Elsevier has created a Monkeypox Information Center in response to the declared public health emergency of international concern, with free information in English on the monkeypox virus. The Monkeypox Information Center is hosted on Elsevier Connect, the company's public news and information website.

Elsevier hereby grants permission to make all its monkeypox related research that is available on the Monkeypox Information Center - including this research content - immediately available in publicly funded repositories, with rights for unrestricted research re-use and analyses in any form or by any means with acknowledgement of the original source. These permissions are granted for free by Elsevier for as long as the Monkeypox Information Center remains active.

Expert Systems With Applications 216 (2023) 119483

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Deep transfer learning approaches for Monkeypox disease diagnosis

Md Manjurul Ahsan [a,*], Muhammad Ramiz Uddin [b], Md Shahin Ali [c], Md Khairul Islam [c], Mithila Farjana [b], Ahmed Nazmus Sakib [d], Khondhaker Al Momin [e], Shahana Akter Luna [f]

[a] Industrial and Systems Engineering, University of Oklahoma, Norman, OK 73019, USA
[b] Department of Chemistry and Biochemistry, University of Oklahoma, Norman, OK, 73019, USA
[c] Department of Biomedical Engineering, Islamic University, Kushtia 7003, Bangladesh
[d] Department of Aerospace and Mechanical Engineering, University of Oklahoma, Norman, OK, 73019, USA
[e] Department of Civil Engineering, Daffodil International University, Dhaka, 1341, Bangladesh
[f] Medicine & Surgery, Dhaka Medical College & Hospital, Dhaka, 1000, Bangladesh

ARTICLE INFO

Keywords:
Deep learning
Disease diagnosis
Image processing
Monkeypox virus
Machine learning

ABSTRACT

Monkeypox has become a significant global challenge as the number of cases increases daily. Those infected with the disease often display various skin symptoms and can spread the infection through contamination. Recently, Machine Learning (ML) has shown potential in image-based diagnoses, such as detecting cancer, identifying tumor cells, and identifying coronavirus disease (COVID)-19 patients. Thus, ML could potentially be used to diagnose Monkeypox as well. In this study, we developed a Monkeypox diagnosis model using Generalization and Regularization-based Transfer Learning approaches (GRA-TLA) for binary and multiclass classification. We tested our proposed approach on ten different convolutional Neural Network (CNN) models in three separate studies. The preliminary computational results showed that our proposed approach, combined with Extreme Inception (Xception), was able to distinguish between individuals with and without Monkeypox with an accuracy ranging from 77% to 88% in Studies One and Two, while Residual Network (ResNet)-101 had the best performance for multiclass classification in Study Three, with an accuracy ranging from 84% to 99%. In addition, we found that our proposed approach was computationally efficient compared to existing TL approaches in terms of the number of parameters (NP) and Floating-Point Operations per Second (FLOPs) required. We also used Local Interpretable Model-Agnostic Explanations (LIME) to explain our model's predictions and feature extractions, providing a deeper understanding of the specific features that may indicate the onset of Monkeypox.

1. Introduction

Monkeypox is an infectious disease caused by the Zoonotic Orthopoxvirus, which is closely related to both cowpox and smallpox belongs to the poxviridae family (a member of the genus Orthopoxvirus) (McCollum & Damon, 2014). It is mostly transmitted by monkeys and rodents; nevertheless, the human-to-human spread is also extremely prevalent (Alakunle, Moens, Nchinda, & Okeke, 2020). The virus was first identified in a monkey's body in 1958 in a laboratory in Copenhagen, Denmark (Moore & Zahra, 2022). In 1970, the Democratic Republic of the Congo recorded the first human case of Monkeypox during an intensified effort to eradicate smallpox (Nolen et al., 2016). Monkeypox is usually exposed in the central and western part of Africa and affects many individuals who reside near the tropical rainforests (Khodakevich, Ježek, & Messinger, 1988). The virus itself contaminates when a person comes in close contact with another infected person, animal, or material. It is transmitted through direct body contact, animal bites, respiratory droplets, or mucous of the eye, nose, or mouth (Nguyen, Ajisegiri, Costantino, Chughtai, & MacIntyre, 2021). Some early-stage symptoms of patients infected with Monkeypox include fever, body aches, and fatigue, wherein the long-term effect has a red bump on the skin (CDC, 2022c).

Although Monkeypox is not significantly contagious compared to Coronavirus Disease (COVID)-19 reported so far, the cases continue to rise. There were only 50 Monkeypox cases in 1990 in West and Central Africa (Doucleff, 2022). However, the cases rose to 5000 in 2020. Monkeypox is claimed to occur only in Africa in the past,

* Corresponding author.
E-mail addresses: ahsan@ou.edu (M.M. Ahsan), muhammadramizuddin@gmail.com (M.R. Uddin), shahinbme.iu@gmail.com (M.S. Ali), khairul.ice06@gmail.com (M.K. Islam), mithilafarjana@ou.edu (M. Farjana), nazmus.sakib@ou.edu (A.N. Sakib), momin.ce@diu.edu.bd (K.A. Momin), shahanaakterluna123@gmail.com (S.A. Luna).

https://doi.org/10.1016/j.eswa.2022.119483
Received 28 August 2022; Received in revised form 24 December 2022; Accepted 27 December 2022
Available online 5 January 2023

wherein in 2022, the identification of the individuals infected by the virus is reported by several other non-African countries in Europe and the United States (WHO, 2022). According to the Centers for Disease Control and Prevention (CDC), in 2022, Monkeypox cases are reported by 94 nations, and the number of total patients is around 83,424 as of December 21, 2022 (CDC, 2022a). As an effect, tremendous anxiety and fear among the people are slowly growing, often reflected through the individual's opinion on social media (Bragazzi, Khamisy-Farah, Tsigalou, Mahroum, & Converti, 2022).

Currently, there is no appropriate treatment for the Monkeypox virus, according to the guidelines provided by the CDC (CDC, 2022b). Nevertheless, to cope up with the urgent need, the CDC approved two oral drugs, Brincidofovir and Tecovirimat, which have mainly been used to treat the smallpox virus, have now been used to treat the Monkeypox virus (Adler et al., 2022). Vaccination is the ultimate solution to the Monkeypox virus. Despite the availability of Food and Drug Administration (FDA)-approved vaccines for the Monkeypox virus, they have not yet been administered to humans in the United States. In other countries, the vaccines for the smallpox virus are used to treat the Monkeypox virus (Park, 2022).

The diagnosis procedure of the Monkeypox disease includes initial observations of the unusual characteristics of skin lesions present and the existing history of exposure. However, the definitive way to diagnose the virus is to test skin lesions using electron microscopy. In addition, the Monkeypox virus can be confirmed using Polymerase Chain Reaction (PCR) (ISU, 2022), which is currently being used extensively in diagnosing the COVID-19 patients (Ahsan et al., 2021; Ahsan, Alam, Trafalis and Huebner, 2020; Ahsan et al., 2020; Ahsan, Nazim, Siddique and Huebner, 2021).

Transfer learning (TL) is an emerging branch of Machine Learning (ML) domains with demonstrated potential in various medical imaging and diagnosis fields. For instance, Dey et al. (2021) use deep Convolutional Neural Network (CNN)-based approaches to detect the malaria parasites in the blood cell images automatically (Dey, Nath, Biswas, Nath, & Ganguly, 2021). Vijayalakshmi et al. (2020) proposed a combination of Visual Geometry Group(VGG)-19 and Support Vector Machine (SVM)-based models to detect malaria from microscopic images. Their proposed model accurately detected malaria-infected images 93.1% of the time (Vijayalakshmi et al., 2020). Gao et al. (2018) proposed a shallow-deep CNN-based model to improve the performance of the CNN model on breast cancer diagnosis; their proposed methods achieved an accuracy of 85% (Gao et al., 2018). Wang et al. (2020) constructed a modified inception-based model using 453 Computed Tomography (CT) scan images and attained an accuracy of 73.1% (Wang, Lin, & Wong, 2020). Sandeep et al. (2022) proposed a low complex CNN to detect skin diseases such as Psoriasis, Melanoma, Lupus, and Chickenpox. They show that using exiting VGGNet; it is possible to detect skin disease 71% accurately using image analysis (Sandeep, Vishal, Shamanth, & Chethan, 2022). In comparison, their proposed solution demonstrates the best results by achieving an accuracy of around 78%. Velasco et al. (2019) proposed a smartphone-based skin disease identification utilizing MobileNet and reported around 94.4% accuracy in detecting patients with Chickenpox symptoms (Velasco et al., 2019). Roy et al. (2019) utilized different segmentation approaches to detect skin diseases such as acne, candidiasis, cellulitis, chickenpox, etc. (Roy et al., 2019).

Over the years, DL has exhibited remarkable success and profoundly influenced the conceptual foundations of ML and Artificial Intelligence (AI). The application of DL-based approaches shows promising results in many industrial domains where it can overcome traditional approaches, which are often costly, time-consuming, and unsuitable for large-scale operations. For instance, Banan et al. (2020) applied DL-based feature extraction to develop automated carp species identifications in the fishery industry. The proposed method achieved around 100% accuracy in identifying four carp species that do not require expert opinion and can be performed in real-time (Banan, Nasiri, & Taheri-Garavand, 2020).

Fan et al. (2020) introduced Karhunen-Loève (KL) decomposition, the multilayer perceptron (MLP), and the Long Short-Term Memory (LSTM) network, named KL-MLP-LSTM for estimating the temperature distributions during the thermal process. That application can be applied to a parabolic distributed parameter with feedback input signals in any field that deals with nonlinear systems. The author claimed that the proposed method could be used in hydrology, reducing the computational cost and, therefore, cheaper to run (Fan, Xu, Wu, Zheng, & Tao, 2020).

Lin et al. (2022) proposed DL-based models to forecast the mean monthly groundwater level using data from 33 different monitoring piezometers. These models include three different layers of Gated Recurrent Unit (GRU) structures and a hybrid of Variational Mode Decomposition (VMD)-GRU. The GRU2 × model is chosen as the best model based on performance evaluation metrics, with an R2 of 0.86, a Root Means Square Error (RMSE) of 0.18 m, and a Total Grade (TG) of 6.21 in the validation stage. The hybrid VMD–GRU model also performed well, with an RMSE of 0.16 m, an R2 of 0.92, and a TG of 3.34 (Lin et al., 2022).

Due to the over-fitting prone of DL to train data, the expressive and trainable hypothesis spaces are not always guaranteed true performance of DL models. This leads to the study of generalization, which helps to identify the model's ability to adapt appropriately to new, previously unseen data drawn from the same distribution as the one used to create the model (Zhang, Ballas and Pineau, 2018).

In clinical diagnosis, generalization is required to validate the model's compatibility and ability to use in the real world. For instance, Kermany et al. (2018) introduced explainable AI approaches to provide their model's generalization and interpretation. The author tested their AI-based pneumonia detection models and used expert opinion to validate their findings (Kermany et al., 2018).

However, their study also addresses that rapid radio-logic image interpretation is not always possible due to the low-resource settings, which can be easily observed during the onset of COVID-19 and the recent outbreak of Monkeypox disease.

There have been very few studies that considered TL approaches for detecting Monkeypox disease. For example, Abdelhamid et al.'s (2022) GoogleNet deep network classifies Monkeypox images using the Al-Biruni Earth Radius Optimization algorithm. The author claimed that their proposed models achieved around 98.8% accuracy. However, the author did not use any model interpretation techniques, which are necessary to understand the behavior of the model's predictions (Abdelhamid et al., 2022).

Sitaula and Shahi (2022) used different TL approaches such as Visual Geometry Groups (VGG)-M, Residual Network(ResNet), Inception, MobileNet, DenseNet, etc. Their preliminary computational result shows that the best performance was observed for the ensemble approaches instead of a single model. The author reported 87.13% accuracy, 85.44% precision, 85.47% recall, and an 85.40% F1-score. For model interpretation, the author used Local Interpretable Model-Agnostic Explanations (LIME)-based approaches (Sitaula & Shahi, 2022). However, one of the main limitations of their proposed study is that the author did not provide any explanation as to whether their model is computationally expensive or not. Other than that, the experiment was performed using a single dataset. Therefore, it is hard to interpret how their proposed model might perform on different datasets.

Akin et al. (2022) used CNN-based approaches to develop auxiliary decision support systems for Monkeypox disease diagnosis. Their proposed models achieved around 98.25% accuracy, 96.55% sensitivity, 100% specificity, and a 98.25% F1-score. The author uses the Gradient-weighted Class Activation Mapping (GradCAM) approach to identify the potentially infected regions (Akin, Gurkan, Budak, & Karataş, 2022). However, the higher accuracy was reported only for the training set, and there were no indications of how their proposed model will

perform on the testing set. The performance of the TL model is more challenging on the test or unseen data compared to the training sample itself. Moreover, the author applied those approaches only to binary classification.

Celaya Padilla et al. (2022) used MiniGoogleNet-based TL approaches and acquired an accuracy of around 97.08%. The experiment was carried out on a single dataset and only for binary classification. Furthermore, no explainable AI approaches are used to provide enough explanations of the proposed models' predictions (Celaya-Padilla, Galván-Tejada, Gamboa-Rosales, & Galván-Tejada, 2022).

Table 1 presents an overview of some of the previously published literature on Monkeypox disease diagnosis using CNN-based approaches. From Table 1, it can be observed that most of the referenced literature does not use model interpretation techniques; therefore, it is difficult to understand whether their proposed model can identify the infected regions or not. Additionally, TL approaches are often computationally expensive, and therefore, on many occasions, it is challenging to implement them in real-world applications for real-time diagnosis. Since none of the studies provided any ideas regarding their models' time computation issues, inferring how those proposed models might perform with various datasets is also imperative.

Due to the limitations of the multiclass Monkeypox dataset, most of the literature considered binary classification, and as a result, it is hard to decode how their proposed approaches might perform on multiclass classification. In addition, the performance of the TL-based models depends on the optimizer used during the training phase. It is not often surprising that the same architectural model with the same optimizer may not demonstrate promising results both on binary and multiclass classification (Mehrotra, Ansari, Agrawal, & Anand, 2020). Therefore, a TL-based model also needs to be evaluated with various optimizers on different datasets in order to understand the model's stability as well. Most of the previous research also did not provide any clear explanation as to whether they had used generalization and regularization approaches (Eid et al., 2022; Haque, Islam, Islam and Ahsan, 2022; Islam & Shin, 2022; Sahin, Oztel, & Yolcu Oztel, 2022). Therefore, it is also not clear if their proposed model is suffering from overfitting issues or not, even though the reported accuracy is much higher (Akin et al., 2022; Haque, Ahmed, Nila, Islam et al., 2022).

From the above discussion and from Table 1, it can be inferred that very limited research has been conducted on Monkeypox disease diagnosis, where the primary concern was to develop an optimized TL-based diagnosis model. Therefore, there is a need to develop a model that is computationally efficient, provide enough model interpretation, consider generalization and regularization approaches to reduce overfitting, and finally test the model on various datasets with different TL-based approaches.

2. Motivation

The conventional ML-based model is effective for small datasets since it is more interpretable and computationally inexpensive (Ahsan, Gupta et al., 2020). Nonetheless, the conventional ML-based model performs poorly with the larger dataset (Brown, Curtis, & Goodwin, 2021). Deep Neural Network (DNN)-based techniques have already outperformed classic ML algorithms such as Random Forest (RF), SVM, and Logistic Regression for high-dimensional data, including several data types (i.e., numerical, categorical, image data) (Ahsan, Alam et al., 2020).

In our previous research we showed that it is feasible to develop AI-based diagnostic models using TL-based approaches that are effective on both small and large datasets, particularly during the early stages of the COVID-19 pandemic. Further details on this research can be found in peer-reviewed Refs. Ahsan, Ahad et al. (2021), Ahsan, Gupta et al. (2020) and Ahsan, Nazim et al. (2021). Our previous study requires a more thorough examination of model overfitting and time complexity issues. Additionally, we should have assessed the complexity of the model in terms of the number of parameters and floating operations. Furthermore, we only utilized a single interpretable technique to evaluate the interpretation of the model's predictions, which limits our ability to verify the model's interpretation with other agnostic methods for further validation.

Considering this opportunity, this study presents the Generalization and Regularization based Transfer Learning Approaches (GRA-TLA) for binary and multiclass classification. The proposed architecture has been implemented and evaluated on a range of CNN models, including VGG16, ResNet50, ResNet101, Xception, EfficientNetB0, EfficientNetB7, Nas Neural Architecture Search (Nas)-NetLarge, EfficientNetV2M, ResNet152V2, and EfficientNetV2L.

At the time of writing, very limited research study has been discovered that indicates the potential of ML approaches in diagnosing Monkeypox disease by utilizing image processing techniques.

Our technical contribution is outlined below:

1. In order to develop a Monkeypox patient detection model using image data for binary and multiclass classification, transfer learning (TL) approaches were introduced and tested on ten CNN models (VGG16, ResNet50, ResNet101, Extreme Inception (Xception), EfficientNetB0, EfficientNetB7, NasNetLarge, EfficientNetV2M, ResNet 152V2, and EfficientNetV2L) during three separate studies at the preliminary stage;
2. Implemented generalization and regularization approach to prevent overfitting and present optimal TL models;
3. Provided post-image analysis explanation using Local Interpretable Model-Agnostic Explanations (LIME) to validate our findings; and
4. Finally, the predicted outcome is visualized using Grad and Grad++ to understand the proposed model's observation and learning procedure.

The remaining paper is structured as follows: Section 3 provides a concise explanation of the experiment's methodology, followed by Section 4's results. Section 5 briefly discusses our study; Section 6 discusses study limitations and scopes, and Section 7 concludes with overall findings and further research directions.

3. Methodology

This section describes the data collection and augmentation technique, the development of the proposed DL model, the experimental setup, and the performance assessment matrices used to conduct the experiment.

3.1. Data collection

Many experts in the medical domain believe that AI systems could reduce the burden on clinical diagnosis with the outbreaks by processing image data (Ahsan, Gupta et al., 2020). During the onset of COVID-19, we observed that hospitals in China and Italy deployed AI-based and image processing-based interpreters to improve the hospitals' efficiency in handling COVID-19 patients (Ahsan, Alam et al., 2020; Ahsan, Nazim et al., 2021; Narin, Kaya, & Pamuk, 2021). However, during the preliminary stage of our experimentation, we could not find any publicly available Monkeypox dataset that hinders taking advantage of deploying an AI-based approach to diagnose and prevent the Monkeypox disease efficiently. As an effect, many researchers and practitioners cannot contribute to detecting Monkeypox disease using advanced AI techniques. Considering these limitations in this work, we collected patients' images with Monkeypox symptoms and the dataset will be regularly updated with data contributed by numerous global entities. We followed the following procedure to collect the data samples.

Table 1
Referenced literature that considered CNN-based approaches in Monkeypox disease diagnosis.

Reference	Contributions	Algorithms	Dataset	Data type	Performance evaluation	Interpretable model	Generalization/ regularization	Classification
Sahin et al. (2022)	Human Monkeypox classification	ResNet18, GoogleNet, EfficientNetbo, NasnetMobile, ShuffleNet, MobileNetv2	Monkeypox Skin Lesion Dataset (MSLD) (Ali et al., 2022)	228 images	MobileNetv2 (91.11%)	×	×	Binary
Sitaula and Shahi (2022)	Compared different pre-trained DL models	VGG, ResNet, InceptionV3, InceptionResNet, Xception, MobileNet, DenseNet, EfficientNet	Monkeypox-dataset-2022 (Ahsan, Uddin, & Luna, 2022)	1753 images	Ensemble approach (Precision: 0.85; Recall: 0.85; F1-score: 0.85; and Accuracy: 87.13%)	LIME	×	Multiclass
Akin et al. (2022)	Auxiliary decision support systems for hospitals	CNN	Monkeypox Skin Images Dataset (MSID) (Bala, 2022)	572 images	MobileNetV2 (Accuracy: 98.25%, Sensitivity: 0.96, Specificity: 1.0 and F1-Score: 0.98)	GradCAM	×	Binary
Haque, Ahmed et al. (2022)	Integrate deep TL-based methods, and convolutional block attention module (CBAM)	VGG19, Extreme Inception (Xception), DenseNet121, EfficientNetB3, and MobileNetV2	MSID (Bala, 2022)	572 images	Xception-CBAM-Dense (accuracy: 83.89%)	×	×	Binary
Islam and Shin (2022)	Blockchain-based data acquisition incorporated with federated learning	ResNet18	MSLD (Ali et al., 2022)	3192 images	Accuracy: 99.81%, Precision: 0.9981, Recall: 0.9981, and F1-score:0.9981	×	×	Binary
Celaya-Padilla et al. (2022)	Diagnostic support for Monkeypox detection	MiniGoogleNet	MSLD (Ali et al., 2022)	2067 images	Accuracy: 97.08%, Loss function: 0.1442	×	×	Binary
Irmak, Aydin, and Yağanoğlu (2022)	Monkeypox skin lesion detection	MobileNetV2, VGGNet	MSLD (Ali et al., 2022)	770 Images	MobileNetV2 (Accuracy: 91.38%, Precision: 0.90, Recall: 0.86 and F1 score:0.88)	×	×	Multiclass
Alcalá-Rmz et al. (2023)	Exanthematic disease diagnosis using Monkeypox infected images	MiniGoggleNet	MSLD	2067 images	Accuracy: 97%, Area Under Curve (AUC): 0.76	×	×	Binary

1. As there is no established shared dataset available by the authorized and designated hospital, clinic, or viable source, therefore, to establish a preliminary dataset, the Monkeypox patient data is collected from various sources such as websites, newspapers, and online portals and publicly shared samples. To do so, the google search engine is used for the initial searching procedure.
2. To develop the non-Monkeypox samples, a similar procedure is used in collecting the data sample, which contains search terms "Monkeypox" and "Normal image" (i.e., photos of both hands, legs, and faces).
3. To increase the data sample size, additional Normal images are collected manually from various participants with their consent who do not have any skin disease symptoms. A consent form is used to get approval from all the participants.

Table 2 summarizes the characteristics of the datasets developed throughout this study. While TL and conventional ML can perform well with a small number of images, deep architecture such as DL networks, CNN, Recurrent Neural Network (RNN), and Generative Adversarial Networks (GAN) require a significant amount of data samples to construct a model (Jiao, Deng, Luo, & Lu, 2020).

Although the dataset contains only 1830 samples, using the traditional ML and TL approach, it can be applied to develop a disease diagnosis model, as previously demonstrated by many studies during the onset of COVID-19 when the data samples were very limited. For instance, some study uses only 40–100 samples and develop DL models to classify COVID-19 patients (Ahsan, Gupta et al., 2020; Narin et al., 2021). However, we expect that the data size will expand over time as we will collect more data from various open-source (i.e., data available to use without privacy concerns, data from journals, and online).

Table 2
Characteristics of the dataset that has been collected in this study.

Dataset	Total sample
Monkeypox	43
Normal	33
Monkeypox augmented	587
Normal augmented	1167
Total samples	1830

3.2. Data augmentation

"Data augmentation" refers to approaches used in data analysis to expand the quantity of data by adding slightly changed copies of either existing data or newly created synthetic data derived from existing data (Shah, 2022). It has become one of the most prevalent techniques for augmenting the quantity of data required to train successful ML

Fig. 1. Sample set of images from the developed dataset including (a) Monkeypox and (b) normal images.

Table 3
Data augmentation techniques used in this study.

Generator type	Facility
Width shift	Up to 2%
Rotation range	Randomly 0°–45°
Zoom range	2%
Height shift	Up to 2%
Shear range	2%
Fill mode	Reflective
Horizontal flip	True

models. It is crucial for fields where getting high-quality data can be challenging, such as during the onset of Monkeypox. It functions as a regularizer and assists in preventing overfitting during ML model training (Sagar, 2019).

Keras image processing library such as *ImageDataGenerator* is used to augment the dataset. *ImageDataGenerator* function provides various options such as rotation, width and height shifting, and flipping. A details facility provided by *ImageDataGenerator* can be found in Tensorflow (2022). In this work following parameter is used to augment the image data as shown in Table 3. The generator type and facility types are selected randomly as suggested in Bhattiprolu (2020).

Algorithm 1 shows the pseudocode for data augmentation techniques used in this study. For instance, up to 20 iterations, the data sample is generated, and each iteration creates 16 new samples.

Algorithm 1 Pseudo-Code of Data Augmentation

```
Input: read original image samples x using OpenCV.
Resize image into 128 × 128.
Store resize image as an array inside a list.
Call Image data generator function
for n ← 1 to 20 do
    Batch size = 16
    Save to directory
    Save format as "png"
end for
End of Pseudo-Code.
```

Fig. 1 displays sample images of our datasets developed throughout this study.

Several alternatives can be used to increase the data size apart from traditional data augmentation techniques. Other feasible approaches include oversampling, GAN, and neural style transfer approaches. However, each approach has its own limitations. For instance, oversampling approaches often create samples where the major sample overlaps with minor samples (Hu & Li, 2013). GAN-based approaches require a lot of data and are often hard to train (Sarmad, Lee, & Kim, 2019). On the other hand, neural style is computationally much more expensive than traditional data augmentation techniques (Jing et al., 2019). Therefore, in this work, we have used traditional data augmentation techniques, which help to increase the data size and make it feasible to adopt without much complexity.

3.3. Convolutional neural network

Convolutional Neural Networks (CNN) are at the forefront of DL research, with applications including image recognition, object detection, and natural language processing. Though there are various CNN variants available, most of the CNN models presented in medical domains follow a basic structure that includes the Convolutional (Conv) layer, Pooling layer, Dense layer, and Softmax layer (Ahsan & Siddique, 2022). Fig. 2 shows the conventional structure of CNN models used in medical image analysis.

The Conv layer is utilized in the process of automatically extracting high-dimensional features of the images. This process allows the use of convolutional operation in order to filter the noise that is present in the initial images (refer to Section 3.4). In deep CNN, each layer uses multiple filters to extract valuable information from the images for further classification. The Pooling layer is used to reduce the dimension of the images, ultimately minimizing unnecessary parameters of the features. Max pooling and Average pooling are two of the most popular Pooling layers that are often used interchangeably with the Conv layer and vary from one CNN to another CNN architecture. The dense layer is applied so that the information from the Pooling layer may be transformed into 1D vectors, which then helps with the classification of the images in the Softmax layer. In the Softmax layer, the representative vector obtained from the Pooling layer is reshaped and mapped into a probability distribution so that it can be classified. This takes place throughout the classification process. Backpropagation, often known as BP, is eventually used to train the entirety of the CNN by combining it with gradient-based optimization techniques (Kukkar et al., 2022). After training, the CNN's parameters are tweaked and improved through optimization. An ideal CNN is obtained as a consequence of this process, and it can subsequently be used for either classification or prediction (Simonyan & Zisserman, 2014).

There are several alternative approaches to CNN that are commonly used in the field of ML, such as:

- RNN: generally used to process sequential data, such as time series or natural language (Wang, Li, Li, Sun, & Wang, 2022).
- Autoencoders: these are trained to reconstruct their inputs by learning an efficient representation of the data (He et al., 2022).
- GAN: are used to generate new data samples similar to a given training set (Goodfellow et al., 2020).
- LSTM networks: suitable for sequential data (Abbasimehr, Shabani, & Yousefi, 2020).

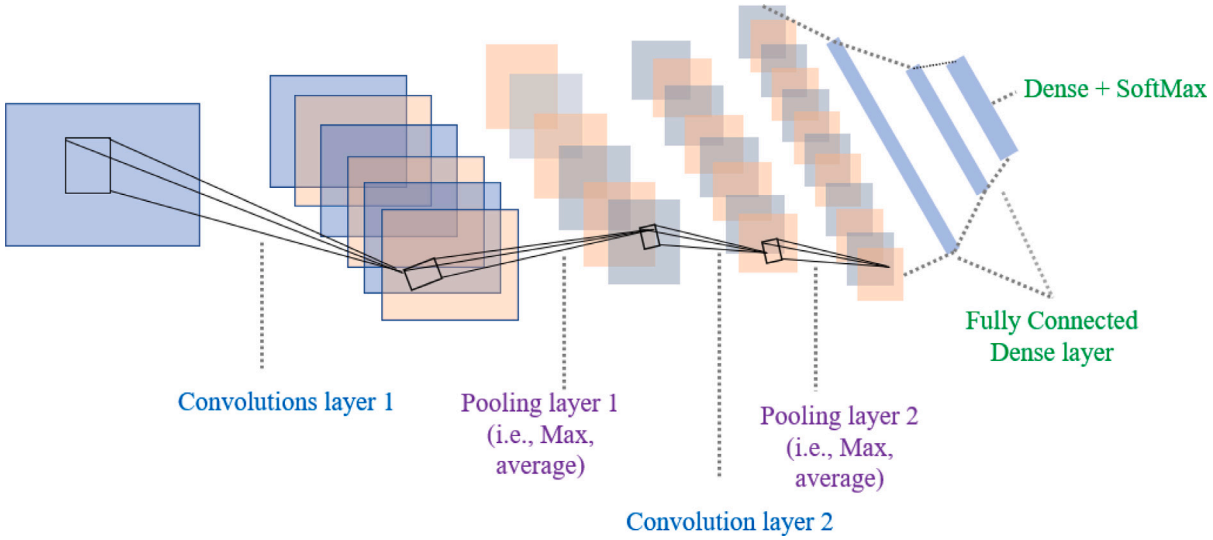

Fig. 2. The fundamental architecture of CNN for the classification of images.

- Attention mechanisms: allow a neural network to focus on certain parts of its input when processing the data (Li, Xiao, Zhang, & Fan, 2021).

However, in this work, we have used traditional CNN-based approaches as they are less complicated and demonstrate better performance than other approaches.

3.4. Feature extraction

In CNN architecture, feature extraction is one of the most important parts (Varshni, Thakral, Agarwal, Nijhawan, & Mittal, 2019). Feature extraction is a technique for reducing the dimension of vast amounts of data to analyze and improve the efficiency of DL models. The CNN's DL model comprises numerous layers that recognize and extract features from data. To obtain the input of local feature a_1, the CNN heavily relies on various weighted kernel W^l for each layer l (Popescu & Sasu, 2014) :

$$o^l = W^{l^T} a^{(l-1)} + b^l \tag{1}$$

where

o^l = Output feature map

b^l = Bias of the layer

The pooling layer is used to extract the maximum feature values as follows:

$$P_m^l = max_{(p,q)} o^l \tag{2}$$

where (m, n) denotes the side of the window

P_m^l = pooling layer

The last layer is also known as linked layer. If we consider $l-1$ is a linked layer where layer l get the input $p_1^{(l-1)}$ as a feature maps with a size of $p_2^{(l-1)} \times p_3^{(l-1)}$, then the final link layer could be define as:

$$K_i^{(l)} = f(Z_i^{(l)}) \tag{3}$$

$$Z_i^{(l)} = \sum_{j=1}^{p_1^{(l-1)}} \sum_{r=1}^{p_2^{(l-1)}} \sum_{s=1}^{p_3^{(l-1)}} w_{i,j,r,s}^{(l)} (K_j^{(l-1)})_{r,s} \tag{4}$$

where $w_{i,j,r,s}^{(l)}$ specifies the weights used to describe the ith unit's location (r, s) to the jth feature map in layer $(l-1)$.

3.5. Generalization

Let $x \in X$ be an input and $y \in Y$ be a target. Let L be a loss function. Let $R[f]$ be the expected risk of a function f, $f, R[f] = E_{(x,y\sim P(x,y))}[L(f(x), y)]$, where $P(X, Y)$ is the true distribution. Then,

Generalization gap = $R[fA(S)] - RS[fA(S)]$ where $R[fA(S)]$ = Expected risk

$RS[fA(S)]$ = Empirical risk

We typically aim to minimize the non-computable expected risk by reducing the computable empirical risk (Kawaguchi, Kaelbling, & Bengio, 2017).

3.6. Regularization

Regularization is a supplementary technique that aims at making the model generalize better, i.e., producing better results on the test set. This may include various properties of the loss function, the loss optimization algorithm, and other techniques. A classical regularizer is weight decay (Zhang, Wang, Xu and Grosse, 2018):

$$R(\omega) = \lambda \frac{1}{2} \parallel \omega \parallel_2^2 \tag{5}$$

where

R = Regularizer

λ = Weight controlling

ω = Weight

During the training phase, optimization techniques are required to develop the optimal and best model (Sutskever, Martens, Dahl, & Hinton, 2013). Therefore, we have evaluated three standard optimization algorithms: adaptive learning rate optimization algorithm (Adam) (Kingma & Ba, 2014), stochastic gradient descent (Sgd) (Zhang et al., 2018), and root-mean-square propagation (Rmsprop) (Dauphin, De Vries, & Bengio, 2015).

Adam uses exponential moving averages to calculate the average of the gradients and the square gradients, using the gradients obtained from the current mini-batch (Zhang, 2018):

$$m_t = \beta_1 m(t-1) + (1-\beta_1) g_t \tag{6}$$

$$v_t = \beta_2 v(t-1) + (1-\beta_2) g_t^2 \tag{7}$$

where m and v are moving averages, g is the gradient of current mini-batch, and β is the hyper-parameter of the algorithm.

Sgd is an iterative procedure that identifies the minimal function to get local minima. In this procedure, the next point is determined by a gradient at the present position, scaled, and then subtracted from the current position. The process can be expressed as follows (Amari, 1993):

$$P_{n+1} = P_n - \eta \nabla f(P_n) \tag{8}$$

where

η = Learning rate
P_{n+1} = Next point
P_n = Current point

The smaller the learning rate, the longer it takes for Sgd to converge or reach maximum iteration without finding the optimal points (Liu, Papailiopoulos, & Achlioptas, 2020).

In the DL model, *Rmsprop* is another optimization approach utilized frequently. The algorithm is designed to maintain the moving average of the squared gradient for each weight. The gradient is then divided by the mean's square root. The procedure can be stated as follows (Dauphin et al., 2015):

$$E[g^2]_t = \beta E[g^2]_{t-1} + (1-\beta)(\frac{\delta C}{\delta W})^2 \tag{9}$$

$$W_t = W_{t-1} - \frac{\eta}{\sqrt{E[g^2]_t}} \frac{\delta C}{\delta W} \tag{10}$$

where

$E[g]$ = Moving average of squared gradients
$\frac{\delta C}{\delta W}$ = Gradient of the cost function with respect to the weight
η = Learning rate
β = Moving average parameter

3.7. VGG16

VGG stands for Visual Geometry Group, which proposed two deep CNN models in their work. The models are 16- and 19-layer depth and are named VGG16 and VGG19, respectively. They trained their proposed models using one million samples collected from the ImageNet dataset. The VGG16 model initially takes 224 × 224 size images as input. Images are initially passed through several Conv layers containing filters ranging from 64 to 512. Max pool (2 × 2 filter) is used as the pooling layer, whereas Rectified linear unit (ReLu) is used as an activation function with a stride size of 2. In the dense layer, the Pooling layer is converted to a 1D vector. The final step in the process involves applying the Softmax activation function to the output layer in order to categorize samples into one of 1000 categories (Simonyan & Zisserman, 2014).

3.8. ResNet50

ResNet50 is an implementation of the ResNet model. It has 48 convolution layers, one layer each of max pooling, average pooling, and regular pooling. In the first layer of the architecture of ResNet50, there is a convolution with a kernel size of 7 × 7 and 64 kernels, each of which has a stride size of 2. Following this is a max-pooling layer with a stride size of 2. After that, nine convolutional layers with three types of kernel filters, 64, 64, and 256 for each layer, are applied. The last nine levels each include 512, 512, and 2048 kernel filters. Then, 1000 nodes are added to an FC layer, including an average pooling layer. The Softmax function is used as the output layer's activation function (Akiba, Suzuki, & Fukuda, 2017; He, Zhang, Ren, & Sun, 2016a).

3.9. ResNet101

ResNet101's model architecture is nearly comparable to ResNet50's. The network accepts the 224 × 224 resolution image size. The key distinction between ResNet50 and ResNet101 is that the ResNet101 model has an additional three-block layer in the fourth block, which comprises 256, 256, and 1024 filters (He et al., 2016a).

3.10. Xception

The Xception model is based on the concept of the Inception model. The model is split into three main components: entry, center, and exit. The model is constructed with separable convolutional layers, which substantially reduces the number of trainable parameters. On the ImageNet dataset, the model achieved roughly 94.5% accuracy for the top five object classifications (Alam et al., 2022; Nguyen et al., 2022).

3.11. EfficientNetB0

EfficientNetB0 is a deep CNN models that uses scaling methods to scale all the dimensions uniformly. The base of EfficientNetB0 is based on the inverted residual blocks of MobileNetV2. The model uses squeeze and excitation methods inside the blocks and contains around 237 layers (Sharma, Vijayeendra, Gopakumar, Patni, & Bhat, 2022). This model's performance on the CIFAR-100 dataset is approximately 91.7%, making it one of the most common transfer learning techniques employed by academics (Alam et al., 2022).

3.12. EfficientNetB7

EfficientNetB7 is one of the eight versions (0–7) of the EfficientNet model that was created utilizing compound scaling methods. The three key parameters of EfficientNet are alpha, beta, and gamma, and each EfficientNet is constructed with different values for these parameters. The model achieved approximately 84.3% accuracy on the ImageNet dataset (Tan & Le, 2019).

3.13. NasNetLarge

NasNetlarge is a Google-introduced framework for identifying the most effective CNN architecture for a given problem set via reinforcement learning strategies. The goal was to find the optimal setting of parameters (such as filter size, output channel, stride, number of layers, etc.) within the available search space. Using the ImageNet database, the model was roughly 82.5% accurate (Cordoş, Mihailă, Faragó, & Hintea, 2021; Zhang & Davison, 2020).

3.14. EfficientNetV2M

The EfficientNetV2M model is an improved and more time-efficient version of the initial EfficientNet model. The model can be subdivided into its seven component sections, which are then layered with their respective modules. On average, the model was about 85.3% accurate across the ImageNet dataset (Tan & Le, 2021).

3.15. ResNet152V2

ResNet152v2 is the modified version of Residual Network (ResNet). The model contains more than a thousand convolutional layers. ResNet itself also contains a huge number of layers. The major difference between the ResNetV2 and V1 models is that the V2 model uses batch normalization before implementing the weight layer. On the ImageNet dataset, the models achieved around 76.6% accuracy (top 1-accuracy) (Beyer, Hénaff, Kolesnikov, Zhai, & Oord, 2020; He, Zhang, Ren, & Sun, 2016b).

3.16. EfficientNetV2L

The EfficientNet model versions, including EfficientNetV2L, are all built on the same basic CNN-based framework. This model uses a smaller kernel size (3 × 3) than the original EfficientNet, which helps minimize the model's memory access cost and parameters. On ImageNet, the model was accurate about 85.7% of the time (Tan & Le, 2021).

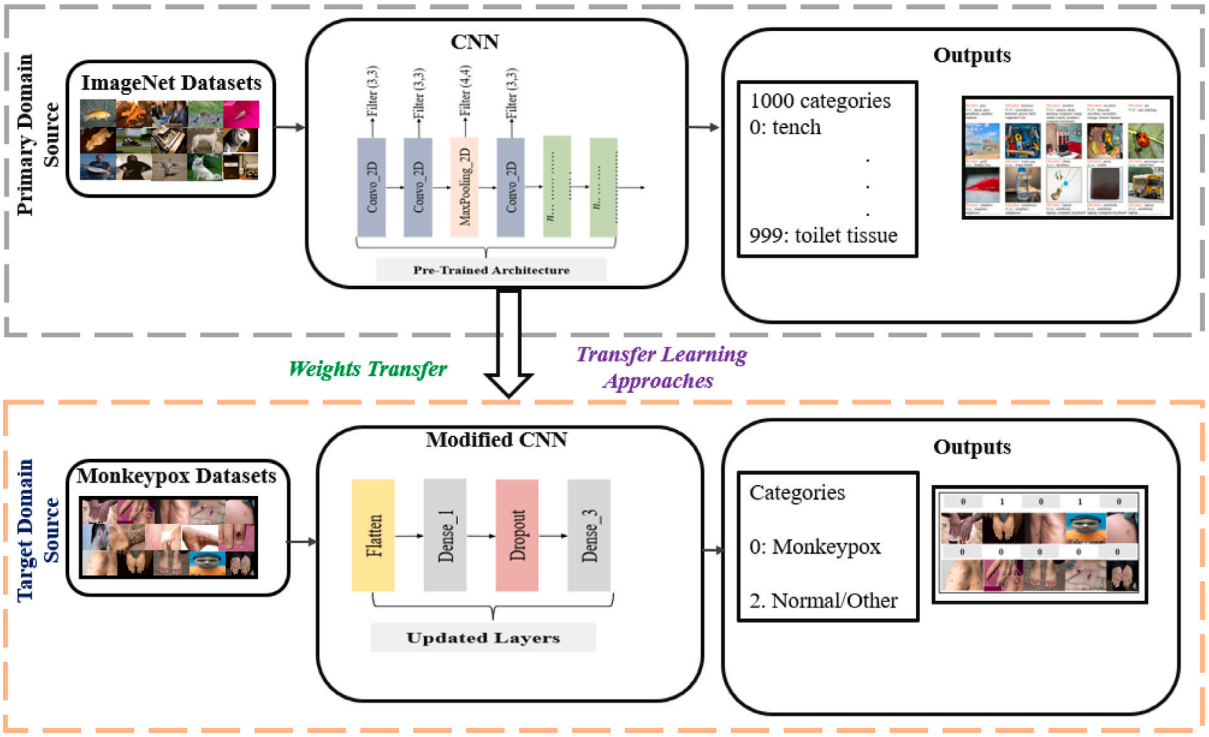

Fig. 3. The framework of the proposed modified method.

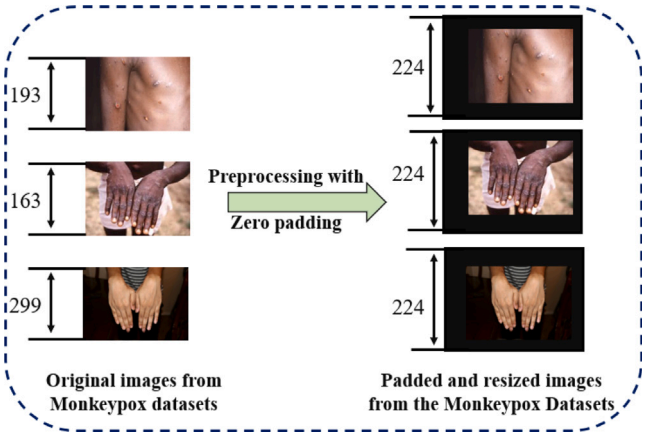

Fig. 4. Preprocessing with zero padding proposed in this work.

3.17. Proposed model

The proposed modified CNN is developed by taking advantage of TL approaches. Fig. 3 illustrates the two components that comprise the proposed method, which is as follows: the preprocessing of the Monkeypox images and the detection of Monkeypox patients based on the proposed CNN.

Recently Graph Neural Networks (GNN) and Capsule Neural Networks (CapsNet) have been used as an alternative to CNN-based approaches. However, many researchers raised concerns about using GNN or CapsNet for image-based model development (Peer, Stabinger, & Rodriguez-Sanchez, 2021). They are still ongoing research, whereas CNN-based approaches are specifically designed to perform better on image-based data. One potential drawback of the GNN-based approach is excessive hardware dependencies, which makes this approach much more computationally expensive than traditional CNN-based approaches. GNN can only operate on a limited number of points. In addition, GNN-based techniques are not resilient against noisy data, which is an additional challenge when applying them to an image-based dataset with complex data points (Anil, 2021).

3.17.1. Preprocessing of the Monkeypox images

All-Region of Interest (ROI) patches from the Monkeypox image datasets are firstly preprocessed to the dimension of 224 × 224 × 3 using the zero-padding method. Fig. 4 shows the flowchart of preprocessing procedure for the Monkeypox image.

In the meantime, grayscale images are read in and transformed into RGB images using OpenCV functions so that the image can be fitted to the input layer of the proposed CNN. Apart from reducing the computational time complexity, no further preprocessing steps are employed for the proposed models. Algorithm 2 shows the pseudocode for the preprocessing of the proposed models.

Algorithm 2 Pseudo-Code of data preprocessing

```
Input: Monkeypox or Normal images
Output: Preprocessed resize image
for x ← 1 to n do
  Label= Split label from the sample and store as matrix
  for image ← 1 to x do
    Call Preprocessimage
    R_i ← Read image
    R_s ← Resize image in to 224 × 224
    C_i ← Convert image in to RGB
    D ← Append image data
  end for
  C_d ← convert data into matrix
  C_l ← Convert label into categories
end for
End of Pseudo-Code.
```

3.17.2. Proposed CNN architecture

The core model consists of three essential elements: pre-trained architecture, an updated layer, and a prediction class (partially adapted from Ahsan, Gupta et al. (2020)). The ImageNet dataset is chosen as the primary domain source for the pre-trained models. ImageNet is one of the largest visual databases, with over 14 million images categorized into one thousand classifications. Most existing state-of-the-art algorithms are trained or evaluated on the ImageNet dataset. In addition, a pre-trained model on such an extensive dataset enables the model to capture essential features, which facilitates the adoption of DL-based models in various areas (Studer et al., 2019).

The pre-trained architecture is used to identify high-dimensional features and is further added to the updated, modified layer. Fig. 3 illustrates the proposed CNN models. As shown in Figure, after the initial input layer (consider 224 × 224 images only), two convolutional layer (containing a 3 × 3 filter) is added, followed by a Max Pooling layer, followed by another two convolutional and one Max Pooling layer until it reaches to the modified layer sections. The modified layer Flattened the architecture, followed by the three dense and one dropout layers.

During our experiment, for the binary classification, the following loss function is calculated:

$$L = -(\frac{1}{N}) \times \sum_{i=1}^{N} [y_i \log(p_i \log) + (1 - y_i) log(1 - p_i)] \quad (11)$$

where N is the number of samples, y_i is the ground truth label for the ith sample (either 0 or 1), and p_i is the predicted probability of the ith sample belonging to the positive class.

Whereas for the multi-class classification, the following loss function is considered:

$$L = -(\frac{1}{N}) \times \sum_{i=1}^{N} \sum_{j=1}^{K} y_{ij} \times \log(p_{ij}) \quad (12)$$

where N is the number of samples, K is the number of classes, y_{ij} is the ground truth label for the ith sample in the jth class, and p_{ij} is the predicted probability of the ith sample belonging to the jth class.

3.18. LIME as explainable AI

LIME is one of the powerful tools that can help to analyze the model's true prediction and offer the opportunity to understand the blackbox behind any CNN model's final predictions (Ribeiro, Singh, & Guestrin, 2016a). According to Ribeiro et al. (2016), for an interpretability model to be locally faithful, it must demonstrate how the model behaves in the predicted sample's neighborhood (Ribeiro, Singh, & Guestrin, 2016b). Using LIME, the following formula is used to develop the optimization model:

$$\xi(x) = \underset{g \in G}{\operatorname{argmin}} \mathcal{L}(f, g, \pi_x) + \omega(g) \quad (13)$$

where G is the instance of explanation models, L is the loss function that ensures that g is fitted to f in a local neighborhood around x. x is identified by the weighted kernel π_x, which is an exponential kernel of the distance function. In this study, we have used Euclidian distance, with a fixed bandwidth σ, which can be further explained as $\pi_x = exp(-|x^* - x|/\sigma)$. Note that $\Sigma(g)$ is a penalty term, and in this study, we have used $l1$ as the regularization loss, which helps to prevent overfitting and reduces the complexity of g (Ghalebikesabi, 2022)

LIME's impressive performance in describing the complexities of image classification has led to its extensive application in recent years (Cian, van Gemert, & Lengyel, 2020). In the case of image classification, LIME uses superpixel. When an image is over-segmented, superpixels are produced. Superpixels stores much data and help to identify essential features of the images during the primary prediction (Ahsan, Gupta et al., 2020). Table 4 represents the LIME parameters that have been used in this study to calculate the superpixel values. Different superpixel sizes can affect the accuracy and interpretability of the LIME explanations. Larger superpixels are usually more accurate in approximating the behavior of the model, as they cover a larger area of the input space and are less affected by noise (Garreau & Mardaoui, 2021).

Table 4
Parameter used to identify superpixels.

Function	Value	Optimal value
Maximum distance	100, 150, 200	200
Kernel size	2, 4, 6	4
Ratio	0.2, 0.3, 0.4	0.2

Therefore, the choice of superpixel size needs to be determined based on the research objectives and goals. Therefore, in our research, we used different superpixel values, and the best interpretable value was used for the model's final interpretations.

Note that the parameters are proven to be useful in many image prediction analyses, as referred by many existing literatures (Ahsan, Gupta et al., 2020; Pan et al., 2020).

Apart from LIME, other alternative model agnostic approaches are available, and SHapley Additive exPlanations (SHAP) is one of them Lundberg and Lee (2017). However, our choice of LIME over SHAP was influenced by some of the following factors (Okte & Al-Qadi, 2021; Ribeiro et al., 2016a):

- LIME is generally easier to implement and can be applied to any black-box model. At the same time, SHAP requires more computational resources and may only be suitable for some large or complex models.
- LIME explanations are often more concise and easier to understand, as they only focus on the essential features for a given prediction. SHAP explanations can be more detailed but more challenging to interpret due to the more significant number of features and combinations considered.

Ultimately, the choice between LIME and SHAP will depend on the specific goals and constraints of the model. Both methods have their own strengths and limitations, and it may be worthwhile to compare their results and consider using both approaches in combination.

3.19. Experiment setup

The experiment was conducted using a traditional laptop within the specification of Windows 10, 16 GB RAM, and Intel Core I7. The overall experiment was run five times, and the final result is presented by averaging all the five computational outcomes.

Table 5 provides a summary of the dataset utilized for this study. In this work, we have considered three separate studies. In Studies One and Two, binary classification was investigated, whereas multiclass classification was examined in Study Three. For Studies One and Two, we analyzed our developed "Monkeypox2022" dataset, whereas for Study Three, data is obtained from the Kaggle dataset. We used 80% of the sample data for training and the remaining 20% for testing the model, which is standard in ML domains (Menzies, Greenwald, & Frank, 2006; Mohanty, Hughes, & Salathé, 2016; Stolfo, Fan, Lee, Prodromidis, & Chan, 2000).

In this work, early stopping is used to avoid overfitting and data leakage and to provide the actual model's performance. It is one of the most commonly used regularization techniques in DL. Early stopping helps to evaluate the validation loss less frequently and saves the trained model periodically (Corneanu, Madadi, Escalera, & Martinez, 2020).

Table 5
Assignment of data employed to train and test the proposed modified deep transfer learning models.

Study	Label	Train set	Test set	Total	Classification
One	Monkeypox	34	9	43	Binary
	Normal	26	7	33	
	Total	60	16	76	
Two	Monkeypox	470	117	587	Binary
	Others	933	234	1167	
	Total	1403	351	1754	
Three	Chickenpox	80	20	100	Multiclass
	Measles	64	16	80	
	Monkeypox	211	53	264	
	Normal	172	43	215	
	Total	527	132	659	

3.20. Hyperparameters

The batch size, number of epochs, and learning rate are initially examined during parameter tuning to maximize the performance of the proposed model. The following experiment parameters are selected at the beginning of Study One (inspired by Ahsan, Ahad et al. (2021) and Bergstra and Bengio (2012)):

Batch size = [5, 10, 15, 20]
Learning rate = [0.1, 0.01, 0.001]
Number of Epochs = [30, 35, 40, 45, 50]

Using the grid search method following parameters are identified as the most optimal ones:

Batch size = 10
Learning rate = 0.001
Number of Epochs = 30

For Study Two following parameters are used to develop the optimal model:

Batch size = [30, 40, 45, 50]
Learning rate = [0.1, 0.01, 0.001]
Number of Epochs = [30, 40, 50, 70]

And the best result was achieved with:

Batch size = 45
Learning rate = 0.01
Number of Epochs = 30

For Study Three following parameters are used to develop the optimal model:

Batch size = [30, 40, 50]
Learning rate = [0.1, 0.01, 0.001]
Number of Epochs = [30, 40, 50]

And the best result was achieved with:

Batch size = 40
Learning rate = 0.1
Number of Epochs = 30

Table 6 summarizes some of the combinations of multiple iterations of hyperparameters that have been utilized to find the optimal parameters for three studies. The optimal combination of hyperparameters is chosen once the highest average accuracy is achieved.

3.21. Performance evaluation

The overall experimental outcome is measured and presented using the most widely used statistical approaches such as accuracy, precision, recall, F1-score, sensitivity, and specificity. Due to the limited samples, the overall statistical results are represented with a 95% confidence interval followed by previously reported literature that also used a small dataset (Narin et al., 2021; Wang et al., 2020). In our dataset, Monkeypox might be classified as true positive (T_p) or true negative (T_n) if individuals have distinguished accurately, and it might be classified into false positive (F_p) or false negative (F_n) if misdiagnosed. The designated statistical metrics are explained in detail below.

Accuracy: The accuracy is the overall number of successfully identified instances across all cases. Using the following formulas, accuracy can be determined:

$$Accuracy = \frac{T_p + T_n}{T_p + T_n + F_p + F_n} \tag{14}$$

Precision: Precision is assessed as the ratio of accurately predicted positive outcomes out of all expected positive outcomes.

$$Precision = \frac{T_p}{T_p + F_p} \tag{15}$$

Recall: Recall refers to the ratio of relevant outcomes that the algorithm accurately identifies.

$$Recall = \frac{T_p}{T_n + F_p} \tag{16}$$

Sensitivity: Sensitivity refers to the only accurate positive metric relative to the total number of occurrences and can be measured as follows:

$$Sensitivity = \frac{T_p}{T_p + F_n} \tag{17}$$

Specificity: It identifies the number of accurately identified and calculated true negatives and can be found using the following formula:

$$Specificity = \frac{T_n}{T_n + F_p} \tag{18}$$

F1-score: The F1-score is the harmonic mean of precision and recall. The maximum possible F score is 1, which indicates perfect recall and precision.

$$F1 - score = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{19}$$

Computational complexity: To understand the model's complexity, we have measured the Floating-Point Operations per Second (FLOPs). FLOPs is the number of floating-point operations that a computing entity can accomplish in one second. In a CNN model, the total number of floating-point operations required for a single forward pass is measured in FLOPs (Jin & Finkel, 2020). Lets consider a CNN models where

F_p = FLOPS
c = Convolutional layer
F_c = Fully connected layers
P_l = Pooling layers
I_s = Input size
K_n = Number of kernel
K_s = Kernel shape
O_s = Output Shape
H = Height of the image
D = Depth of the image
w = Width of the image

Then F_p can be calculated as follows Hobbhahn (2021):

$$F_p = c + 2 \times K_n \times K_s \times O_s \tag{20}$$

$$F_p = F_c + 2 \times I_s \times O_s \tag{21}$$

$$F_p = P_l + H \times D \times W \tag{22}$$

Table 6
Optimal parameter searching using grid search methods incorporated with various parameters. M_{ts}–mean test score, B_s–batch size, E–epochs, L_r–learning rate.

Study one				Study two				Study three			
M_{ts}	B_s	E	L_r	M_s	B_s	E	L_r	M_{ts}	B_s	E	L_r
0.7333 (0.0623)	5	30	0.001	0.6072 (0.0099)	30	30	0.001	0.6337 (0.0316)	30	30	0.001
0.7500 (0.0408)	5	30	0.01	0.5865 (0.0195)	30	30	0.01	0.5920 (0.0107)	30	30	0.01
0.7666 (0.0471)	5	30	0.1	0.6001 (0.0215)	30	30	0.1	0.6034 (0.0089)	30	30	0.1
0.7500 (0.0408)	5	45	0.01	0.5773 (0.0249)	30	40	0.001	**0.6394 (0.0149)**	**40**	**30**	**0.1**
0.7833 (0.0235)	5	50	0.001	0.5744 (0.0087)	30	40	0.01	0.6261 (0.0153)	50	50	0.001
0.8333 (0.0235)	**10**	**30**	**0.001**	0.5659 (0.0216)	30	40	0.1	0.6109 (0.0104)	50	50	0.01
0.7666 (0.0235)	10	30	0.01	0.6022 (0.0128)	30	50	0.001	0.6110 (0.0023)	50	50	0.1
0.7333 (0.0471)	15	30	0.1	**0.6222 (0.0158)**	**45**	**30**	**0.01**	0.5862 (0.0246)	30	50	0.1

Table 7
Model's performance using Adam optimizer for Study One, along with confidence interval (α = 0.05). A_c–accuracy, P_r–precision, R_e–recall, F_s–F1-score, S_n–sensitivity, S_p–specificity.

Algorithm	Training set						Testing set					
	A_c	P_r	R_e	F_s	S_n	S_p	A_c	P_r	R_e	F_s	S_n	S_p
VGG16	98% ± 1.6	0.98 ± 0.016	0.98 ± 0.016	0.98 ± 0.016	1	0.96 ± 0.023	88% ± 7.59	0.9 ± 0.069	0.88 ± 0.076	0.87 ± 0.079	1	0.71 ± 0.118
ResNet50	57% ± 7.420	0.32 ± 0.093	0.57 ± 0.074	0.41 ± 0.087	1	0	56% ± 14.53	0.32 ± 0.181	0.56 ± 0.145	0.4 ± 0.170	1	0
ResNet101	57% ± 7.420	0.32 ± 0.093	0.57 ± 0.074	0.41 ± 0.087	1	0	56% ± 14.53	0.32 ± 0.181	0.56 ± 0.145	0.4 ± 0.170	1	0
Xception	67% ± 6.50	0.71 ± 0.061	0.67 ± 0.065	0.62 ± 0.070	0.94 ± 0.028	0.3077 ± 0.094	69% ± 12.2	0.8 ± 0.098	0.69 ± 0.122	0.63 ± 0.133	1	0.2857 ± 0.185
EfficientNetB0	57% ± 7.420	0.32 ± 0.093	0.57 ± 0.074	0.41 ± 0.087	1	0	56% ± 14.53	0.32 ± 0.181	0.56 ± 0.145	0.4 ± 0.170	1	0
EfficientNetB7	57% ± 7.420	0.32 ± 0.093	0.57 ± 0.074	0.41 ± 0.087	1	0	56% ± 14.53	0.32 ± 0.181	0.56 ± 0.145	0.4 ± 0.170	1	0
NasNetLarge	**100%**	**1**	**1**	**1**	**1**	**1**	**88 ± 7.759**	**0.9 ± 0.069**	**0.88 ± 0.076**	**0.87 ± 0.079**	**1**	**0.7143 ± 0.117**
EfficientNetV2M	57% ± 7.420	0.32 ± 0.093	0.57 ± 0.074	0.41 ± 0.087	1	0	56% ± 14.53	0.32 ± 0.181	0.56 ± 0.145	0.4 ± 0.170	1	0
ResNet152V2	**100%**	**1**	**1**	**1**	**1**	**1**	**88% ± 7.59**	**0.9 ± 0.069**	**0.88 ± 0.076**	**0.87 ± 0.079**	**1**	**0.7143 ± 0.117**
EfficientNetV2L	57% ± 7.420	0.32 ± 0.093	0.57 ± 0.074	0.41 ± 0.087	1	0	56% ± 14.53	0.32 ± 0.181	0.56 ± 0.145	0.4 ± 0.170	1	0

Table 8
Model's performance using Sgd optimizer for Study One, along with confidence interval (α = 0.05). A_c–accuracy, P_r–precision, R_e–recall, F_s–F1-score, S_n–sensitivity, S_p–specificity.

Algorithm	Training set						Testing set					
	A_c	P_r	R_e	F_s	S_n	S_p	A_c	P_r	R_e	F_s	S_n	S_p
VGG16	98%± 1.6	0.98 ± 0.016	0.98 ± 0.016	0.98 ± 0.016	1	0.96 ± 0.023	88% ± 7.59	0.9 ± 0.069	0.88 ± 0.076	0.87 ± 0.079	1	0.71 ± 0.118
ResNet50	57% ± 7.420	0.32 ± 0.093	0.57 ± 0.074	0.41 ± 0.087	1	0	56% ± 14.53	0.32 ± 0.181	0.56 ± 0.145	0.4 ± 0.170	1	0
ResNet101	60% ± 7.157	0.79 ± 0.052	0.54 ± 0.077	0.44 ± 0.085	1	0.076 ± 0.109	56% ± 14.536	0.28 ± 0.186	0.5 ± 0.155	0.36 ± 0.175	1	0
Xception	**100%**	**1**	**1**	**1**	**1**	**1**	**88% ± 7.59**	**0.90 ± 0.069**	**0.88 ± 0.076**	**0.87 ± 0.079**	**1**	**0.71 ± 0.118**
EfficientNetB0	57% ± 7.420	0.32 ± 0.093	0.57 ± 0.074	0.41 ± 0.087	1	0	56% ± 14.53	0.32 ± 0.181	0.56 ± 0.145	0.4 ± 0.170	1	0
EfficientNetB7	57% ± 7.420	0.32 ± 0.093	0.57 ± 0.074	0.41 ± 0.087	1	0	56% ± 14.53	0.32 ± 0.181	0.56 ± 0.145	0.4 ± 0.170	1	0
NasNetLarge	100%	1	1	1	1	1	81% ± 9.56	0.81 ± 0.096	0.81 ± 0.096	0.81 ± 0.096	0.88 ± 0.076	0.7143 ± 0.117
EfficientNetV2M	57% ± 7.420	0.32 ± 0.093	0.57 ± 0.074	0.41 ± 0.087	1	0	56% ± 14.53	0.32 ± 0.181	0.56 ± 0.145	0.4 ± 0.170	1	0
ResNet152V2	100%	1	1	1	1	1	81%± 9.55	0.86 ± 0.082	0.81 ± 0.096	0.8 ± 0.098	1	0.5714 ± 0.143
EfficientNetV2L	50% ± 8.001	0.60 ± 0.072	0.50 ± 0.080	0.40 ± 0.088	0.21 ± 0.101	0.88 ± 0.039	50% ± 15.49	0.58 ± 0.142	0.5 ± 0.155	0.45 ± 0.163	0.22 ± 0.194	0.85 ± 0.085

4. Results

During this experiment, the statistical performance was measured in terms of accuracy, precision, recall, F1-score, sensitivity, and specificity for ten different DL—VGG16, ResNet50, ResNet101, Xception, EfficientNetB0, EfficientNetB7, NasNetLarge, EfficientNetV2M, ResNet152 -V2, EfficientNetV2L—approaches in three separate studies with three optimizers, "Adam", "Sgd", and "Rmsprop", using Eqs. (14)–(19) for both the training and testing sets. Here, the best performance is highlighted using bold font during each study.

4.1. Study one

The performance of ten DL-based models on the training and test sets with Adam's optimizer is summarized in Table 7. NasNetLarge and ResNet152V2 displayed the best performance among all models, obtaining 100% accuracy on the training set and 88% ± 7.59% accuracy on the testing set, as shown in the Table 7. The models trained with VGG16 had the second-best performance, achieving approximately 98% ± 1.6% and 88% ± 7.59% accuracy on the training and testing sets, respectively. Apart from these three models, the performance of the other seven DL-based models with the Adam optimizer was unsatisfactory, as shown in Table 7. The best experimental outcome for NasNetLarge, ResNet152V2, and VGG16 remains constant for other statistical measures such as precision, recall, F1-score, sensitivity, and specificity.

For Study One, Xception models trained with Sgd displayed the best performance, while EfficientNetV2L demonstrated the worst performance across all measures, as shown in Table 8.

Table 9 displays the performance of each model with the Rmsprop optimizer on both the training and testing sets, as well as confidence intervals of 95%. Across all measures, the Xception model shows the highest performance, while ResNet50, EfficientNetB0, EfficientNetB7, EfficientNetV2M, and EfficientNetV2L exhibit the worst performance.

4.2. Study two

The performance of ten DL-based models for the second dataset is presented in Study Two for all three optimizers. Table 10 shows that the Xception model performs better with the Adam optimizer compared to other DL-based models used in this study. At the same time, ResNet50, ResNet101, EfficientNetB0, EfficientNetB7, EfficientNetV2M, and EfficientNetV2L displayed the worst performance considering all statistical measurements.

For Study Two, Xception models trained with Sgd displayed the best performance, while ResNet50, ResNet101, EfficientNetB0, EfficientNetB7, EfficientNetV2M, and EfficientNetV2L demonstrated the worst performance among all of the models, as shown in Table 11 (see Table 12).

Table 13 displays the performance of each model with the Rmsprop optimizer on both the training and testing sets, as well as confidence intervals of 95%. Across all measures, the Xception model shows the highest performance, while ResNet50, ResNet101 EfficientNetB0, EfficientNetB7, EfficientNetV2M, and EfficientNetV2L exhibit the worst performance.

4.3. Study three

Table 14 displays the performance of each model with the Adam optimizer on both the training and testing sets for Study Three. From the table, it can be inferred that the best performance was observed for

Table 9
Model's performance using Rmsprop optimizer for Study One, along with confidence interval (α = 0.05). A_c–accuracy, P_r–precision, R_e–recall, F_s–F1-score, S_n–sensitivity, S_p–specificity.

Algorithm	Training set						Testing set					
	A_c	P_r	R_e	F_s	S_n	S_p	A_c	P_r	R_e	F_s	S_n	S_p
VGG16	100%	1	1	1	1	1	88% ± 7.591	0.90 ± 0.069	0.88 ± 0.076	0.87 ± 0.079	1	0.7143 ± 0.117
ResNet50	57% ± 7.420	0.32 ± 0.093	0.57 ± 0.074	0.41 ± 0.087	1	0	56% ± 14.53	0.32 ± 0.181	0.56 ± 0.145	0.4 ± 0.170	1	0
ResNet101	62% ± 6.976	0.80 ± 0.051	0.56 ± 0.075	0.48 ± 0.082	1	0.12 ± 0.106	56% ± 14.536	0.28 ± 0.186	0.5 ± 0.155	0.36 ± 0.175	1	0
Xception	**100%**	**1**	**1**	**1**	**1**	**1**	**94%± 5.36**	**0.94 ± 0.054**	**0.94 ± 0.054**	**0.94 ± 0.054**	**1**	**0.8571 ± 0.083**
EfficientNetB0	57% ± 7.420	0.32 ± 0.093	0.57 ± 0.074	0.41 ± 0.087	1	0	56% ± 14.53	0.32 ± 0.181	0.56 ± 0.145	0.4 ± 0.170	1	0
EfficientNetB7	57% ± 7.420	0.32 ± 0.093	0.57 ± 0.074	0.41 ± 0.087	1	0	56% ± 14.53	0.32 ± 0.181	0.56 ± 0.145	0.4 ± 0.170	1	0
NasNetLarge	100%	1	1	1	1	1	81% ± 9.56	0.81 ± 0.096	0.81 ± 0.096	0.81 ± 0.096	0.88 ± 0.076	0.7143 ± 0.117
EfficientNetV2M	57% ± 7.420	0.32 ± 0.093	0.57 ± 0.074	0.41 ± 0.087	1	0	56% ± 14.53	0.32 ± 0.181	0.56 ± 0.145	0.4 ± 0.170	1	0
ResNet152V2	100%	1	1	1	1	1	81%± 9.55	0.86 ± 0.082	0.81 ± 0.096	0.8 ± 0.098	1	0.5714 ± 0.143
EfficientNetV2L	57% ± 7.420	0.32 ± 0.093	0.57 ± 0.074	0.41 ± 0.087	1	0	56% ± 14.53	0.32 ± 0.181	0.56 ± 0.145	0.4 ± 0.170	1	0

Table 10
Model's performance using Adam optimizer for Study Two, along with confidence interval (α = 0.05). A_c–accuracy, P_r–precision, R_e–recall, F_s–F1-score, S_n–sensitivity, S_p–specificity.

Algorithm	Training set						Testing set					
	A_c	P_r	R_e	F_s	S_n	S_p	A_c	P_r	R_e	F_s	S_n	S_p
VGG16	84% ± 0.93	0.84 ± 0.009	0.84 ± 0.009	0.84 ± 0.009	0.73 ± 0.012	0.89 ± 0.008	76% ± 2.3	0.76 ± 0.023	0.76 ± 0.023	0.76 ± 0.023	0.55 ± 0.031	0.86 ± 0.018
ResNet50	67% ± 1.344	0.44 ± 0.018	0.67 ± 0.013	0.53 ± 0.016	0	1	67% ± 2.7	0.44 ± 0.035	0.67 ± 0.027	0.53 ± 0.032	0	1
ResNet101	67% ± 1.344	0.44 ± 0.018	0.67 ± 0.013	0.53 ± 0.016	0	1	67% ± 2.7	0.44 ± 0.035	0.67 ± 0.027	0.53 ± 0.032	0	1
Xception	**87% ± 0.843**	**0.88 ± 0.008**	**0.87 ± 0.008**	**0.86 ± 0.009**	**0.65 ± 0.014**	**0.97 ± 0.004**	**80% ± 2.1**	**0.80 ± 0.021**	**0.80 ± 0.021**	**0.79 ± 0.021**	**0.53 ± 0.032**	**0.93 ± 0.012**
EfficientNetB0	67% ± 1.344	0.44 ± 0.018	0.67 ± 0.013	0.53 ± 0.016	0	1	67% ± 2.7	0.44 ± 0.035	0.67 ± 0.027	0.53 ± 0.032	0	1
EfficientNetB7	67% ± 1.344	0.44 ± 0.018	0.67 ± 0.013	0.53 ± 0.016	0	1	67% ± 2.7	0.44 ± 0.035	0.67 ± 0.027	0.53 ± 0.032	0	1
NasNetLarge	100%	1	1	1	0.99 ± 0.002	0.99 ± 0.002	67% ± 2.7	0.78 ± 0.022	0.67 ± 0.027	0.55 ± 0.031	0.019 ± 0.046	1
EfficientNetV2M	67% ± 1.34	0.44 ± 0.018	0.67 ± 0.013	0.53 ± 0.016	0	1	67% ± 2.7	0.78 ± 0.022	0.67 ± 0.027	0.55 ± 0.031	0.019 ± 0.046	1
ResNet15V2	75% ± 1.17	0.82 ± 0.010	0.75 ± 0.012	0.7 ± 0.013	0.26 ± 0.020	0.99 ± 0.002	75% ± 2.3	0.82 ± 0.020	0.75 ± 0.023	0.7 ± 0.026	0.26 ± 0.040	0.99 ± 0.005
EfficientNetV2L	67% ± 1.34	0.44 ± 0.018	0.67 ± 0.013	0.53 ± 0.016	0	1	67% ± 2.7	0.44 ± 0.035	0.67 ± 0.027	0.53 ± 0.032	0	1

Table 11
Model's performance using Sgd optimizer for Study Two, along with confidence interval (α = 0.05). A_c–accuracy, P_r–precision, R_e–recall, F_s–F1-score, S_n–sensitivity, S_p–specificity.

Algorithm	Training set						Testing set					
	A_c	P_r	R_e	F_s	S_n	S_p	A_c	P_r	R_e	F_s	S_n	S_p
VGG16	84% ± 0.93	0.84 ± 0.009	0.84 ± 0.009	0.84 ± 0.009	0.73 ± 0.012	0.89 ± 0.008	76% ± 2.292	0.76 ± 0.023	0.76 ± 0.023	0.76 ± 0.023	0.55 ± 0.031	0.86 ± 0.018
ResNet50	67% ± 1.344	0.44 ± 0.018	0.67 ± 0.013	0.53 ± 0.016	0	1	67% ± 2.7	0.44 ± 0.035	0.67 ± 0.027	0.53 ± 0.032	0	1
ResNet101	67% ± 1.344	0.44 ± 0.018	0.67 ± 0.013	0.53 ± 0.016	0	1	67% ± 2.7	0.44 ± 0.035	0.67 ± 0.027	0.53 ± 0.032	0	1
Xception	**100%**	**1**	**1**	**1**	**1**	**1**	**80% ± 2.1**	**0.80 ± 0.021**	**0.80 ± 0.021**	**0.80 ± 0.021**	**0.63 ± 0.028**	**0.88 ± 0.016**
EfficientNetB0	67% ± 1.34	0.44 ± 0.018	0.67 ± 0.013	0.53 ± 0.016	0	1	67% ± 2.68	0.33 ± 0.038	0.50 ± 0.033	0.4 ± 0.036	0	1
EfficientNetB7	67% ± 1.34	0.44 ± 0.018	0.67 ± 0.013	0.53 ± 0.016	0	1	67% ± 2.68	0.33 ± 0.038	0.50 ± 0.033	0.4 ± 0.036	0	1
NasNetLarge	100%	1	1	1	0.99 ± 0.002	0.99 ± 0.002	79% ± 2.10	0.79 ± 0.021	0.79 ± 0.021	0.79 ± 0.021	0.62 ± 0.029	0.88 ± 0.016
EfficientNetV2M	67% ± 1.34	0.44 ± 0.018	0.67 ± 0.013	0.53 ± 0.016	0	1	67% ± 2.68	0.33 ± 0.038	0.50 ± 0.033	0.4 ± 0.036	0	1
ResNet152V2	100%	1	1	1	0.99 ± 0.002	0.99 ± 0.002	79% ± 2.1	0.78 ± 0.022	0.79 ± 0.021	0.78 ± 0.022	0.55 ± 0.031	0.91 ± 0.014
EfficientNetV2L	67% ± 1.34	0.44 ± 0.018	0.67 ± 0.013	0.53 ± 0.016	0	1	67% ± 2.68	0.33 ± 0.038	0.50 ± 0.033	0.4 ± 0.036	0	1

Table 12
Model's performance using Sgd optimizer for Study Two, along with confidence interval (α = 0.05). A_c–accuracy, P_r–precision, R_e–recall, F_s–F1-score, S_n–sensitivity, S_p–specificity.

Algorithm	Trainset						Testset					
	A_c	P_r	R_e	F_s	S_n	S_p	A_c	P_r	R_e	F_s	S_n	S_p
VGG16	84% ± 0.93	0.84 ± 0.009	0.84 ± 0.009	0.84 ± 0.009	0.73 ± 0.012	0.89 ± 0.008	76% ± 2.292	0.76 ± 0.023	0.76 ± 0.023	0.76 ± 0.023	0.55 ± 0.031	0.86 ± 0.018
ResNet50	67% ± 1.344	0.44 ± 0.018	0.67 ± 0.013	0.53 ± 0.016	0	1	67% ± 2.7	0.44 ± 0.035	0.67 ± 0.027	0.53 ± 0.032	0	1
ResNet101	67% ± 1.344	0.44 ± 0.018	0.67 ± 0.013	0.53 ± 0.016	0	1	67% ± 2.7	0.44 ± 0.035	0.67 ± 0.027	0.53 ± 0.032	0	1
Xception	**100%**	**1**	**1**	**1**	**1**	**1**	**0.8 ± 0.021**	**80% ± 2.1**	**0.80 ± 0.021**	**0.80 ± 0.021**	**0.63 ± 0.028**	**0.88 ± 0.016**
EfficientNetB0	67% ± 1.34	0.44 ± 0.018	0.67 ± 0.013	0.53 ± 0.016	0	1	67% ± 2.68	0.33 ± 0.038	0.50 ± 0.033	0.4 ± 0.036	0	1
EfficientNetB7	67% ± 1.34	0.44 ± 0.018	0.67 ± 0.013	0.53 ± 0.016	0	1	67% ± 2.68	0.33 ± 0.038	0.50 ± 0.033	0.4 ± 0.036	0	1
NasNetLarge	100%	1	1	1	0.99 ± 0.002	0.99 ± 0.002	0.79 ± 0.021	79% ± 2.1	0.79 ± 0.021	0.79 ± 0.021	0.62 ± 0.029	0.88 ± 0.016
EfficientNetV2M	67% ± 1.34	0.44 ± 0.018	0.67 ± 0.013	0.53 ± 0.016	0	1	67% ± 2.68	0.33 ± 0.038	0.50 ± 0.033	0.4 ± 0.036	0	1
ResNet152V2	100%	1	1	1	0.99 ± 0.002	0.99 ± 0.002	0.79 ± 0.021	0.78 ± 0.022	0.79 ± 0.021	0.78 ± 0.022	0.55 ± 0.031	0.91 ± 0.014
EfficientNetV2L	67% ± 1.34	0.44 ± 0.018	0.67 ± 0.013	0.53 ± 0.016	0	1	67% ± 2.68	0.33 ± 0.038	0.50 ± 0.033	0.4 ± 0.036	0	1

Table 13
Model's performance using Rmsprop optimizer for Study Two, along with confidence interval (α = 0.05). A_c–accuracy, P_r–precision, R_e–recall, F_s–F1-score, S_n–sensitivity, S_p–specificity.

Algorithm	Training set						Testing set					
	A_c	P_r	R_e	F_s	S_n	S_p	A_c	P_r	R_e	F_s	S_n	S_p
VGG16	87% ± 0.844	0.87 ± 0.008	0.87 ± 0.008	0.87 ± 0.008	0.82 ± 0.010	0.89 ± 0.008	77% ± 2.244	0.77 ± 0.022	0.77 ± 0.022	0.77 ± 0.022	0.62 ± 0.029	0.85 ± 0.018
ResNet50	67% ± 1.344	0.44 ± 0.018	0.67 ± 0.013	0.53 ± 0.016	0	1	67% ± 2.7	0.44 ± 0.035	0.67 ± 0.027	0.53 ± 0.032	0	1
ResNet101	67% ± 1.344	0.44 ± 0.018	0.67 ± 0.013	0.53 ± 0.016	0	1	67% ± 2.7	0.44 ± 0.035	0.67 ± 0.027	0.53 ± 0.032	0	1
Xception	**94% ± 0.57**	**.94 ± 0.006**	**.94 ± 0.006**	**.94 ± 0.006**	**0.95 ± 0.005**	**0.92 ± 0.007**	**75% ± 2.3**	**0.75 ± 0.023**	**0.75 ± 0.023**	**0.75 ± 0.023**	**0.58 ± 0.030**	**0.84 ± 0.019**
EfficientNetB0	67% ± 1.344	0.44 ± 0.018	0.67 ± 0.013	0.53 ± 0.016	0	1	67% ± 2.7	0.44 ± 0.035	0.67 ± 0.027	0.53 ± 0.032	0	1
EfficientNetB7	67% ± 1.344	0.44 ± 0.018	0.67 ± 0.013	0.53 ± 0.016	0	1	67% ± 2.7	0.44 ± 0.035	0.67 ± 0.027	0.53 ± 0.032	0	1
NasNetLarge	91% ± 0.7	0.91 ± 0.007	0.91 ± 0.007	0.91 ± 0.007	0.93 ± 0.006	0.9 ± 0.007	75% ± 2.3	0.75 ± 0.023	0.75 ± 0.023	0.75 ± 0.023	0.58 ± 0.030	0.84 ± 0.019
EfficientNetV2M	67% ± 1.3	0.44 ± 0.018	0.67 ± 0.013	0.53 ± 0.016	0	1	75% ± 2.3	0.75 ± 0.023	0.75 ± 0.023	0.75 ± 0.023	0.58 ± 0.030	0.84 ± 0.019
ResNet152V2	90% ± 0.7	0.90 ± 0.007	0.9 ± 0.007	0.9 ± 0.007	0.93 ± 0.006	0.88 ± 0.008	77% ± 2.2	0.78 ± 0.022	0.77 ± 0.022	0.78 ± 0.022	0.73 ± 0.024	0.79 ± 0.021
EfficientNetV2L	90% ± 0.7	0.91 ± 0.007	0.9 ± 0.007	0.9 ± 0.007	0.93 ± 0.006	0.88 ± 0.008	67% ± 2.7	0.44 ± 0.035	0.67 ± 0.027	0.53 ± 0.032	0	1

ResNet101, while the worst performance was found for ResNet152V2. Note that even though some of the DL models, such as VGG16 and ResNet50, demonstrate almost perfect accuracy on the training set, that performance is significantly lower on the testing set. In contrast, the performance of ResNet101 remains consistent for both the training and testing sets.

Table 15 presents the overall accuracy, precision, recall, F1 score, sensitivity, and specificity scores derived from the preliminary computations performed on the train and test set for ten different models using Sgd optimizer. Across all measures, the EfficientNetB7 and EfficientNetV2M models show the best performance. while ResNet50, Xception, NasNetLarge, and ResNet152V2 exhibit the worst performance.

Table 14
Model's performance using Adam optimizer for Study Three, along with confidence interval (α = 0.05). A_c–accuracy, P_r–precision, R_e–recall, F_s–F1-score, S_n–sensitivity, S_p–specificity.

Algorithm	Training set						Testing set					
	A_c	P_r	R_e	F_s	S_n	S_p	A_c	P_r	R_e	F_s	S_n	S_p
VGG16	100%	1	1	1	0.99 ± 0.004	0.99 ± 0.004	83% ± 3.1	0.82 ± 0.032	0.83 ± 0.031	0.81 ± 0.033	0.92 ± 0.022	0.73 ± 0.040
ResNet50	100%	1	1	1	1	.99 ± 0.004	86% ± 2.9	0.86 ± 0.029	0.86 ± 0.029	0.84 ± 0.031	0.94 ± 0.019	0.77 ± 0.037
ResNet101	**99% ± 0.381**	**0.99 ± 0.004**	**0.99 ± 0.004**	**0.99 ± 0.004**	**0.99 ± 0.004**	**0.98 ± 0.005**	**99% ± 0.8**	**0.99 ± 0.008**	**0.99 ± 0.008**	**0.99 ± 0.008**	**0.99 ± 0.008**	**0.98 ± 0.011**
Xception	40% ± 2.95	0.49 ± 0.027	0.4 ± 0.030	0.23 ± 0.034	0.75 ± 0.019	0.25 ± 0.033	40% ± 5.9	0.16 ± 0.070	0.4 ± 0.059	0.23 ± 0.067	0.75 ± 0.038	0.25 ± 0.066
EfficientNetB0	100%	1	1	1	0.99 ± 0.004	0.99 ± 0.004	86% ± 2.9	0.87 ± 0.028	0.86 ± 0.029	0.86 ± 0.029	0.95 ± 0.017	0.8 ± 0.034
EfficientNetB7	100%	1	1	1	0.99 ± 0.004	0.99 ± 0.004	86% ± 2.9	0.87 ± 0.028	0.86 ± 0.029	0.86 ± 0.029	0.95 ± 0.017	0.8 ± 0.034
NASNetLarge	61% ± 2.38	0.46 ± 0.028	0.61 ± 0.024	0.52 ± 0.026	0.84 ± 0.015	0.42 ± 0.029	58% ± 4.9	0.43 ± 0.058	0.58 ± 0.049	0.49 ± 0.054	0.8 ± 0.034	0.39 ± 0.060
EfficientNetV2M	100%	1	1	1	1	1	86% ± 2.9	0.87 ± 0.028	0.86 ± 0.029	0.87 ± 0.028	0.95 ± 0.017	0.82 ± 0.032
ResNet152V2	40% ± 2.95	0.16 ± 0.035	0.4 ± 0.030	0.23 ± 0.034	0.75 ± 0.019	0.25 ± 0.033	40% ± 5.9	0.16 ± 0.070	0.4 ± 0.059	0.23 ± 0.067	0.75 ± 0.038	0.25 ± 0.066
EfficientNetV2L	100%	1	1	1	1	1	85% ± 0.030	0.85 ± 0.030	0.85 ± 0.030	0.85 ± 0.030	0.95 ± 0.017	0.82 ± 0.032

Table 15
Model's performance using Sgd optimizer for Study Three, along with confidence interval (α = 0.05). A_c–accuracy, P_r–precision, R_e–recall, F_s–F1-score, S_n–sensitivity, S_p–specificity.

Algorithm	Training set						Testing set					
	A_c	P_r	R_e	F_s	S_n	S_p	A_c	P_r	R_e	F_s	S_n	S_p
VGG16	96% ± 0.8	0.96 ± 0.008	0.96 ± 0.008	0.96 ± 0.008	0.98 ± 0.005	0.94 ± 0.009	84% ± 3.10	0.85 ± 0.030	0.84 ± 0.031	0.83 ± 0.031	0.94 ± 0.019	0.75 ± 0.038
ResNet50	40% ± 3.0	0.16 ± 0.035	0.4 ± 0.030	0.23 ± 0.034	0.75 ± 0.019	0.25 ± 0.033	86% ± 2.9	0.85 ± 0.030	0.86 ± 0.029	0.85 ± 0.030	0.94 ± 0.019	0.78 ± 0.036
ResNet101	100%	1	1	1	0.99 ± 0.004	0.99 ± 0.004	84% ± 3.1	0.83 ± 0.031	0.84 ± 0.031	0.83 ± 0.031	0.94 ± 0.019	0.76 ± 0.037
Xception	40% ± 3.0	0.16 ± 0.035	0.4 ± 0.030	0.23 ± 0.034	0.75 ± 0.019	0.25 ± 0.033	40% ± 5.9	0.16 ± 0.070	0.4 ± 0.059	0.23 ± 0.067	0.75 ± 0.038	0.25 ± 0.066
EfficientNetB0	100%	1	1	1	1	1	88% ± 2.6	0.88 ± 0.026	0.88 ± 0.026	0.88 ± 0.026	0.95 ± 0.017	0.825 ± 0.032
EfficientNetB7	**100%**	**1**	**1**	**1**	**1**	**1**	**89% ± 2.5**	**0.88 ± 0.026**	**0.89 ± 0.025**	**0.88 ± 0.026**	**0.95 ± 0.017**	**0.85 ± 0.030**
NASNetLarge	40% ± 0.030	0.28 ± 0.032	0.4 ± 0.030	0.24 ± 0.033	0.75 ± 0.019	0.25 ± 0.033	40% ± 5.9	0.16 ± 0.070	0.4 ± 0.059	0.23 ± 0.067	0.75 ± 0.038	0.25 ± 0.066
EfficientNetV2M	**100%**	**1**	**1**	**1**	**1**	**1**	**89% ± 2.5**	**0.88 ± 0.026**	**0.89 ± 0.025**	**0.88 ± 0.026**	**0.95 ± 0.017**	**0.85 ± 0.030**
ResNet152V2	40% ± 3.0	0.16 ± 0.035	0.4 ± 0.030	0.23 ± 0.034	0.75 ± 0.019	0.25 ± 0.033	40% ± 5.9	0.16 ± 0.070	0.4 ± 0.059	0.23 ± 0.067	0.75 ± 0.038	0.25 ± 0.066
EfficientNetV2L	100%	1	1	1	0.99 ± 0.004	0.99 ± 0.004	86% ± 2.90	0.86 ± 0.029	0.86 ± 0.029	0.86 ± 0.029	0.94 ± 0.019	0.83 ± 0.031

Table 16
Model's performance using Rmsprop optimizer for Study Three, along with confidence interval (α = 0.05). A_c–accuracy, P_r–precision, R_e–recall, F_s–F1-score, S_n–sensitivity, S_p–specificity.

Algorithm	Training set						Testing set					
	A_c	P_r	R_e	F_s	S_n	S_p	A_c	P_r	R_e	F_s	S_n	S_p
VGG16	100%	1	1	1	0.99 ± 0.004	0.99 ± 0.004	83% ± 3.1	0.83 ± 0.031	0.83 ± 0.031	0.82 ± 0.032	0.93 ± 0.020	0.73 ± 0.040
ResNet50	100%	1	1	1	0.99 ± 0.004	0.99 ± 0.004	86% ± 2.9	0.85 ± 0.030	0.86 ± 0.029	0.84 ± 0.031	0.94 ± 0.019	0.78 ± 0.036
ResNet101	99% ± 0.381	0.99 ± 0.004	0.99 ± 0.004	0.99 ± 0.004	0.99 ± 0.004	0.98 ± 0.005	83% ± 3.1	0.82 ± 0.032	0.83 ± 0.031	0.82 ± 0.032	0.94 ± 0.019	0.74 ± 0.039
Xception	52% ± 2.6	0.49 ± 0.027	0.52 ± 0.026	0.42 ± 0.029	0.79 ± 0.017	0.34 ± 0.031	52% ± 5.30	0.51 ± 0.053	0.52 ± 0.053	0.42 ± 0.058	0.79 ± 0.035	0.33 ± 0.062
EfficientNetB0	**100%**	**1**	**1**	**1**	**0.97 ± 0.007**	**0.97 ± 0.007**	**89% ± 2.50**	**0.88 ± 0.026**	**0.89 ± 0.025**	**0.88 ± 0.026**	**0.96 ± 0.015**	**0.83 ± 0.031**
EfficientNetB7	99% ± 0.381	0.99 ± 0.004	0.99 ± 0.004	0.99 ± 0.004	0.99 ± 0.004	0.98 ± 0.005	86% ± 2.9	0.88 ± 0.026	0.86 ± 0.029	0.86 ± 0.029	0.95 ± 0.017	0.86 ± 0.029
NasNetLarge	62% ± 2.4	0.59 ± 0.024	0.62 ± 0.024	0.56 ± 0.025	0.86 ± 0.014	0.45 ± 0.028	61% ± 4.8	0.6 ± 0.048	0.61 ± 0.048	0.53 ± 0.052	0.84 ± 0.031	0.41 ± 0.059
EfficientNetV2M	100%	1	1	1	1	1	86% ± 2.9	0.86 ± 0.029	0.86 ± 0.029	0.86 ± 0.029	0.94 ± 0.019	0.83 ± 0.03
ResNet15V2	40% ± 3.0	0.16 ± 0.035	0.4 ± 0.030	0.23 ± 0.034	0.75 ± 0.019	0.25 ± 0.033	40% ± 5.9	0.16 ± 0.070	0.4 ± 0.059	0.23 ± 0.067	0.75 ± 0.038	0.25 ± 0.066
EfficientNetV2L	99% ± 0.4	0.99 ± 0.004	0.99 ± 0.004	0.99 ± 0.004	0.99 ± 0.004	0.98 ± 0.005	85% ± 3.0	0.84 ± 0.031	0.85 ± 0.030	0.84 ± 0.031	0.94 ± 0.019	0.81 ± 0.033

In Table 16, for Study Three, EfficientNetB0 models trained with Rmsprop displayed the best performance, while ResNet152V2 demonstrated the worst performance among all of the models across all measures.

Fig. 5 depicts some of the DL model's performance on the training and testing sets during each Study. Based on Fig. 5, it can be inferred that the Xception model with Sgd, the ResNet152V2 model with Rmsprop, and the EfficientNetB7 model with Adam all performed well and did not show signs of overfitting during the training phase. On the other hand, the EfficientNetV2L model with Sgd, the EfficientNetV2M model with Rmsprop, and the NasNetLarge model with Adam all showed minimal performance.

Fig. 6 presents some of the DL model's performance using a confusion matrix for the train set. From the figure, it can be observed that in Study One, VGG16 trained with Adam misclassified only one train sample. In Study Two, Xception trained with Sgd classified all of the training samples correctly, whereas in Study Three, EfficientNetB0 misclassified one sample.

Fig. 7 presents some of the DL model's performance using a confusion matrix for the training set. From the figure, it can be observed that in Study One, VGG16 trained with Adam misclassified only one train sample. In Study Two, Xception trained with Sgd misclassified 70 samples, whereas in Study Three, EfficientNetB0 misclassified 15 sample.

In Fig. 8, the Area Under the Receiver Operating Characteristic curve(AUC-ROC) is plotted as a curve on a graph with the true positive rate (TPR) on the y-axis and the false positive rate (FPR) on the x-axis. AUC-ROC ranges from 0 to 1, with higher values indicating better performance. In Fig. 8(a) and (b), the AUC-ROC curve is plotted for binary classification, whereas in Fig. 8(c), the AUC-ROC curve is plotted for multiclass classification.

4.4. Missclassification

The total number of false positives and negatives predicted by each DL-based model for both the training set and testing set is measured in Table 17. Here, the algorithm with the lowest misclassification rate is emphasized in bold red fonts and is the primary concern that will assist in determining the true potential of the DL-based models' performance across three independent studies. In Study One, Xception models trained with Rmsprop performed the best (as shown in the table), incorrectly identifying a total of merely one test sample. In Study Two, Xception models trained with Sgd demonstrated promising performance by misclassifying seventy test samples, whereas in Study Three, EfficeintNetV2M misclassified fifteen test samples and displayed the best outcomes once trained with the Sgd optimizer.

4.5. Complexity of the model

In Table 18, we measured various models' complexity using FLOps. The table shows that DL models with our proposed architecture reduced the number of parameters (NP) and the number of FLOPs. For instance, with our proposed TL approaches, we reduced the VGG16 model's NP almost nine times, and the FLOPs were reduced up to 200M. The optimal FLOPs were calculated for NasNetLarge models, where the FLOPs value was reduced up to 86%.

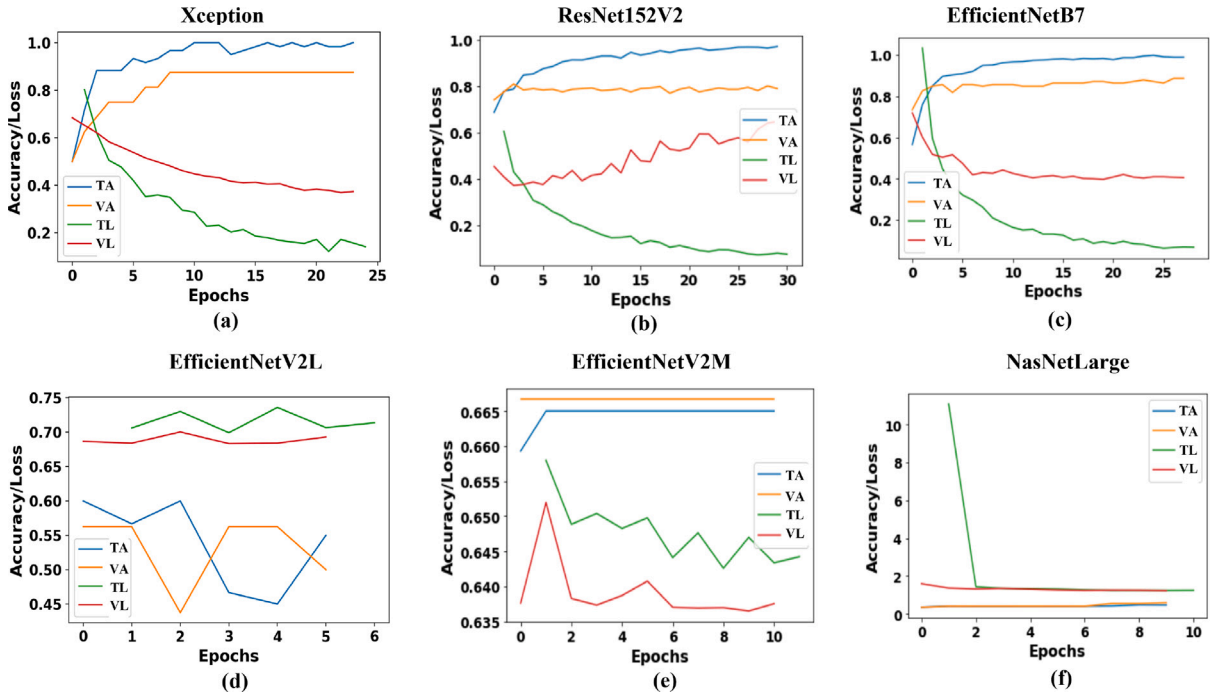

Fig. 5. Accuracy and loss visualization per each epoch of some DL-based models used in this study. For Study One, (a) the Xception trained with Sgd; for Study Two, (b) ResNet152V2 trained with Rmsprop; and (c) EfficientNetB7 trained with Adam for Study Three shows no overfitting. On the other hand, (d) EfficientNetV2L trained with Sgd during Study One, (e) EfficientNetV2M trained with Rmsprop during Study Two, and (f) NasNetLarge trained with Adam shows minimal performance for Study Three.

	M_x	N_l
M_x	34	0
N_l	1	25

(a)

	M_x	N_l
M_x	470	0
N_l	0	933

(b)

	C_x	M_s	M_x	N_l
C_x	79	0	1	0
M_s	0	64	0	0
M_x	0	0	211	0
N_l	0	0	0	172

(c)

Fig. 6. Confusion matrices of some of the DL-based models considering the training set: in Study One, the performance was observed for (a) VGG16 using Adam; in Study Two, (b) Xception using Sgd; and in Study Three, (c) EfficientNetB0 using the Rmsprop optimizer. M_x–Monkeypox, C_x–Chickenpox, M_s–Measles, N_l–Normal.

	M_x	N_l
M_x	9	0
N_l	1	6

(a)

	M_x	N_l
M_x	74	43
N_l	27	207

(b)

	C_x	M_s	M_x	N_l
C_x	11	1	7	1
M_s	1	14	1	0
M_x	0	0	52	1
N_l	2	0	1	40

(c)

Fig. 7. Confusion matrices of some of the DL-based models considering the testing set: in Study One, the performance was observed for (a) VGG16 using Adam; in Study Two, (b) Xception using Sgd; and in Study Three, (c) EfficientNetB0 using the Rmsprop optimizer. C_x–Chickenpox, M_s–Measles, N_l–Normal.

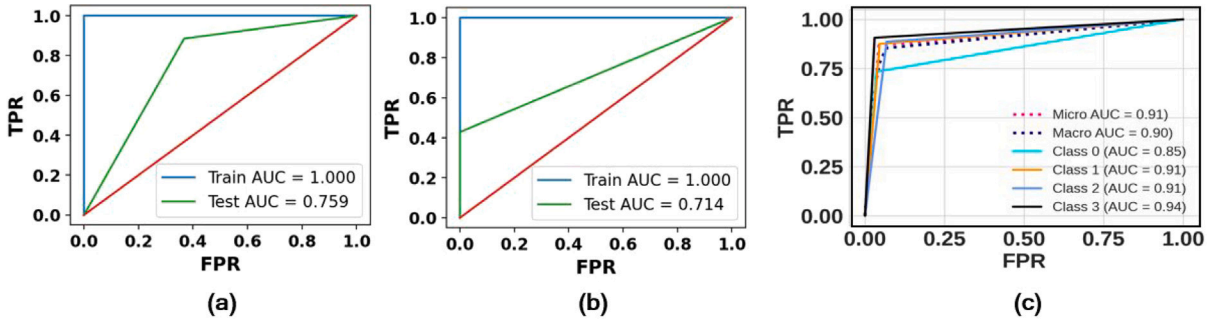

Fig. 8. AUC-ROC score of some of the DL-based models: in Study One, the performance was observed for (a) VGG16 using Adam; in Study Two, (b) Xception using Sgd; and in Study Three, (c) EfficientNetB0 using the Rmsprop optimizer. TPR—true positive rate, FPR—false positive rate.

Table 17
Misclassification of different transfer learning models used in three separate study. T_r–train set, T_s–test set.

Study	Optimizer	Misclassification = False positive (F_p) + False negative (F_n)																	
		Vgg16		ResNet50		ResNet101		Xception		EfficientNetB0		EfficientNetB7		NASNetLarge		EfficientNetV2M		ResNet152V2		EfficientNetV2L	
		T_r	T_s	T_r	T_s	T_r	T_s	T_r	T_s	T_r	T_s	T_r	T_s	T_r	T_s	T_r	T_s	T_r	T_s	T_r	T_s
One	Adam	**1**	**1**	26	7	26	7	20	5	26	7	26	7	0	2	0	7	0	2	26	7
	Sgd	1	2	26	7	26	7	**0**	**2**	26	7	26	7	0	3	26	7	0	3	30	8
	Rmsprop	0	2	26	7	26	7	**0**	**1**	26	7	26	7	0	3	26	7	0	32	6	7
Two	Adam	224	84	470	117	470	117	**186**	**70**	470	117	470	117	461	115	470	117	345	101	470	117
	Sgd	224	84	470	117	470	117	**0**	**70**	470	117	470	117	4	73	470	117	6	74	470	117
	Rmsprop	182	80	470	117	470	117	**70**	**86**	470	117	470	117	123	81	470	117	140	79	470	117
Three	Adam	2	23	1	18	5	21	315	79	**0**	**18**	2	19	204	55	**0**	**18**	316	79	0	20
	Sgd	20	21	1	19	1	21	316	79	0	16	0	15	314	79	**0**	**15**	316	79	1	18
	Rmsprop	2	22	1	19	4	22	253	63	**1**	**15**	4	19	198	52	0	18	316	80	5	20

Table 18
Computational complexity of the transfer learning model used in this study. FLOPS–floating-point operations per second, NP–number of parameters.

Algorithm	Regular parameters		Optimized parameters	
	FLOPS (Millions)	NP	FLOPS (Millions)	NP
VGG16	30 960M	138M	30 713.53M	**15M**
ResNet50	7751M	23.58M	7753M	24.76M
ResNet101	15 195M	42.65M	15 196.89M	43.83M
Xceptions	16 773.72M	22.91M	9136.42M	22.04M
EfficientNetB0	803.20M	5.33M	802M	4.78M
EfficientNetB7	76 868.53M	66.65M	**10 528.44M**	65.57M
NasNetLarge	47 801.95 M	88.94M	20 667.92M	87.23M
EfficientNetV2M	49 574.083M	54.43M	10 813.16M	53.88M
ResNet15V2	21 879.78 904M	60.38M	21 878.02M	59.51M
EficientNetV2L	11 2877.75M	119.02M	24 619.02M	118.48M

In Table 19, we have outlined the overall process time for each module deployed during the course of this study. An early stop mechanism is utilized during training to prevent data leaks and overfitting issues. The patience level was set to 3, indicating that if the validation loss performance does not change after three iterations, the model will halt training to prevent further overfitting and computational difficulties. We compute process time per epoch due to the fact that each algorithm and optimizer stops at a separate epoch. VGG16 and Xception demonstrate the most promising outcomes in terms of total processing times, as indicated by the bold and red fonts, as shown in Table 19.

We ran additional experiments on the ResNet50 and Xception models using five more optimizers,[1] including Adadelta, Adagrad, Adamax, Follow the Regularized Leader (Ftrl), and Nesterov-accelerated Adaptive Moment Estimation (Nadam), to comprehend the proposed models' performance on different optimizers as shown in Table 20. The validation accuracy and loss were calculated for epochs 1, 5, and 10, respectively. We discovered that there is no major performance difference among optimizers for ResNet50. However, for the Xception model, we discovered that the validation accuracy of Adam, Rmsprop, and Sgd is considerably more promising, as shown in Table 20 (highlighted with bold font).

5. Discussions

There are currently limited publications that suggest a CNN-based Monkeypox disease diagnosis; as an effect, direct comparison of our findings with previous studies on a broad scale is limited, but a higher-level assessment of the provided performance indicators is still possible. Table 21 compares the performance of several TL algorithms presented in recent research for Monkeypox disease diagnosis. For example, Islam et al. (2022) used ShuffleNet-V2 to attain a maximum F-measure of 0.67 and a precision of 0.79 (Islam, Hussain, Chowdhury, & Islam, 2022). Using ResNet50, Ali et al. (2022) achieve a maximum precision of 0.85 and recall of 0.83 (Ali et al., 2022). Our investigation in Studies One and Two showed that Xception models performed the best and had similar results when trained with Sgd and Rmsprop optimizers. In Study

[1] Adadelta, Adagrad, and Adamax are all variants of stochastic gradient descent (SGD) (Dogo, Afolabi, Nwulu, Twala, & Aigbavboa, 2018).

Table 19
Overall process time for each modules used in this study. E_{sp}–early stops/epochs, P_{ts}–process time/seconds, Pt_{es}–process time/epochs(seconds).

Algorithm	Study one									Study two									Study three								
	Adam			Sgd			Rmsprop			Adam			Sgd			Rmsprop			Adam			Sgd			Rmsprop		
	E_{sp}	P_{ts}	Pt_{es}	E_{sp}	P_{ts}	Pt_{es}	E_{sp}	P_{ts}	Pt_{es}	E_{sp}	P_{ts}	Pt_{es}	E_{sp}	P_{ts}	Pt_{es}	E_{sp}	P_{ts}	Pt_{es}	E_{sp}	P_{ts}	Pt_{es}	E_{sp}	P_{ts}	Pt_{es}	E_{sp}	P_{ts}	Pt_{es}
VGG16	50	44.51	0.89	7	7.01	1.001	14	13.35	0.953	54	875.63	16.22	11	194.69	17.69	24	389.43	16.23	22	12.83	0.58	24	13.36	0.56	19	10.81	0.59
ResNet50	18	21.17	1.17	6	10.09	1.68	12	15.42	1.285	34	529.22	15.57	10	157.74	15.77	12	191.66	15.97	19	15.004	0.789	26	15.87	0.61	23	14.93	0.65
ResNet101	17	25.95	1.53	29	45.52	1.57	16	25.29	1.58	16	277.22	17.32	12	211.89	17.65	9	158.83	17.64	12	16.95	1.42	23	24.51	1.06	15	19.91	1.33
Xception	11	10.41	0.946	24	13.25	0.55	11	7.81	0.71	27	135.37	5.01	50	247.77	4.96	35	165.21	4.72	11	13.57	1.23	30	21.63	0.72	6	6.23	1.03
EfficientNetB0	11	10.41	0.94	12	10.26	0.85	50	20.15	0.403	14	84.10	6	8	53.60	6.7	9	60.49	6.72	28	22.64	0.81	19	17.09	0.89	19	15.26	0.803
EfficientNetB7	50	79.55	1.591	9	32.13	3.57	50	78.34	1.56	8	162.62	20.33	18	331.84	18.44	15	289.05	19.27	17	106.37	6.25	28	151.64	5.41	20	113.62	5.68
NasNetLarge	10	33.61	3.361	22	47.72	2.17	19	45.58	2.39	6	142.43	23.74	41	630.23	15.37	16	353.74	22.11	10	71.58	7.16	30	160.54	5.35	16	93.43	5.84
EfficientNetV2M	50	57.57	1.15	6	25.57	4.26	8	27.45	3.43	7	118.91	16.98	11	186.42	16.94	10	174.93	17.49	10	45.97	4.59	23	78.74	3.42	21	71.92	3.42
ResNet152V2	12	19.02	1.58	18	23.19	1.28	9	16.62	1.84	13	143.89	11.06	30	333.22	11.107	19	217.28	11.44	30	42.93	1.43	30	38.35	1.28	30	39.65	1.32
EfficientNetV2L	50	97.24	1.94	6	36.08	6.013	5	34.29	6.85	7	184.24	26.32	13	297.89	22.91	10	264.48	26.44	21	151.30	7.2	22	151.91	6.91	12	96.23	8.01

Table 20
ResNet50 and Xception model performance with different optimizer. V_a–validation accuracy, V_l–validation loss.

Algorithm	Optimizer	Epoch1		Epoch5		Epoch10		Total time
		V_a	V_l	V_a	V_l	V_a	V_l	
ResNet50	Adadelta	0.437	0.965	0.437	0.909	0.437	0.864	6.856
	Adagrad	0.562	0.690	0.562	0.687	0.562	0.691	6.520
	Adamax	0.562	0.992	0.562	0.698	0.562	0.690	6.277
	Ftrl	0.562	0.690	0.562	0.689	0.562	0.688	6.821
	Nadam	0.562	0.686	0.562	0.693	0.562	0.692	6.634
	Adam	0.437	0.846	0.562	0.693	0.562	0.697	6.86
	SGD	0.562	0.702	0.562	0.692	0.562	0.692	6.369
	RMSprop	0.562	0.693	0.562	0.692	0.562	0.691	6.898
Xception	Adadelta	0.375	0.866	0.375	0.851	0.375	0.833	9.499
	Adagrad	0.812	0.626	0.875	0.424	0.937	0.343	6.104
	Adamax	0.750	0.587	0.8125	0.4119	0.812	0.364	6.127
	Ftrl	0.562	0.692	0.562	0.692	0.562	0.691	6.21
	Nadam	0.875	0.414	0.875	0.372	0.812	0.622	6.456
	Adam	0.750	0.630	0.875	0.342	0.937	0.415	6.189
	SGD	0.750	0.573	0.875	0.320	0.875	0.293	6.119
	RMSprop	0.625	0.915	0.875	0.398	0.875	0.643	6.261

Table 21
Comparison of the proposed DL-based model with existing literature that considered the Monkeypox disease diagnosis model.

Reference	Method	Dataset size	Accuracy	Precision	Recall	F-measure
(Islam et al., 2022)	ResNet50	117 Monkeypox, 687 others	72	0.59	0.51	0.55
	Inception-V3		71	0.71	0.53	0.61
	ShuffleNet-V2		79%	0.79	0.58	0.67
(Ali et al., 2022)	VGG16	102 Monkeypox, 126 others	81.48 ± 6.87	0.85 ± 0.08	0.81 ± 0.05	0.83 ± 0.06
	ResNet50		82.96 ± 4.57	0.87 ± 0.07	0.83 ± 0.02	0.84 ± 0.03
	InceptionV3		74.07 ± 3.78	0.74 ± 0.02	0.81 ± 0.07	0.78 ± 0.04
	Ensemble		79.26 ± 1.05	0.84 ± 0.05	0.79 ± 0.07	0.81 ± 0.02
(Irmak et al., 2022)	MobileNetV2	770 images	91.38%	0.905	0.86	0.88
Our best model (Test set)						
Study one	**Xception**	43 Monkeypox, 33 others	**94% ± 7.591**	**0.94 ± 0.054**	**0.94 ± 0.054**	**0.94 ± 0.054**
Study two		587 Monkeypox, 1167 others	**80% ± 0.021**	**0.80 ± 0.021**	**0.80 ± 0.021**	**0.80 ± 0.021**
Study three	**ResNet101**	264 Monkeypox, 395 others	**99% ± 0.80**	**0.99 ± 0.008**	**0.99 ± 0.008**	**0.99 ± 0.008**

Three, we discovered that ResNet101 had the best performance on the training and testing sets when trained with the Adam.

Most of the previous studies did not provide enough explanation regarding their higher accuracy results. Therefore, it is difficult to infer what factors play an essential role in their TL approaches. In our work, we found that without early stopping, the accuracy reaches almost 100% for most of the dataset, and also, during the training phase, the model started to overfit. An overfitting model produces biased results and is not a good option for model deployments in real-world diagnosis. To overcome such limitations, we have used early stopping in our work, which helps our model stop before it enters the overfitting phase. Because we employed early stopping approaches, our model's accuracy only indicates the performance prior to overfitting. During Studies One and Two, we noticed that some TL models, such as ResNet50 and EfficientNetV2M, had a lower performance.

One potential reason could be that the model's performance on the training data may not represent its true ability, as it has not been fully trained due to early stopping. In order to provide more justification for this issue, it may be helpful to perform additional experiments or analyses to assess the impact of early stopping on the model's performance.

To understand the performance of our model, we used LIME and presented the results for some of the DL-based models in Fig. 9. For example, in Fig. 9 (a,d) the minimum parameter settings for interpreting the Monkeypox-infected image using LIME are a maximum distance of 100, a kernel size of 2, and a ratio of 0.2. The maximum parameter settings are a maximum distance of 200, a kernel size of 6, and a ratio of 0.4 as shown in Fig. 9 (b,e). The optimal performance was observed with a maximum distance of 200, a kernel size of 4, and a ratio of 0.2 (Fig. 9 (c,f)). These results show that the proposed model was able to identify the top features that are important for making predictions about infected regions when using optimal parameter settings.

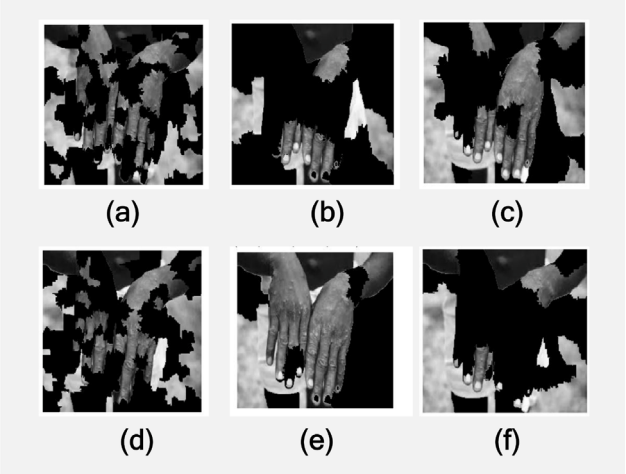

Fig. 9. Using LIME, the top four features that aid the proposed model in identifying potentially infected regions with the (a) minimum, (b) maximum, and (c) optimal parameters for VGG16, as well as (d) minimum, (e) maximum, and (f) optimal parameters for Xception, have been identified.

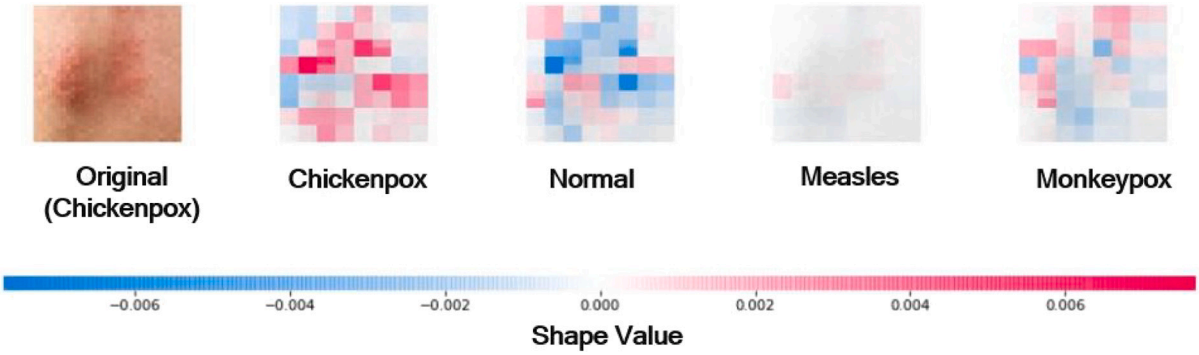

Fig. 10. Visualizing the SHAP value for test image.

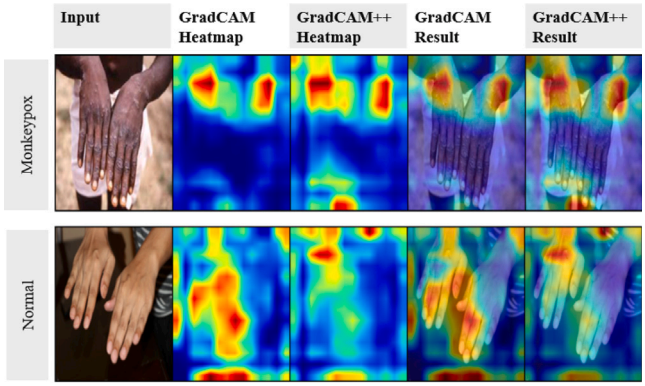

Fig. 11. Visual explanation of proposed models predictions using Grad-CAM and Grad-CAM++.

In Addition, we have used another agnostic-based interpretable approaches, SHAP, to interpret our proposed model's performance as well. Fig. 10, we have used the Chickenpox images, which were predicted using the ResNet50 model, and finally, SHAP is used to interpret the model's predictions. In Fig. 10, the color of each pixel in the image plot indicates that pixel's importance to the model's prediction. Pixels that are colored blue have a low importance, while pixels that are colored red have a high importance, and that information can be used to identify the essential features in the input image. From Fig. 10, we can understand what specific features play an essential role in the ResNet50 model to identify chickenpox-infected images correctly.

We further analyze the model performance using GradCAM (Selvaraju et al., 2016) and GradCAM++ (Chattopadhay, Sarkar, Howlader, & Balasubramanian, 2018). Based on our overall analysis, we found that, the top features identified by the LIME class activation function align with the region of GradCAM and GradCAM++. As an effect, our findings provide clearer explanations of our proposed models' predictions and their stability (see Fig. 11).

During the initial phase of the experiment, the majority of CNN models reached an accuracy of nearly 100%. Using early stopping approaches and testing many CNN models with various optimizers, we discovered that the model's actual accuracy is far lower compared to the initial findings. Using ResNet50, Ali et al. (2022) were able to attain a much superior performance result (Ali et al., 2022). Nevertheless, our research demonstrates that the performance of ResNet50 with different optimizers is dramatically reduced when the appropriate generalization and regularization techniques are applied throughout the investigation.

Before we began our study, there was no single image-based Monkeypox dataset. Therefore, it was difficult for the researcher and practitioners to develop and deploy an ML-based Monkeypox disease diagnosis model. Therefore, we develop a new dataset that can be used to train and develop ML models to classify the Monkeypox disease using image analysis techniques. In addition, a TL-based CNN model is developed, and its ability to differentiate between patients with and without Monkeypox disease is evaluated in three separate studies. A recent report presented by World Health Organization (WHO) encouraged any ML model needs to provide proper interpretation before being applied to a clinical trial (Organization et al., 2021). Considering this necessity, in this work, we have explained and overlooked our post-prediction analysis using one of the popular explainable AI techniques, LIME. Using LIME, we demonstrate that our models are capable of learning from the infected regions and localizing those areas.

It can be assumed that our new dataset will provide an immense opportunity for researchers and practitioners to practice and develop image-based analysis tools for the pilot test or analyzing Monkeypox disease diagnosis.

6. Limitations of the study and future works

During the onset of the Monkeypox disease, we came across several limitations that were beyond the scope of this research at the time. We provide the following as shortcomings of our study, which can be addressed in our future works in terms of tool and method selection:

1. At the time of this writing, the dataset and reference literature availability is minimal, making it difficult to assess the performance of our models confidently. The privacy issue is one of the primary reasons for this limited data availability. As Monkeypox often affects the entire body, it is frequently challenging to obtain and use images of infected faces, particularly for children. Nonetheless, there are a number of open repositories that are constantly gathering new data in order to increase the quantity of data. Therefore, data scarcity difficulties may diminish in the near future, and these datasets could be considered in future research to determine the nature of the performance of the proposed models.
2. Using early stopping techniques, it is frequently challenging to adequately compare the performance of several models, as the models may not have been trained to their full capacity. In the near future, an additional experiment will be considered to compare the model's performance with and without early stopping or to study the model's performance on various subsets of the training data to determine how it changes over time.
3. We could not conduct a pilot test in a clinical setting because we needed permission and enough facilities. However, in future work, we plan to conduct the pilot test in Bangladesh, where we expect it to be much easier to get permission.
4. A DL-based model trained on a dataset that combines all the patient's information, such as age, gender, and other physical symptoms, in conjunction with Monkeypox skin disease, will help to design a more convincing and accurate model on a large scale. However, during the data collection process, it was

almost impossible to find any patient's detailed information along with their infected body images that could be used for research purposes. We hope that, over time, various hospitals and clinical institutions will release such information to assist future researchers and practitioners in developing more complex models.

7. Conclusions

The study aims to develop a transfer learning (TL) model to distinguish between Monkeypox and normal individuals. Due to the data scarcity, we first develop a Monkeypox dataset. We conducted three separate studies considering binary classification (Study One and Two) and multiclass classification (Study Three) wherein the TL approach is used and tested with ten popular CNN models. Our findings suggest that, on one hand, using TL approaches, the proposed modified Xception models can distinguish patients with Monkeypox symptoms from others in both Study One and Two with accuracy ranging from 75% to 88%. On the other hand, ResNet101 demonstrated the best performance for multiclass classification (in Study Three), with accuracy ranging from 84% to 99%. By implementing Generalization and Regularization Approaches (GRA), we demonstrate that the TL-based model requires fewer trainable parameters and is computationally efficient in terms of performance. Finally, we have used LIME to present the proper explanation of the reason behind our model's prediction, which is one of the current demands in deploying ML models for clinical trials. We intend to emphasize the possibilities of artificial intelligence-based approaches, which might play an essential role in diagnosing and preventing the contamination of the onset of the Monkeypox virus.

We hope our publicly available dataset will play an important role and provide the opportunity to the ML researcher who cannot develop an AI-based model and is unable to conduct the experiment due to data scarcity. As our proposed model is supported by many previously published literature that uses the TL approach in developing an AI-based diagnosis model, it will also encourage future researchers and practitioners to take advantage of the TL approach to develop and deploy AI-based Monkeypox disease diagnosis in real world settings. As discussed in Section 6 of our work, a few limitations can be addressed by continuously adding new images of Monkeypox-infected patients to the dataset, testing the proposed model on highly imbalanced data, and developing a mobile-based diagnosis tool using our proposed model.

CRediT authorship contribution statement

Md Manjurul Ahsan: Conceptualization, Methodology, Software, Validation, Formal analysis, Writing – original draft, Visualization. **Muhammad Ramiz Uddin:** Conceptualization, Investigation, Resources, Data curation. **Md Shahin Ali:** Software, Investigation, Resources, Data curation. **Md Khairul Islam:** Investigation, Writing – review & editing. **Mithila Farjana:** Resources, Writing – review & editing. **Ahmed Nazmus Sakib:** Writing – review & editing. **Khondhaker Al Momin:** Writing – review & editing. **Shahana Akter Luna:** Writing – review & editing, Investigation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The dataset associated with this study is now publicly available and can be obtained from the following URL: https://github.com/mahsan2/Monkeypox-dataset-2022

References

Abbasimehr, H., Shabani, M., & Yousefi, M. (2020). An optimized model using LSTM network for demand forecasting. *Computers & Industrial Engineering*, *143*, Article 106435.

Abdelhamid, A. A., El-Kenawy, E.-S. M., Khodadadi, N., Mirjalili, S., Khafaga, D. S., Alharbi, A. H., et al. (2022). Classification of monkeypox images based on transfer learning and the Al-Biruni Earth Radius Optimization algorithm. *Mathematics*, *10*(19), 3614.

Adler, H., Gould, S., Hine, P., Snell, L. B., Wong, W., Houlihan, C. F., et al. (2022). Clinical features and management of human monkeypox: a retrospective observational study in the UK. *The Lancet Infectious Diseases*.

Ahsan, M. M., Ahad, M. T., Soma, F. A., Paul, S., Chowdhury, A., Luna, S. A., et al. (2021). Detecting SARS-CoV-2 from chest X-Ray using artificial intelligence. *Ieee Access*, *9*, 35501–35513.

Ahsan, M. M., Alam, T. E., Trafalis, T., & Huebner, P. (2020). Deep MLP-CNN model using mixed-data to distinguish between COVID-19 and Non-COVID-19 patients. *Symmetry*, *12*(9), 1526.

Ahsan, M. M., Gupta, K. D., Islam, M. M., Sen, S., Rahman, M., Shakhawat Hossain, M., et al. (2020). Covid-19 symptoms detection based on nasnetmobile with explainable ai using various imaging modalities. *Machine Learning and Knowledge Extraction*, *2*(4), 490–504.

Ahsan, M. M., Nazim, R., Siddique, Z., & Huebner, P. (2021). Detection of COVID-19 patients from CT scan and chest X-ray data using modified MobileNetV2 and LIME. In *Healthcare, Vol. 9* (p. 1099). Multidisciplinary Digital Publishing Institute.

Ahsan, M. M., & Siddique, Z. (2022). Machine learning-based heart disease diagnosis: A systematic literature review. *Artificial Intelligence in Medicine*, Article 102289.

Ahsan, M. M., Uddin, M. R., & Luna, S. A. (2022). Monkeypox image data collection. arXiv preprint arXiv:2206.01774.

Akiba, T., Suzuki, S., & Fukuda, K. (2017). Extremely large minibatch sgd: Training resnet-50 on imagenet in 15 minutes. arXiv preprint arXiv:1711.04325.

Akin, K. D., Gurkan, C., Budak, A., & Karataş, H. (2022). Classification of monkeypox skin lesion using the explainable artificial intelligence assisted convolutional neural networks. *Avrupa Bilim ve Teknoloji Dergisi*, (40), 106–110.

Alakunle, E., Moens, U., Nchinda, G., & Okeke, M. I. (2020). Monkeypox virus in Nigeria: infection biology, epidemiology, and evolution. *Viruses*, *12*(11), 1257.

Alam, M. S., Rashid, M. M., Roy, R., Faizabadi, A. R., Gupta, K. D., & Ahsan, M. M. (2022). Empirical study of autism spectrum disorder diagnosis using facial images by improved transfer learning approach. *Bioengineering*, *9*(11), 710.

Alcalá-Rmz, V., Villagrana-Bañuelos, K. E., Celaya-Padilla, J. M., Galván-Tejada, J. I., Gamboa-Rosales, H., & Galván-Tejada, C. E. (2023). Convolutional neural network for monkeypox detection. In *International conference on ubiquitous computing and ambient intelligence* (pp. 89–100). Springer.

Ali, S. N., Ahmed, M., Paul, J., Jahan, T., Sani, S., Noor, N., et al. (2022). Monkeypox skin lesion detection using deep learning models: A feasibility study. arXiv preprint arXiv:2207.03342.

Amari, S.-i. (1993). Backpropagation and stochastic gradient descent method. *Neurocomputing*, *5*(4–5), 185–196.

Anil (2021). Limitations of graph neural networks. (accessed on November 20, 2022). https://wandb.ai/syllogismos/machine-learning-with-graphs/reports/18-Limitations-of-Graph-Neural-Networks--VmlldzozODUxMzQ.

Bala, D. (2022). Monkeypox skin images dataset. (accessed on November 22, 2022). https://www.kaggle.com/datasets/dipuiucse/monkeypoxskinimagedataset.

Banan, A., Nasiri, A., & Taheri-Garavand, A. (2020). Deep learning-based appearance features extraction for automated carp species identification. *Aquacultural Engineering*, *89*, Article 102053.

Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, *13*(2).

Beyer, L., Hénaff, O. J., Kolesnikov, A., Zhai, X., & Oord, A. v. d. (2020). Are we done with imagenet? arXiv preprint arXiv:2006.07159.

Bhattiprolu, S. (2020). Data augmentation. (accessed on may 10, 2022). https://github.com/bnsreenu.

Bragazzi, N. L., Khamisy-Farah, R., Tsigalou, C., Mahroum, N., & Converti, M. (2022). Attaching a stigma to the LGBTQI+ community should be avoided during the monkeypox epidemic. *Journal of Medical Virology*.

Brown, O., Curtis, A., & Goodwin, J. (2021). Principles for evaluation of AI/ML model performance and robustness. arXiv preprint arXiv:2107.02868.

CDC (2022a). 2022 Monkeypox and orthopoxvirus outbreak global map. accessed on June 10, 2022. https://www.cdc.gov/poxvirus/monkeypox/response/2022/world-map.html.

CDC (2022b). Monkeypox and smallpox vaccine. (accessed on May 30, 2022). https://www.cdc.gov/poxvirus/monkeypox/clinicians/treatment.html.

CDC (2022c). Monkeypox signs and symptoms. (accessed on May 30, 2022). https://www.cdc.gov/poxvirus/monkeypox/symptoms.html.

Celaya-Padilla, J. M., Galván-Tejada, J. I., Gamboa-Rosales, H., & Galván-Tejada, C. E. (2022). Convolutional neural network for monkeypox detection. In *Proceedings of the international conference on ubiquitous computing & ambient intelligence (UCAmI 2022), Vol. 594* (p. 89). Springer Nature.

Chattopadhay, A., Sarkar, A., Howlader, P., & Balasubramanian, V. N. (2018). Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks. In *2018 IEEE winter conference on applications of computer vision* WACV, (pp. 839–847). IEEE.

Cian, D., van Gemert, J., & Lengyel, A. (2020). Evaluating the performance of the LIME and grad-CAM explanation methods on a LEGO multi-label image classification task. arXiv preprint arXiv:2008.01584.

Cordoş, C., Mihailă, L., Faragó, P., & Hintea, S. (2021). ECG signal classification using convolutional neural networks for biometric identification. In *2021 44th international conference on telecommunications and signal processing (TSP)* (pp. 167–170). IEEE.

Corneanu, C., Madadi, M., Escalera, S., & Martinez, A. (2020). Explainable early stopping for action unit recognition. In *2020 15th IEEE international conference on automatic face and gesture recognition (FG 2020)* (pp. 693–699). IEEE.

Dauphin, Y., De Vries, H., & Bengio, Y. (2015). Equilibrated adaptive learning rates for non-convex optimization. *Advances in Neural Information Processing Systems*, *28*.

Dey, S., Nath, P., Biswas, S., Nath, S., & Ganguly, A. (2021). Malaria detection through digital microscopic imaging using deep greedy network with transfer learning. *Journal of Medical Imaging*, *8*(5), Article 054502.

Dogo, E., Afolabi, O., Nwulu, N., Twala, B., & Aigbavboa, C. (2018). A comparative analysis of gradient descent-based optimization algorithms on convolutional neural networks. In *2018 international conference on computational techniques, electronics and mechanical systems* CTEMS, (pp. 92–99). IEEE.

Doucleff, M. (2022). Scientists warned us about monkeypox in 1988. Here's why they were right. (accessed on May 27, 2022). https://www.npr.org/sections/goatsandsoda/2022/05/27/1101751627/scientists-warned-us-about-monkeypox-in-1988-heres-why-they-were-right.

Eid, M. M., El-Kenawy, E.-S. M., Khodadadi, N., Mirjalili, S., Khodadadi, E., Abotaleb, M., et al. (2022). Meta-heuristic optimization of LSTM-based deep network for boosting the prediction of monkeypox cases. *Mathematics*, *10*(20), 3845.

Fan, Y., Xu, K., Wu, H., Zheng, Y., & Tao, B. (2020). Spatiotemporal modeling for nonlinear distributed thermal processes based on KL decomposition, MLP and LSTM network. *IEEE Access*, *8*, 25111–25121.

Gao, F., Wu, T., Li, J., Zheng, B., Ruan, L., Shang, D., et al. (2018). SD-CNN: A shallow-deep CNN for improved breast cancer diagnosis. *Computerized Medical Imaging and Graphics*, *70*, 53–62.

Garreau, D., & Mardaoui, D. (2021). What does LIME really see in images? In *International conference on machine learning* (pp. 3620–3629). PMLR.

Ghalebikesabi, S. (2022). Model-agnostic local explanation models from a statistical viewpoint. accessed on June 02, 2022. https://towardsdatascience.com/model-agnostic-local-explanation-models-from-a-statistical-viewpoint-i-bd04039c7040.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., et al. (2020). Generative adversarial networks. *Communications of the ACM*, *63*(11), 139–144.

Haque, M., Ahmed, M., Nila, R. S., Islam, S., et al. (2022). Classification of human monkeypox disease using deep learning models and attention mechanisms. arXiv preprint arXiv:2211.15459.

Haque, R., Islam, N., Islam, M., & Ahsan, M. M. (2022). A comparative analysis on suicidal ideation detection using NLP, machine, and deep learning. *Technologies*, *10*(3), 57.

He, K., Chen, X., Xie, S., Li, Y., Dollár, P., & Girshick, R. (2022). Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 16000–16009).

He, K., Zhang, X., Ren, S., & Sun, J. (2016a). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770–778).

He, K., Zhang, X., Ren, S., & Sun, J. (2016b). Identity mappings in deep residual networks. In *European conference on computer vision* (pp. 630–645). Springer.

Hobbhahn, M. (2021). How to measure FLOP/s for neural networks empirically? (accessed on November 22, 2022). https://www.lesswrong.com/posts/jJApGWG95495pYM7C/how-to-measure-flop-s-for-neural-networks-empirically.

Hu, F., & Li, H. (2013). A novel boundary oversampling algorithm based on neighborhood rough set model: NRSBoundary-SMOTE. *Mathematical Problems in Engineering*, *2013*.

Irmak, M. C., Aydin, T., & Yağanoğlu, M. (2022). Monkeypox skin lesion detection with MobileNetV2 and VGGNet models. In *2022 medical technologies congress* TIPTEKNO, (pp. 1–4). IEEE.

Islam, T., Hussain, M. A., Chowdhury, F. U. H., & Islam, B. R. (2022). Can artificial intelligence detect monkeypox from digital skin images? *BioRxiv*.

Islam, A., & Shin, S. Y. (2022). A blockchain-based privacy sensitive data acquisition scheme during pandemic through the facilitation of federated learning. In *2022 13th international conference on information and communication technology convergence* ICTC, (pp. 83–87). IEEE.

ISU (2022). Diagnostic tests. (accessed on may 30, 2022). https://www.nj.gov/agriculture/divisions/ah/diseases/monkeypox.html.

Jiao, Y., Deng, Y., Luo, Y., & Lu, B.-L. (2020). Driver sleepiness detection from EEG and EOG signals using GAN and LSTM networks. *Neurocomputing*, *408*, 100–111.

Jin, Z., & Finkel, H. (2020). Analyzing deep learning model inferences for image classification using OpenVINO. In *2020 IEEE international parallel and distributed processing symposium workshops* IPDPSW, (pp. 908–911). IEEE.

Jing, Y., Yang, Y., Feng, Z., Ye, J., Yu, Y., & Song, M. (2019). Neural style transfer: A review. *IEEE Transactions on Visualization and Computer Graphics*, *26*(11), 3365–3385.

Kawaguchi, K., Kaelbling, L. P., & Bengio, Y. (2017). Generalization in deep learning. arXiv preprint arXiv:1710.05468.

Kermany, D. S., Goldbaum, M., Cai, W., Valentim, C. C., Liang, H., Baxter, S. L., et al. (2018). Identifying medical diagnoses and treatable diseases by image-based deep learning. *Cell*, *172*(5), 1122–1131.

Khodakevich, L., Ježek, Z., & Messinger, D. (1988). Monkeypox virus: ecology and public health significance. *Bulletin of the World Health Organization*, *66*(6), 747.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Kukkar, A., Gupta, D., Beram, S. M., Soni, M., Singh, N. K., Sharma, A., et al. (2022). Optimizing deep learning model parameters using socially implemented IoMT systems for diabetic retinopathy classification problem. *IEEE Transactions on Computational Social Systems*.

Li, A., Xiao, F., Zhang, C., & Fan, C. (2021). Attention-based interpretable neural network for building cooling load prediction. *Applied Energy*, *299*, Article 117238.

Lin, H., Gharehbaghi, A., Zhang, Q., Band, S. S., Pai, H. T., Chau, K.-W., et al. (2022). Time series-based groundwater level forecasting using gated recurrent unit deep neural networks. *Engineering Applications of Computational Fluid Mechanics*, *16*(1), 1655–1672.

Liu, S., Papailiopoulos, D., & Achlioptas, D. (2020). Bad global minima exist and sgd can reach them. *Advances in Neural Information Processing Systems*, *33*, 8543–8552.

Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems*, *30*.

McCollum, A. M., & Damon, I. K. (2014). Human monkeypox. *Clinical Infectious Diseases*, *58*(2), 260–267.

Mehrotra, R., Ansari, M., Agrawal, R., & Anand, R. (2020). A transfer learning approach for AI-based classification of brain tumors. *Machine Learning with Applications*, *2*, Article 100003.

Menzies, T., Greenwald, J., & Frank, A. (2006). Data mining static code attributes to learn defect predictors. *IEEE Transactions on Software Engineering*, *33*(1), 2–13.

Mohanty, S. P., Hughes, D. P., & Salathé, M. (2016). Using deep learning for image-based plant disease detection. *Frontiers in Plant Science*, *7*, 1419.

Moore, M., & Zahra, F. (2022). Monkeypox. (accessed on May 22, 2022). https://www.ncbi.nlm.nih.gov/books/NBK574519/.

Narin, A., Kaya, C., & Pamuk, Z. (2021). Automatic detection of coronavirus disease (covid-19) using x-ray images and deep convolutional neural networks. *Pattern Analysis and Applications*, *24*(3), 1207–1220.

Nguyen, P.-Y., Ajisegiri, W. S., Costantino, V., Chughtai, A. A., & MacIntyre, C. R. (2021). Reemergence of human monkeypox and declining population immunity in the context of urbanization, Nigeria, 2017–2020. *Emerging Infectious Diseases*, *27*(4), 1007.

Nguyen, H.-P., Luu, T.-N., Le, N.-B., Vo, V.-T., Huynh, N.-T., Phan, Q.-H., et al. (2022). Combined mueller matrix imaging and artificial intelligence classification framework for Hepatitis B detection. *Journal of Biomedical Optics*, *27*(7), Article 075002.

Nolen, L. D., Osadebe, L., Katomba, J., Likofata, J., Mukadi, D., Monroe, B., et al. (2016). Extended human-to-human transmission during a monkeypox outbreak in the Democratic Republic of the Congo. *Emerging Infectious Diseases*, *22*(6), 1014.

Okte, E., & Al-Qadi, I. L. (2021). Prediction of flexible pavement 3-D finite element responses using Bayesian neural networks. *International Journal of Pavement Engineering*, 1–11.

Organization, W. H., et al. (2021). *Ethics and governance of artificial intelligence for health: WHO guidance*. Who.

Pan, P., Li, Y., Xiao, Y., Han, B., Su, L., Su, M., et al. (2020). Prognostic assessment of COVID-19 in the intensive care unit by machine learning methods: model development and validation. *Journal of Medical Internet Research*, *22*(11), Article e23128.

Park, A. (2022). There's already a monkeypox vaccine. But not everyone may need it. (accessed on May 27, 2022). https://time.com/6179429/monkeypox-vaccine/.

Peer, D., Stabinger, S., & Rodriguez-Sanchez, A. (2021). Limitation of capsule networks. *Pattern Recognition Letters*, *144*, 68–74.

Popescu, M. C., & Sasu, L. M. (2014). Feature extraction, feature selection and machine learning for image classification: A case study. In *2014 international conference on optimization of electrical and electronic equipment* OPTIM, (pp. 968–973). IEEE.

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016a). " Why should i trust you?" Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 1135–1144).

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016b). Model-agnostic interpretability of machine learning. arXiv preprint arXiv:1606.05386.

Roy, K., Chaudhuri, S. S., Ghosh, S., Dutta, S. K., Chakraborty, P., & Sarkar, R. (2019). Skin disease detection based on different segmentation techniques. In *2019 international conference on opto-electronics and applied optics (Optronix)* (pp. 1–5). IEEE.

Sagar, A. (2019). 5 techniques to prevent overfitting in neural networks. (accessed on Nov 20, 2022). https://www.kdnuggets.com/2019/12/5-techniques-prevent-overfitting-neural-networks.html.

Sahin, V. H., Oztel, I., & Yolcu Oztel, G. (2022). Human monkeypox classification from skin lesion images with deep pre-trained network using mobile application. *Journal of Medical Systems*, *46*(11), 1–10.

Sandeep, R., Vishal, K., Shamanth, M., & Chethan, K. (2022). Diagnosis of visible diseases using CNNs. In *Proceedings of international conference on communication and artificial intelligence* (pp. 459–468). Springer.

Sarmad, M., Lee, H. J., & Kim, Y. M. (2019). Rl-gan-net: A reinforcement learning agent controlled gan network for real-time point cloud shape completion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 5898–5907).

Selvaraju, R. R., Das, A., Vedantam, R., Cogswell, M., Parikh, D., & Batra, D. (2016). Grad-CAM: Why did you say that? arXiv preprint arXiv:1611.07450.

Shah, D. (2022). The essential guide to data augmentation in deep learning. (accessed on Nov 20, 2022). https://www.kdnuggets.com/2019/12/5-techniques-prevent-overfitting-neural-networks.html.

Sharma, N., Vijayeendra, A., Gopakumar, V., Patni, P., & Bhat, A. (2022). Automatic identification of bird species using audio/video processing. In *2022 international conference for advancement in technology* ICONAT, (pp. 1–6). IEEE.

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

Sitaula, C., & Shahi, T. B. (2022). Monkeypox virus detection using pre-trained deep learning-based approaches. *Journal of Medical Systems*, *46*(11), 1–9.

Stolfo, S. J., Fan, W., Lee, W., Prodromidis, A., & Chan, P. K. (2000). Cost-based modeling for fraud and intrusion detection: Results from the JAM project. In *Proceedings DARPA information survivability conference and exposition. DISCEX'00, Vol. 2* (pp. 130–144). IEEE.

Studer, L., Alberti, M., Pondenkandath, V., Goktepe, P., Kolonko, T., Fischer, A., et al. (2019). A comprehensive study of imagenet pre-training for historical document image analysis. In *2019 international conference on document analysis and recognition* ICDAR, (pp. 720–725). IEEE.

Sutskever, I., Martens, J., Dahl, G., & Hinton, G. (2013). On the importance of initialization and momentum in deep learning. In *International conference on machine learning* (pp. 1139–1147). PMLR.

Tan, M., & Le, Q. (2019). Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning* (pp. 6105–6114). PMLR.

Tan, M., & Le, Q. (2021). Efficientnetv2: Smaller models and faster training. In *International conference on machine learning* (pp. 10096–10106). PMLR.

Tensorflow (2022). ImageDataGenerator. (accessed on may 10, 2022). https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator.

Varshni, D., Thakral, K., Agarwal, L., Nijhawan, R., & Mittal, A. (2019). Pneumonia detection using CNN based feature extraction. In *2019 IEEE international conference on electrical, computer and communication technologies* ICECCT, (pp. 1–7). IEEE.

Velasco, J., Pascion, C., Alberio, J. W., Apuang, J., Cruz, J. S., Gomez, M. A., et al. (2019). A smartphone-based skin disease classification using mobilenet cnn. arXiv preprint arXiv:1911.07929.

Vijayalakshmi, A., et al. (2020). Deep learning approach to detect malaria from microscopic images. *Multimedia Tools and Applications*, *79*(21), 15297–15317.

Wang, J., Li, X., Li, J., Sun, Q., & Wang, H. (2022). Ngcu: A new rnn model for time-series data prediction. *Big Data Research*, *27*, Article 100296.

Wang, L., Lin, Z. Q., & Wong, A. (2020). Covid-net: A tailored deep convolutional neural network design for detection of covid-19 cases from chest x-ray images. *Scientific Reports*, *10*(1), 1–12.

WHO (2022). Multi-country monkeypox outbreak in non-endemic countries. (accessed on may 29, 2022). https://www.who.int/emergencies/disease-outbreak-news/item/2022-DON385.

Zhang, Z. (2018). Improved adam optimizer for deep neural networks. In *2018 IEEE/ACM 26th international symposium on quality of service (IWQoS)* (pp. 1–2). Ieee.

Zhang, A., Ballas, N., & Pineau, J. (2018). A dissection of overfitting and generalization in continuous reinforcement learning. arXiv preprint arXiv:1806.07937.

Zhang, Y., & Davison, B. D. (2020). Impact of imagenet model selection on domain adaptation. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision workshops* (pp. 173–182).

Zhang, C., Liao, Q., Rakhlin, A., Miranda, B., Golowich, N., & Poggio, T. (2018). Theory of deep learning IIb: Optimization properties of SGD. arXiv preprint arXiv:1801.02254.

Zhang, G., Wang, C., Xu, B., & Grosse, R. (2018). Three mechanisms of weight decay regularization. arXiv preprint arXiv:1810.12281.