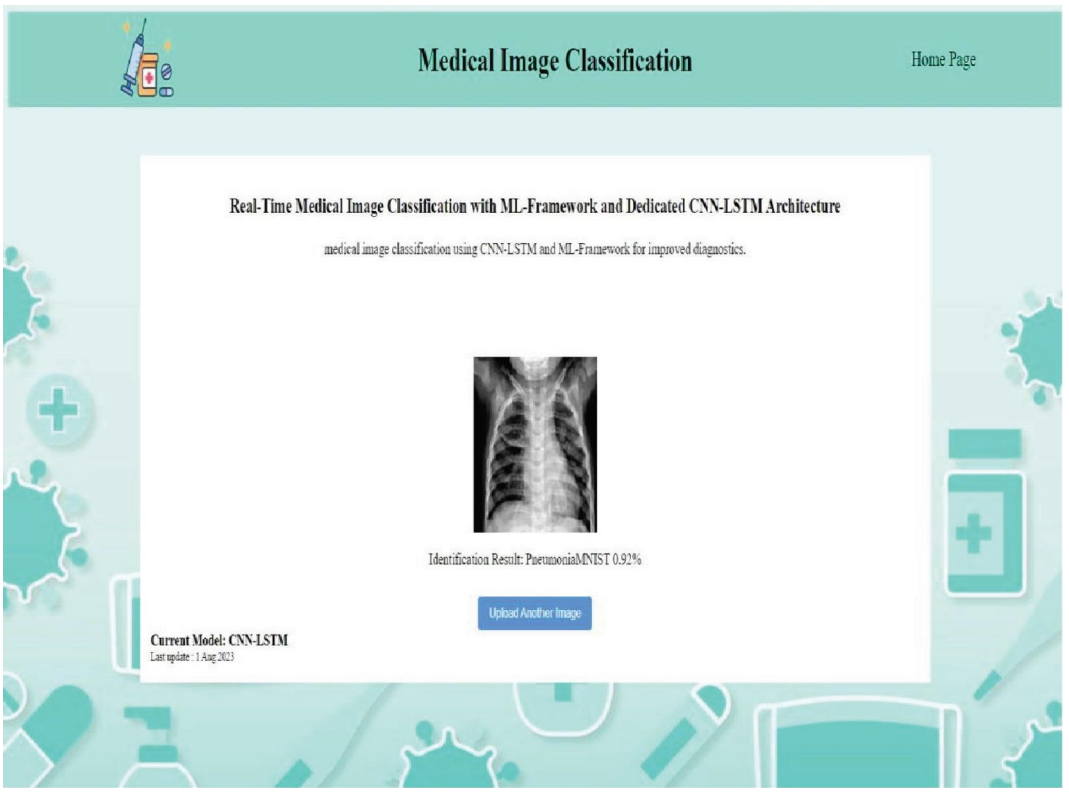

Figure 10: Web application for AutoML medical image classification.

traditional machine learning techniques and statistical analysis. Our evaluation process involves assessing the model's performance across seven different classes of data. We measure the accuracy of our model to determine how well it performs on the test dataset. A high accuracy score indicates that the model is making correct predictions for a significant portion of the data. In addition to accuracy, we also examine other performance metrics like precision, recall, and

```
Input: sample image Output: return model predicted image.
app = Flask (name)
ROUTE "/"
return home page template
ROUTE "/upload" with the method POST
image = get uploaded image
classification = classify_image (image)
return upload page template with the classification
ROUTE "/results"
return results page template with classification
FUNCTION classify_image (image)
model = load pre-trained image classification model
ROUTE "/"
return home page template
```

Algorithm 1: Full process of the web framework.

F1-score. These metrics provide a more comprehensive understanding of the model's behavior for each class. Precision measures the proportion of true positive predictions among all positive predictions, while recall (also known as sensitivity) represents the proportion of true positive predictions among all actual positive instances in the data. The F1-score is the harmonic mean of precision and recall, offering a balanced evaluation of the model's performance. Furthermore, our study addresses the issue of time consumption, which is a significant concern when dealing with medical image datasets. Many pretrained models require considerable time for adaptation to new datasets, and they may perform poorly when faced with small or differently sized data. However, our model is specifically designed to tackle this problem. We understand that many medical institutes may not have access to high-performance devices, and generating results in a timely manner is crucial. Therefore, our model takes into consideration the constraints of resource availability and aims to provide efficient and accurate results even with limited computing resources. Moreover, we have developed a user-friendly web model framework that does not require a programing background for operation. This framework allows medical professionals and individuals without programing expertise to easily classify diseases using normal medical images.

9. Conclusion

This study delved into the realm of medical image classification, exploring advanced approaches such as LSTM and CNN architectures while benchmarking against the MedMNIST datasets. After conducting numerous experiments, we thoroughly analyzed the performance of various model configurations by customizing and randomizing hyperparameters. The results yielded promising outcomes, with our custom MedvCNN and MedvLSTM models outperforming the existing state-of-the art methods on the benchmark dataset by a significant margin. This achievement demonstrates the potential of our novel approaches to revolutionize medical image classification, offering improved accuracy and efficiency. However, it is essential to emphasize that model selection and evaluation remain critical aspects of any model development process. In our research, we employed two comprehensive model evaluation processes, ensuring that the chosen models performed exceptionally well, as anticipated. Moving forward, the success of our custom models and evaluation methods paves the way for further advancements in medical image classification. As we continue our research, we will explore more extensive datasets and real-world medical scenarios to validate the robustness and generalizability of our approaches. Ultimately, our work aims to contribute to 20 dedicated CNN–LSTM architecture for medical image classification the ongoing progress in medical image analysis, facilitating better diagnostic and decision-making processes in the healthcare domain. We are excited about our future plans, which include developing a neural architecture search (NAS) for medical image classification. NAS is a cutting-edge technique that automates the process of finding optimal neural network architectures. By applying NAS to our medical image classification problem, we aim to discover novel and highly efficient models tailored specifically to handle the complexities of medical data. With the integration of NAS into our research, we expect to further improve the performance of our models and potentially uncover new insights that can aid in the early detection and accurate diagnosis of various medical conditions.

Abbreviations

AutoML: Automated machine learning
ARIMA: Auto regressive integrated moving average
DBN: Deep belief network
ETS: Error-trend-seasonality
FC: Fully connected
HIS: Hyperspectral image
LDA: Latent dirichlet allocation
PCA: Principal component analysis
SSFC: Spectral–spatial feature-based classification
SAE: Sparse autoencoder
SGD: Stochastic gradient descent.

Data Availability

Data will be shared upon request from the authors. The data used to support the findings of this study are available online.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Authors' Contributions

I.S. and S.Y. contributed to supervision. I.S. and M.S. contributed to validation data curation, project administration, software, supervision, validation, and visualization. M.S.I. and N.A. contributed to formal analysis. I.S., M.A.B., and S.M.N. contributed to investigation. M.S.I. and I.S.

contributed to methodology. I.S., M.S.I., and S.M.N. contributed to writing—original draft. I.S, M.S., and M.A.B. contributed to writing—review and editing. All authors have read and agreed to the published version of the manuscript.

Acknowledgments

The authors wish to thank members of the Dongseo University Machine Learning/Deep Learning Research Lab and Daffodil International University Innovation Lab and anonymous referees for their helpful comments on earlier drafts of this paper. They especially thanks to Prof. Moon for the review and experiment assistance.

References

[1] M. D. Abràmoff, Y. Lou, A. Erginay et al., "Improved automated detection of diabetic retinopathy on a publicly available dataset through integration of deep learning," *Investigative Opthalmology & Visual Science*, vol. 57, no. 13, pp. 5200–5206, 2016.

[2] S. Azizi, B. Mustafa, F. Ryan et al., "Big self-supervised models advance medical image classification," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 3478–3488, IEEE Computer Society, Los Alamitos, CA, USA, October 2021.

[3] X. Chen, X. Wang, K. Zhang et al., "Recent advances and clinical applications of deep learning in medical image analysis," *Medical Image Analysis*, vol. 79, Article ID 102444, 2022.

[4] T. Phan, "Improving activity recognition via automatic decision tree pruning," in *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication*, pp. 827–832, Association for Computing Machinery, September 2014.

[5] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, "Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine," in *Ambient Assisted Living and Home Care. IWAAL 2012*, J. Bravo, R. Hervás, and M. Rodríguez, Eds., vol. 7657 of *Lecture Notes in Computer Science*, pp. 216–223, Springer, Berlin, Heidelberg, 2012.

[6] E. Batbaatar, M. Li, and K. H. Ryu, "Semantic-emotion neural network for emotion recognition from text," *IEEE Access*, vol. 7, pp. 111866–111878, 2019.

[7] J. M. Gandarias, A. J. García-Cerezo, and J. M. Gómez-de-Gabriel, "CNN-based methods for object recognition with high-resolution tactile sensors," *IEEE Sensors Journal*, vol. 19, no. 16, pp. 6872–6882, 2019.

[8] D. Gholamiangonabadi, N. Kiselov, and G. Katarina, "Deep neural networks for human activity recognition with wearable sensors: leave-one-subject-out cross-validation for model selection," *IEEE Access*, vol. 8, pp. 133982–133994, 2020.

[9] S. S. Yadav and S. M. Jadhav, "Deep convolutional neural network based medical image classification for disease diagnosis," *Journal of Big Data*, vol. 6, no. 1, pp. 1–18, 2019.

[10] X. Zhou, Y. Li, and W. Liang, "CNN-RNN based intelligent recommendation for online medical pre-diagnosis support," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 18, no. 3, pp. 912–921, 2021.

[11] M. Sajjad, S. Khan, K. Muhammad, W. Wu, A. Ullah, and S. W. Baik, "Multi-grade brain tumor classification using deep CNN with extensive data augmentation," *Journal of Computational Science*, vol. 30, pp. 174–182, 2019.

[12] Ş. Öztürk and U. Özkaya, "Gastrointestinal tract classification using improved LSTM based CNN," *Multimedia Tools and Applications*, vol. 79, no. 39-40, pp. 28825–28840, 2020.

[13] R. C. Staudemeyer and E. R. Morris, "Understanding LSTM—a tutorial into long shortterm memory recurrent neural networks," arXiv preprint arXiv: 1909.09586, 2019.

[14] O. Yildirim, U. B. Baloglu, R.-S. Tan, E. J. Ciaccio, and U. R. Acharya, "A new approach for arrhythmia classification using deep coded features and LSTM networks," *Computer Methods and Programs in Biomedicine*, vol. 176, pp. 121–133, 2019.

[15] M. Mateen, J. Wen, Nasrullah, S. Song, and Z. Huang, "Fundus image classification using VGG-19 architecture with PCA and SVD," *Symmetry*, vol. 11, no. 1, Article ID 1, 2019.

[16] Y. Li, H. Zhang, X. Xue, Y. Jiang, and Q. Shen, "Deep learning for remote sensing image classification: a survey," *WIREs Data Mining and Knowledge Discovery*, vol. 8, no. 6, Article ID e1264, 2018.

[17] H. Abbasimehr, M. Shabani, and M. Yousefi, "An optimized model using LSTM network for demand forecasting," *Computers & Industrial Engineering*, vol. 143, Article ID 106435, 2020.

[18] W. Zhao and S. Du, "Spectral–spatial feature extraction for hyperspectral image classification: a dimension reduction and deep learning approach," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, no. 8, pp. 4544–4554, 2016.

[19] S. Gholinejad, A. A. Naeini, and A. Amiri-Simkooei, "Robust particle swarm optimization of RFMs for high-resolution satellite images based on K-fold cross-validation," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 12, no. 8, pp. 2594–2599, 2019.

[20] L. T. Nguyen, M. Zeng, P. Tague, and Z. Joy, "Recognizing new activities with limited training data," in *Proceedings of the 2015 ACM International Symposium on Wearable Computers*, pp. 67–74, Association for Computing Machinery, September 2015.

[21] A. Mehmood, A. Raza, A. Nadeem, and S. Umair, "Study of multi-classification of advanced daily life activities on shimmer sensor dataset," *International Journal of Communication Networks and Information Security (IJCNIS)*, vol. 8, no. 2, pp. 86–92, 2016.

[22] E. Zdravevski, P. Lameski, V. Trajkovik et al., "Improving activity recognition accuracy in ambient-assisted living systems by automated feature engineering," *IEEE Access*, vol. 5, pp. 5262–5280, 2017.

[23] A. Subasi, M. Radhwan, R. Kurdi, and K. Kholoud, "IoT based mobile healthcare system for human activity recognition," in *2018 15th Learning and Technology Conference (L&T)*, pp. 29–34, IEEE, Jeddah, Saudi Arabia, February 2018.

[24] R. J. S. Raj, S. J. Shobana, I. V. Pustokhina, D. A. Pustokhin, D. Gupta, and K. Shankar, "Optimal feature selection-based medical image classification using deep learning model in internet of medical things," *IEEE Access*, vol. 8, pp. 58006–58017, 2020.

[25] M. Z. Uddin and M. M. Hassan, "Activity recognition for cognitive assistance using body sensors data and deep convolutional neural network," *IEEE Sensors Journal*, vol. 19, no. 19, pp. 8413–8419, 2018.

[26] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[27] Y. Lecun and Y. Bengio, "Convolutional networks for images, speech, and time series," in *The Handbook of Brain Theory and Neural Networks*, M. A. Arbib, Ed., pp. 1–9, MIT Press, 1995.
[28] I. Goodfellow, Y. Bengio, and C. Aaron, *Deep Learning*, MIT Press, 2016.
[29] S.-I. Amari, "Backpropagation and stochastic gradient descent method," *Neurocomputing*, vol. 5, no. 4-5, pp. 185–196, 1993.
[30] X. Ren, H. Gu, and W. Wei, "Tree-RNN: tree structural recurrent neural network for network traffic classification," *Expert Systems with Applications*, vol. 167, Article ID 114363, 2021.
[31] Z. Che, D. Kale, W. Li, M. T. Bahadori, and L. Yan, "Deep computational phenotyping," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 507–516, Association for Computing Machinery, August 2015.
[32] H. Harutyunyan, H. Khachatrian, D. C. Kale, G. Ver Steeg, and A. Galstyan, "Multitask learning and benchmarking with clinical time series data," *Scientific Data*, vol. 6, no. 1, Article ID 96, 2019.
[33] L. Medsker and L. C. Jain, *Recurrent Neural Networks: Design and Applications*, CRC Press, 1999.
[34] Q. Liu, F. Zhou, R. Hang, and X. Yuan, "Bidirectional-convolutional LSTM based spectral-spatial feature learning for hyperspectral image classification," *Remote Sensing*, vol. 9, no. 12, Article ID 1330, 2017.
[35] I. Salehin and D.-K. Kang, "A review on dropout regularization approaches for deep neural networks within the scholarly domain," *Electronics*, vol. 12, no. 14, Article ID 3106, 2023.
[36] F. M. Shiri, T. Perumal, N. Mustapha, and R. Mohamed, "A comprehensive overview and comparative analysis on deep learning models: CNN, RNN, LSTM, GRU," arXiv preprint arXiv:2305.17473, 2023.
[37] H. Su, C. R. Qi, Y. Li, and L. J. Guibas, "Render for CNN: viewpoint estimation in images using cnns trained with rendered 3d model views," in *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 2686–2694, IEEE, Santiago, Chile, December 2015.
[38] M. Umer, Z. Imtiaz, S. Ullah, A. Mehmood, G. S. Choi, and B.-W. On, "Fake news stance detection using deep learning architecture (CNN-LSTM)," *IEEE Access*, vol. 8, pp. 156695–156706, 2020.
[39] Z. Liu, R. A. Roberts, M. Lal-Nag, X. Chen, R. Huang, and W. Tong, "AI-based language models powering drug discovery and development," *Drug Discovery Today*, vol. 26, no. 11, pp. 2593–2607, 2021.
[40] J. Yang, R. Shi, and B. Ni, "MedMNIST classification decathlon: a lightweight AutoML benchmark for medical image analysis," in *2021 IEEE 18th International Symposium on Biomedical Imaging (ISBI)*, pp. 191–195, IEEE, Nice, France, April 2021.
[41] J. Yang, R. Shi, D. Wei et al., "MedMNIST v2—a large-scale lightweight benchmark for 2D and 3D biomedical image classification," *Scientific Data*, vol. 10, no. 1, Article ID 41, 2023.
[42] Z. Zhao, W. Chen, X. Wu, P. C. Y. Chen, and J. Liu, "LSTM network: a deep learning approach for short-term traffic forecast," *IET Intelligent Transport Systems*, vol. 11, no. 2, pp. 68–75, 2017.
[43] J. Wang, F. Hu, and L. Li, "Deep bidirectional long short-term memory model for short-term traffic flow prediction," in *Neural Information Processing. ICONIP 2017*, D. Liu, S. Xie, Y. Li, D. Zhao, and E. S. El-Alfy, Eds., vol. 10638 of *Lecture Notes in Computer Science*, pp. 306–316, Springer, Cham, 2017.
[44] L. Li, R. Zhang, J. Sun, Q. He, L. Kong, and X. Liu, "Monitoring and prediction of dust concentration in an open-pit mine using a deep-learning algorithm," *Journal of Environmental Health Science and Engineering*, vol. 19, no. 1, pp. 401–414, 2021.
[45] J. D. Rodriguez, A. Perez, and J. A. Lozano, "Sensitivity analysis of k-fold cross validation in prediction error estimation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 3, pp. 569–575, 2010.