Journal of Medical Systems (2022) 46:79
https://doi.org/10.1007/s10916-022-01863-7

ORIGINAL PAPER

# Human Monkeypox Classification from Skin Lesion Images with Deep Pre-trained Network using Mobile Application

**Veysel Harun Sahin**[1] · **Ismail Oztel**[2] · **Gozde Yolcu Oztel**[1]

Received: 23 July 2022 / Accepted: 7 September 2022 / Published online: 10 October 2022


## Abstract

Recently, human monkeypox outbreaks have been reported in many countries. According to the reports and studies, quick determination and isolation of infected people are essential to reduce the spread rate. This study presents an Android mobile application that uses deep learning to assist this situation. The application has been developed with Android Studio using Java programming language and Android SDK 12. Video images gathered through the mobile device's camera are dispatched to a deep convolutional neural network that runs on the same device. Camera2 API of the Android platform has been used for camera access and operations. The network then classifies images as positive or negative for monkeypox detection. The network's training has been carried out using skin lesion images of monkeypox-infected people and other skin lesion images. For this purpose, a publicly available dataset and a deep transfer learning approach have been used. All training and testing steps have been applied on Matlab using different pre-trained networks. Then, the network that has the best accuracy has been recreated and trained using TensorFlow. The TensorFlow model has been adapted to mobile devices by converting to the TensorFlow Lite model. The TensorFlow Lite model has been then embedded into the mobile application together with the TensorFlow Lite library for monkeypox detection. The application has been run on three devices successfully. During the run-time, the inference times have been gathered. 197 ms, 91 ms, and 138 ms average inference times have been observed. The presented system allows people with body lesions to quickly make a preliminary diagnosis. Thus, monkeypox-infected people can be encouraged to act rapidly to see an expert for a definitive diagnosis. According to the test results, the system can classify the images with 91.11% accuracy. In addition, the proposed mobile application can be trained for the preliminary diagnosis of other skin diseases.

**Keywords** Android · Artificial Intelligence · Mobile Application · Monkeypox · TensorFlow Lite · Deep Learning

## Introduction

Monkeypox is a zoonotic disease and an orthopoxvirus causes it [1]. It shows signs of a smallpox-like disease in humans. Monkeypox in humans first appeared in 1970 in the Democratic Republic of the Congo (DRC), then spread to other regions of Africa. Recently many cases have been diagnosed outside Africa [2] and an outbreak of monkeypox infection has been quickly spreading worldwide [3]. According to [2], there is an increment of monkeypox cases, specifically in the endemic DRC, and a growing median age from young children to young adults. On July 23, 2022, the global monkeypox outbreak was announced as a public health emergency of international concern (PHEIC) by WHO [1]. A PHEIC is the highest level of alert that the UN health body can give. The same alert was given for covid-19, polio, the 2014 outbreak of Ebola, and the spread of the Zika virus in 2016 [4]. According to the WHO [1], from 1 January to 22 June 2022, 3413 laboratory-confirmed cases and one death have been reported to WHO from 5 countries/territories in five WHO Regions.

Monkeypox symptoms last from 2 to 4 weeks and severe cases can occur. According to a WHO report, the recent

✉ Veysel Harun Sahin
vsahin@sakarya.edu.tr

Ismail Oztel
ioztel@sakarya.edu.tr

Gozde Yolcu Oztel
gyolcu@sakarya.edu.tr

[1] Software Engineering Department, Sakarya University, Sakarya 54050, Turkey

[2] Computer Engineering Department, Sakarya University, Sakarya 54050, Turkey

case fatality ratio has been around 3% and 6%. While the incubation period of monkeypox is usually from 6 to 13 days, it can range from 5 to 21 days. The infection is observed in two periods. During the invasion period, the patients are observed with back pain, fever, swelling of the lymph nodes, intense headache muscle aches, and lack of energy. In the second period, within 1-3 days of fever, the skin lesions begin. In 95% of cases, the face is affected. In 75% of them, the palms of the hands and soles of the feet are affected. Moreover, in 70% of cases oral mucous membranes are affected, in 30% of them genitalia, and in 20% of them conjunctivae, as well as cornea [1].

The virus is usually transmitted from one person to another by close contact or contact with contaminated materials such as bedding, clothing, etc. [1]. According to [5], detection of more cases is anticipated. According to [6], the Polymerase Chain Reaction (PCR) and some other biochemical tests are not readily available in sufficient amounts.

WHO recommends investigating a patient with suspected monkeypox. If confirmed, isolation is recommended until the lesions have crusted, the scab has fallen off, and a fresh layer of skin has formed underneath [1]. Therefore, rapid detection of infected individuals is required to reduce the spread.

In recent years, artificial intelligence applications have been frequently used in the field of health as well as in many areas such as biometric recognition [7], commerce [8] and social media [9]. From the viewpoint of health, it assists specialists in pre-diagnosis, counting lesions [10] or mitochondria [11], etc. In particular, visual detection studies produce successful results owing to deep learning techniques, a sub-branch of artificial intelligence.

Today, mobile devices have become ubiquitous [12–15]. They have penetrated our lives in many ways, like smartphones, tablets, and smartwatches. Their capabilities and functionalities have grown over time. Mobile devices can get information from real life through microphone, camera and several different sensors. They have powerful central processing units (CPUs), sufficiently good memory, and large permanent storage unit. Nowadays, we use mobile devices for various needs like entertainment [16], reading [17], solving engineering problems [18], and tracking personal health [19]. They became a part of the life of society.

This study presents an Android mobile application to help reduce the rate of monkeypox spread. The application analyzes the skin lesions with the help of a smartphone camera and indicates the possibility of a monkeypox. People with body lesions and suspected monkeypox can make a preliminary diagnosis using the presented application. Thus, monkeypox-infected people can be encouraged to act more quickly to see an expert for a definitive diagnosis. Eventually, it can be ensured that infected people are isolated more quickly.

The contributions of the study are summarized in the items listed below.

1. A basic, cheap, non-invasive disease diagnosis tool has been developed as a mobile application. To the best of the authors' knowledge, no other study has been found that detects monkeypox in humans using a mobile application.
2. Introduced a low-modified MobilNetV2 model for detection of Monkeypox with skin lesion image data using a mobile application.

## Related works

### Human monkeypox investigation with computer vision techniques

The first human case of monkeypox appeared in 1970. Thus, the human monkeypox studies in the literature date back to the 1970s [20–22]. The research on human monkeypox has recently increased as the outbreak of monkeypox infection has been quickly spreading worldwide. Recently, some researchers [3, 23] have mentioned in their studies that more research is needed on this topic.

Although the human monkeypox disease dates back to ancient times, computer vision based studies for preleminary diagnosis of the disease have recently begun. There are very few studies on it currently. In [24], the authors collected monkeypox-infected image data from Google and analyzed the data with deep learning methods. They used modified VGG16 network for this purpose. In [6] the authors created a human monkeypox image database and classified them. For classification, they used four deep learining networks. These are VGG16, ResNet50, InceptionV3 and Ensemble. Their results are reported in Experimental results section in this study (Table 5), and compared with proposed system results.

Apart from the studies mentioned above, as a contribution to the literature, we propose a mobile application that classifies skin lesions of patients as monkeypox-infected vs. non-infected.

### Mobile applications for identification of skin lesion images

Smartphones are becoming increasingly significant in monitoring and delivery of healthcare. With the help of mobile healthcare applications, people can quickly get information about their diseases and can be followed by their therapists. In addition, this task does not need additional device costs. The patients can receive various health services from the mobile device they are already accustomed to [25].

There are variety of mobile applications to improve healthcare [25–28]. Some of them are developed for identification of skin lesions. In [29, 30], the authors developed a mobile system to diagnose skin cancer lesions. They used deep learning in their studies. In [31], the authors presented a mobile application that acquires and identifies moles in skin images and classifies them according to their severity into melanoma, nevus, and benign lesion. In [32], the authors developed a image-processing based smartphone application for psoriasis segmentation and classification. They used image processing and computer vision techniques.

## Methodology

The overall system flow is illustrated in Fig. 1. First, the human monkeypox lesion images have been prepared for training using image preprocessing methods. In this stage, we have normalized the pixel values from [0, 255] to [-1, 1]. Then, the system has been trained with the transfer learning method using pre-trained networks. Various pre-trained networks have been used and their performances have been compared in this step. The training and testing steps have been applied on Matlab 2022a. Afterward, the network with the best performance was recreated using TensorFlow. The TensorFlow model has been converted to the TensorFlow Lite model [33, 34] for use on mobile and edge devices. Lastly, an Android mobile application has been developed to detect monkeypox using the TensorFlow Lite model. When launched, the mobile application automatically opens the device's camera and takes the real-time video stream. It extracts image frames from the stream, classifies them, and outputs the detection as positive or negative.

### Image classification using deep transfer learning

Deep learning approaches allow automatic learning of complex features needed for visual pattern recognition. Convolutional Neural Networks (CNN) is a type of deep learning approach. CNN has been used for several computer vision tasks such as facial expression recognition [35, 36], text recognition [37], face recognition [38], gender classification [39], age classification [40], and action recognition [41]. CNN has also recently shown outstanding performance in biomedical applications, including pattern recognition and computer vision [42–45].

CNNs are structured with different combinations of convolution, pooling, and fully connected layers. In the convolution layer, filters are convolved with the input data. The convolution process is shown in Eq. (1) [46].

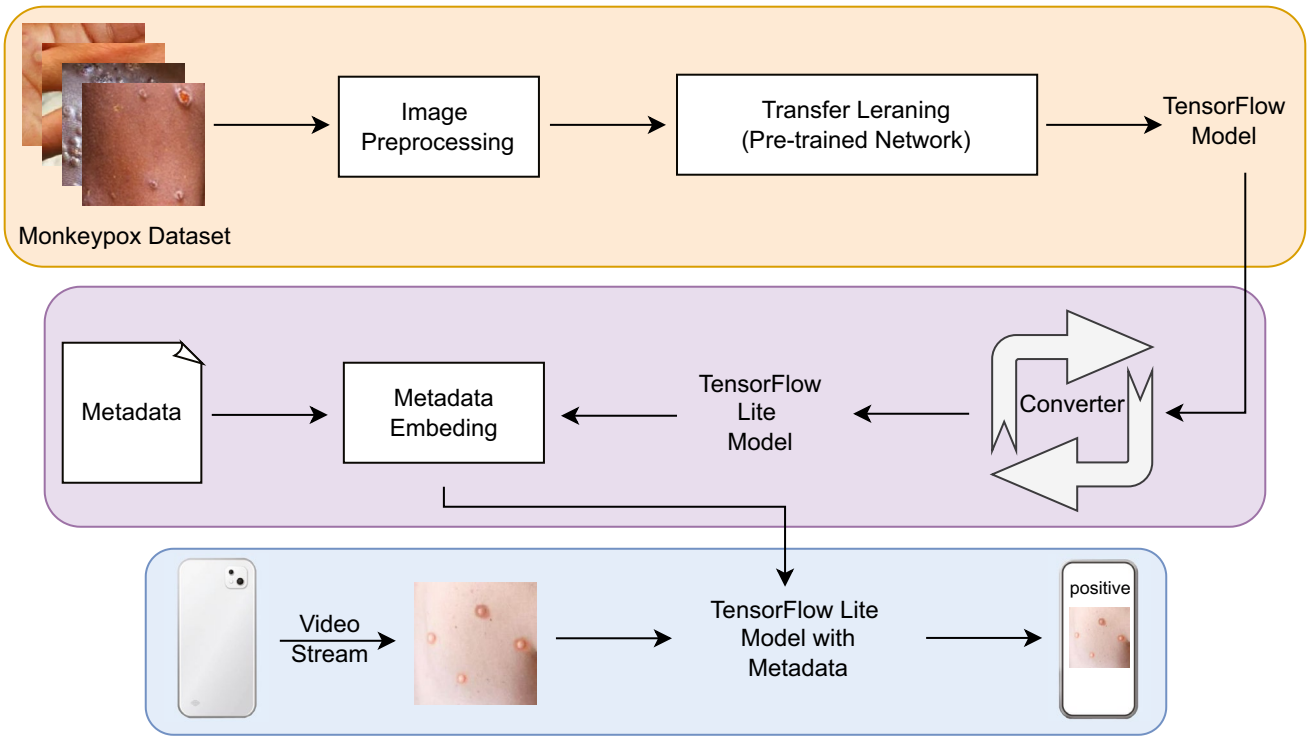


**Fig. 1** A perspective of the proposed method

$$S(i,j) = (I * K)(i,j) = \sum_{m}\sum_{n} I(i+m, j+n)K(m,n) \quad (1)$$

In the equation, $I$ represents an input image with two dimensions, $K$ represents a two-dimensional kernel, $S$ represents the two-dimensional output after the convolution process. $(i,j)$ are matrix indexes and $(m,n)$ are filter sizes.

Convolutional layers are often followed by a non-linear activation layer. ReLu activation function has been used in this study. The pooling layer reduces the input size in width and height. The dropout layer is used to prevent overfitting in the CNNs. The fully connected layer is structured just before the classification layer. It connects to all nodes in the previous layer.

After the CNN model design, the next step is determining filter numbers, the value of strides, and dimensions. Then, the network is trained using the feed-forward strategy. Feed-forward means information is taken in the first layer and forwarded through the following layers. In the last layer, the error value is calculated. In this calculation, the produced result by the network and the target result are used. In this study, cross-entropy loss function shown in Eq. (2) [47] has been used.

$$loss = -\frac{1}{N}\sum_{n=1}^{N}\sum_{i=1}^{K} T_{ni} ln Y_{ni} \quad (2)$$

In the equation, $N$ and $K$ are the number of observations and classes, respectively. $Y$ is the input, and $T$ is the target.

In deep learning, instead of designing a network from scratch, a network prepared for a task can be used for another purpose. That is called transfer learning. Transfer learning takes features learned on one problem and leverages them on a new problem [48]. For example, a network that learns features of object detection can be used for a disease diagnosis. In this study, different pre-trained networks have been used for the human monkeypox classification task. For this purpose, some modifications have been made in the last layers of the network. The last layer with learnable weights in the networks is a fully connected layer. In this study, these fully connected layers have been replaced with a new fully connected layer with 2 outputs. Thus, the networks were prepared for the classification.

The used pre-trained networks and their properties are given in Table 1. The main purpose of this study is to develop a mobile application for human monkeypox classification. Thus, we select low-size and parameter models to run speedily on mobile devices.

**Table 1** Details of the pre-trained networks used as feature extractor

| Network Name | Depth | Size (MB) | Parameters | Input Size |
|---|---|---|---|---|
| ResNet18 | 18 | 44 | 11.7 | 224x224 |
| GoogleNet | 22 | 27 | 7.0 | 224x224 |
| EfficientNetb0 | 82 | 20 | 5.3 | 224x224 |
| NasnetMobile | * | 20 | 5.3 | 224x224 |
| ShuffleNet | 50 | 5.4 | 1.4 | 224x224 |
| MobileNetv2 | 53 | 13 | 3.5 | 224x224 |

## Mobile application for human monkeypox detection

The first step in application development is to create a problem statement. The problem statement of this study is the need for an application to detect monkeypox lesions and inform the user with the help of a mobile device and its camera. According to this problem statement, three main requirements emerge.

- The users should be able to detect the lesion using a mobile device.
- The video stream from the mobile device's camera should be gathered and provided both to the user and the subsystem that performs prediction.
- The application should perform prediction, and the users should see the prediction results on the device's screen.

The authors chose to use smartphones with Android operating system (OS) [49] to meet the first requirement. Android OS is a mobile platform that powers devices like smartphones, tablets, smartwatches, car infotainment systems, and smart televisions. Therefore, the development of an Android mobile application has been decided. There are three main ways to develop an Android application. The first one is to use the tools of the Android ecosystem [50] and develop a native Android mobile application solely for the Android platform. The second one is to use a multi-platform framework like React Native [51], Flutter [52], Kotlin Multiplatform Mobile (KMM) [53] and develop an application that targets multiple platforms. The third option is to use Progressive Web Applications (PWAs) [54] again for the development of multi-platform mobile applications. The authors decided to utilize the Android ecosystem tools for this study. Therefore, the development has been carried out using Android Studio (Chipmunk 2021.2.1) integrated development environment (IDE). To develop an Android application, an Android standard development kit (SDK) is needed with Android Studio. In this study, Android 12 SDK, in other words, application programming interface (API) level 31 has been used. The Android platform supports multiple languages like Java and Kotlin. In this study, Java programming language has been chosen.

**Table 2** Design decisions for the mobile application

| Feature Type | Feature Name |
|---|---|
| Mobile Device | Smartphone |
| Operating System | Android OS |
| IDE | Android Studio Chipmunk 2021.2.1 |
| SDK | Android 12 SDK (API level 31) |
| Minimum SDK | Android 5.0 SDK (API level 21) |
| Programming Language | Java |
| Camera API | Camera2 API |
| ML Library | TensorFlow Lite |

The camera of the Android smartphone should be accessed, and the frames of the video stream should be gathered to meet the second requirement. For camera applications, the Android platform provides two APIs: Camera2 API and CameraX API. The authors chose to use Camera2 API. The minimum requirement for Camera2 API is Android 5.0 SDK (API level 21). That means the developed application can run on smartphones that have Android 5.0 Lollipop and higher.

For the third requirement, a machine learning (ML) library compatible with mobile and edge devices is needed. Today, ML tasks can be handled using mobile applications. One way of achieving this is getting help from TensorFlow Lite [33, 34]. The TensorFlow Lite is a library that helps to run ML models on mobile and edge devices. It is optimized to handle machine learning tasks on resource constraint mobile and edge devices. Currently, the TensorFlow Lite binary is approximately 1MB in size when all supported operators are linked for 32-bit ARM builds. It supports Android, iOS, and embedded Linux devices. It also supports microcontrollers. The TensorFlow Lite library can use the hardware accelerator on the mobile device, if any, by getting help from the Android Neural Networks API (NNAPI). There is multiple programming language support. The library can be used with Java, Python, C++, Objective-C, and Swift programming languages.

In this study, the authors used the TensorFlow Lite library to develop a mobile application that runs on Android devices. The development has been carried out using Java programming language.

The design decisions regarding the development are shown in Table 2.

TensorFlow Lite uses its specific ML model format. This format is small, portable, and can also be optimized for fast execution. Any ML model that is prepared elsewhere must be converted to this format for deployment on Android devices. In this study, the design and development of the deep learning network have been done on Matlab. Afterward, the same network was realized using TensorFlow [33] and TensorFlow model was created. Then, the TensorFlow model was converted to the TensorFlow Lite model and embedded into the Android application.

The flow of ML tasks on an Android application is shown in Fig. 2. The mobile application first takes the input data for prediction. Then the data is converted to a different format for use by TensorFlow Lite. This format is called a tensor. A tensor is a multidimensional NumPy [55] array. Then, the tensor is given to the TensorFlow Lite runtime. The TensorFlow Lite library provides this runtime for the execution of ML models. The runtime executes the ML model with the provided tensor(s) and produces output as tensor(s). In other words, it makes a prediction. In the TensorFlow Lite context, this phase is called inference. Lastly, these tensors are interpreted as prediction results.

The TensorFlow Lite library provides two APIs: TensorFlow Lite Task API and TensorFlow Lite Interpreter API. Both of them can be used for ML tasks in Android applications. The TensorFlow Lite Interpreter API is a low-level API. The TensorFlow Lite Task API is a high-level API that wraps the functionality of the TensorFlow Lite Interpreter API. In this study, TensorFlow Lite Task API has been used. From the user's point of view, one of the differences between these two APIs is the metadata requirement. TensorFlow Lite Task API requires embedding the model metadata to the TensorFlow Lite model. That is shown in Fig. 1.

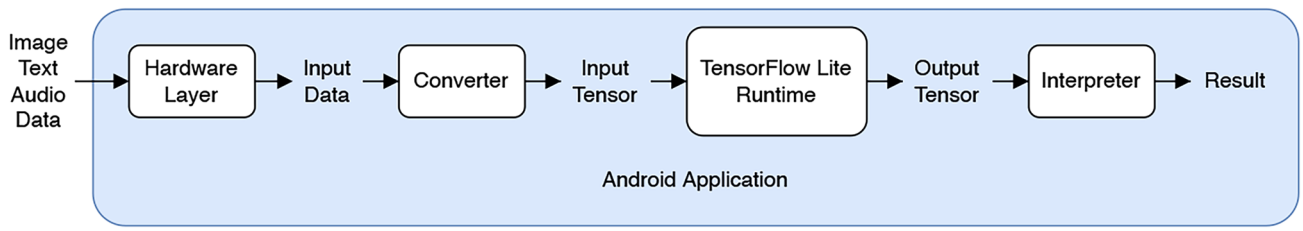


**Fig. 2** The flow of ML tasks on Android application

## Experimental results

### Dataset and data preparation

For human monkeypox classification task, Monkeypox Skin Lesion Dataset (MSLD) [6] has been used. This dataset has been obtained from the Kaggle website [56]. It includes binary classification data for monkeypox vs. non-monkeypox. The non-monkeypox lesions are either chickenpox or measles. These lesions are similar to monkeypox lesions.

The database was prepared for computer-aided monkeypox detection from skin lesion images. MSLD was created by collecting and processing images using web-scrapping such as news portals, publicly accessible case reports, and websites. The database consists of three folders. These are original images, augmented images, and Fold1. The *original images* folder includes 228 images; 102 belong to monkeypox class, and 126 belong to non-monkeypox class. Varied data augmentation methods were applied to original images like hue, saturation, contrast, rotation, translation, reflection, shear, brightness jitter, noise, and scaling. Thus, the image number increased to 1428 and 1764 for monkeypox and non-monkeypox, respectively. These images were stored in the *augmented images* folder. The original images were divided into training, validation, and test sets with the approximate ratio of 70:10:20. Patient independence was maintained while doing this. The data augmentation was only applied to the validation and training sets. These datasets were stored in *Fold1* folder. In the proposed study, Fold1 has been used for the training, validation, and testing phase.

### Experiments on image classification

Various pre-trained networks have been modified for the human monkeypox classification task. All networks have been trained with transfer learning up to 60 epochs. These networks were originally trained to classify 1000 object categories from the ImageNet database [57]. On the other hand, the proposed method considers monkeypox detection as a binary classification problem. Thus, the last layers have been modified in accordance with the proposed task. The networks have been trained to classify the images monkeypox vs. non-monkeypox. The comparative results for the networks are shown in Table 3. As can be seen, the network accuracies differ according to the epoch numbers. The best performances were achieved with MobileNetv2 and EfficientNetb0 in 60 epochs.

The EfficientNetb0 and MobileNetv2 networks, which show the best performances for the human monkeypox classification task, have also been evaluated in terms of different criteria indexes. These criteria indexes, their formulas, and the network results are illustrated in Table 4.

The confusion matrices of the networks EfficientNetb0 and MobileNetv2, which produced the best accuracies, are given in Fig. 3. Figure 4 shows some visual results from the system: (a) shows some outputs from EfficientNetb0 and (b) from MobileNetv2. These sample images are from the test set. Although the test set includes different backgrounds and lesions from different parts of the human body, the system results are satisfactory.

The proposed system has been compared to the literature and the results are reported in Table 5. All studies in the table used the same MSLD database. As seen in the table, the proposed method that used MobileNetv2 outperforms other methods in terms of Precision, Sensitivity, F1 score, and Accuracy.

**Table 3** Pre-tarined network performances for human monkeypox classification task

| Network Name | Accuracy (%) | | | | |
|---|---|---|---|---|---|
| | Epochs | 15 | 30 | 45 | 60 |
| ResNet18 | | **86.87** | 66.67 | 75.56 | 73.33 |
| GoogleNet | | 55.56 | **82.22** | 80.00 | 77.78 |
| EfficientNetb0 | | 82.22 | 88.89 | 88.89 | **91.11** |
| NasnetMobile | | 82.22 | 84.44 | 84.44 | **86.67** |
| ShuffleNet | | 77.78 | 77.78 | **80.00** | **80.00** |
| MobileNetv2 | | 82.22 | 88.89 | 88.89 | **91.11** |

**Table 4** Objective evaluations of EfficientNetb0 and MobileNetv2

| Measure | Formula | EfficientNetb0 | MobileNetv2 |
|---|---|---|---|
| Jaccard | *TP / (TP+FP+FN)* | 82.61 | 81.82 |
| Precision | *TP/(TP+FP)* | 86.36 | 90.00 |
| Sensitivity | *TP/(TP+FN)* | 95.00 | 90.00 |
| F1 Score | *2*TP/(2*TP+FP+FN)* | 90.48 | 90.00 |
| Accuracy | *(TP+TN)/ (TP+TN+FP+FN)* | 91.11 | 91.11 |

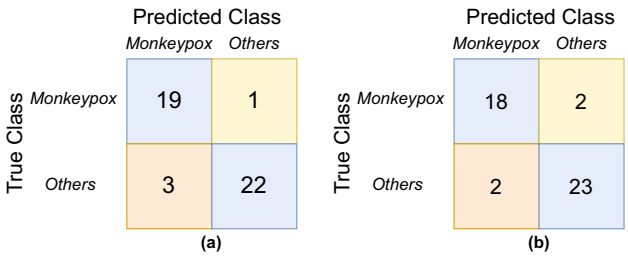


**Fig. 3** Confusion matrices of the networks which produced the best accuracy: (**a**) EfficientNetb0, (**b**) MobileNetv2

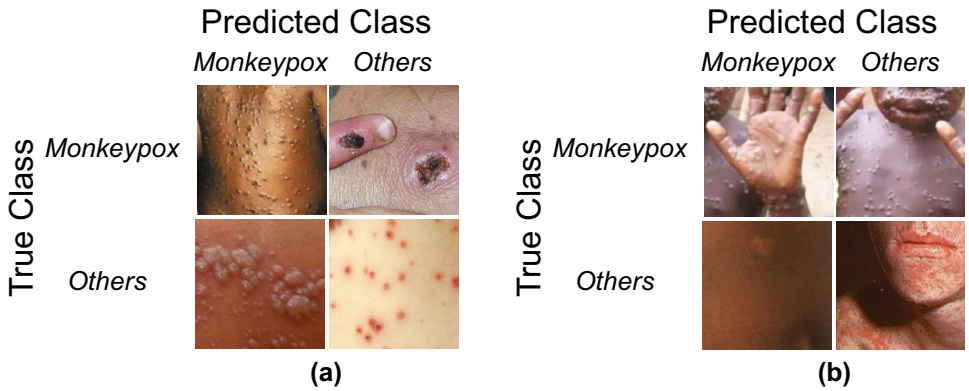


**Fig. 4** Some visual results of the networks: (**a**) EfficientNetb0, (**b**) MobileNetv2

## Experiments on mobile application

In this study, an Android mobile application has been realized to detect monkeypox. Instead of developing a new application from scratch, the authors chose to use the ImageClassifier example application from TensorFlow Lite GitHub repository [58] as a basis for this work. The TensorFlow Lite GitHub repository provides several example applications for ML, like text classification, pose estimation, image segmentation, and image classification. The example applications are licensed under the open source Apache License Version 2 [59]. The ImageClassification example application has been downloaded and modified for the use of monkeypox detection.

The realized application uses the camera of the Android device for input. For this purpose, the Camera2 API of the Android platform has been used. In the first phase, the video stream from the camera is provided to the application.

The video stream frames are converted from YUV to RGB in the second phase. The frames are also previewed on the interface of the application. The RGB image is processed to make it available for use in the TensorFlow Lite runtime.

In the third phase, the prepared image is provided to the ImageClassifier object of the TensorFlow Lite library. The ImageClassifier object makes inference (prediction) based on the TensorFlow Lite model and returns a list of recognitions for each image frame. The recognition list contains the classes and their possibilities. In our case, the ImageClassifier returns positive and negative classes with their respective possibilities.

In the last phase, the application shows the name of the class with the higher possibility on the top left portion of the device screen. The application also indicates the inference time in milliseconds. The inference time is the prediction time. Some example screenshots of the developed Android application are shown in Fig. 5. The first and second screenshots are a sample of true positive, the third one is a true negative, and the last one is a false positive. Although sometimes it produces false responses, in general, the system produces satisfactory results. System performance can be further increased by using a dataset containing more images.

The application has been compiled with the minimum Android SDK version 21. Then, it has been deployed and successfully run on three different Android devices for testing. During the run-time, inference times for devices have been observed, and then the average inference time for each device has been calculated. The devices are shown in Table 6. As seen from the table, there is a difference between inference times, although the core count and the frequency of the devices' CPUs are similar. This difference may be related to the different characteristics of the mobile processor platforms of the devices like GPU and image signal processor.

**Table 5** Comparative results with other studies that used the same MSLD database

| Network | Jaccard | Precision | Sensitivity | F1 Score | Accuracy |
|---|---|---|---|---|---|
| VGG16 [6] | - | 0.85 | 0.81 | 0.83 | 81.48 |
| ResNet50 [6] | - | 0.87 | 0.83 | 0.84 | 82.96 |
| InceptionV3 [6] | - | 0.74 | 0.81 | 0.78 | 74.07 |
| Ensemble [6] | - | 0.84 | 0.79 | 0.81 | 79.26 |
| Proposed(MobileNetv2) | 0.81 | 0.90 | 0.90 | 0.90 | 91.11 |

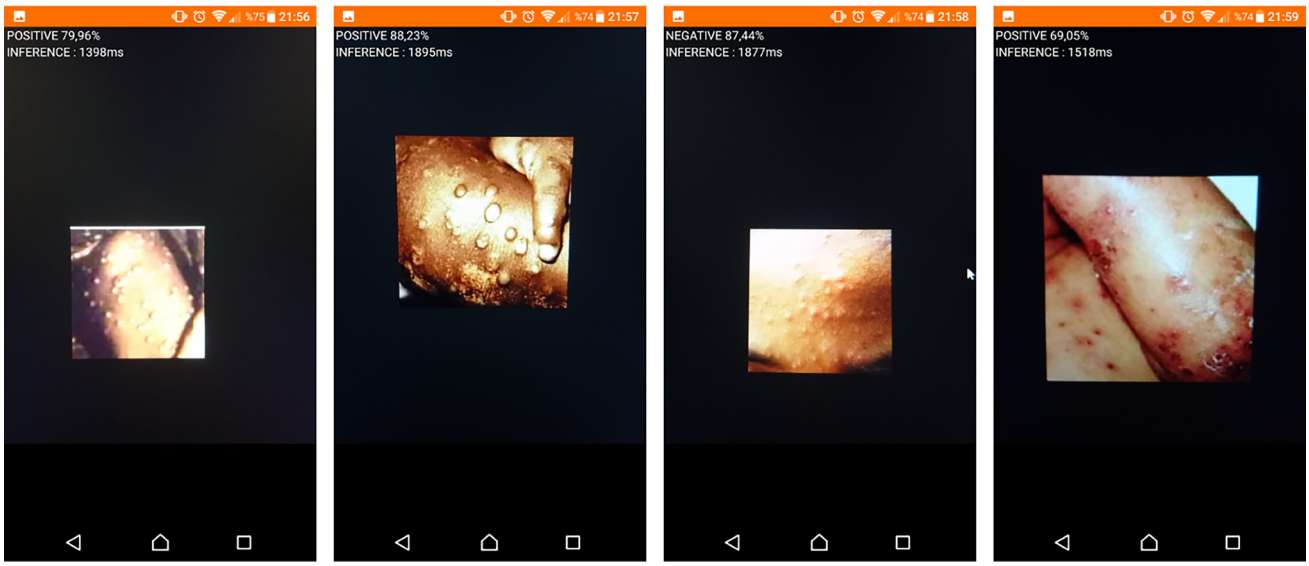


**Fig. 5** Sample screenshots from the mobile application. The first and second screenshots are sample of true positive, third one is true negative and the last one is false positive

**Table 6** Device information and performance

| Device Model | Android Version | CPU | RAM | Inference Time |
|---|---|---|---|---|
| Device 1 | 7.1.1 | Octa-core Max 2.0GHz | 3GB | 197 ms |
| Device 2 | 10 | Octa-core Max 2.2GHz | 4GB | 91 ms |
| Device 3 | 11 | Octa-core Max 2.0GHz | 4GB | 138 ms |

## Conclusion

This study presents a mobile system to automatically detect human monkeypox skin lesions. For this purpose, first, a deep-transfer learning-based system has been trained using MSLD database images. In this stage, different pre-trained networks have been retrained using the transfer learning approach, and the network results have been compared. Then, MobileNetv2 which showed one of the best performance in terms of accuracy as 91.11% was adapted into an Android mobile application. The proposed study was also compared with other studies using the same database and produced better results.

The system allows monkeypox suspects to conduct preliminary screening from home comfort using their mobile devices. It enables infected people to take action in the early stages of the infection. Therefore, the system helps to reduce the spread rate of the human monkeypox outbreak. This study is one of the pioneering studies for visual monkeypox detection using mobile devices.

In this study, the authors chose to use the tools provided by the Android platform and developed an Android application. However, as mentioned above, a multi-platform framework like React Native, Flutter, and KMM can also be used to create mobile applications to target multiple platforms. In addition, PWAs are another alternative to developing multi-platform mobile applications.

In the future, the usage of multi-platform frameworks and PWAs for solving these kinds of problems can be exercised. Also, the proposed system's network can be made more robust using more stable and comprehensive monkeypox datasets. In addition, the proposed system can be extended to classify different skin diseases.

**Author contributions** All authors of this study developed the system, wrote and reviewed the manuscript

**Funding** The authors did not receive support from any organization for the submitted work.

**Data availability** Data used in this article were obtained from the Kaggle (https://www.kaggle.com/datasets/nafin59/monkeypox-skin-lesion-dataset?resource=download).

## Declarations

**Ethics Approval** Not applicable

**Research involving human and animal participants** Not applicable

**Consent to participate** Not applicable

**Consent for publication** Not applicable

**Conflicts of interest** The authors declare they have no financial interests

## References


1. WHO: Multi-country monkeypox outbreak in non-endemic countries:Update. Accessed 20 July 2022 (2022). https://www.who.int/emergencies/disease-outbreak-news/item/2022-DON388
2. Bunge, E.M., Hoet, B., Chen, L., Lienert, F.,Weidenthaler, H., Baer, L.R., Steffen, R.: The changing epidemiology of human monkeypoxa potential threat? a systematic review. PLoS Neglected Tropical Diseases 16, 0010141 (2022). https://doi.rog/10.1371/journal.pntd.0010141
3. Bragazzi, N.L., Kong, J.D., Mahroum, N., Tsigalou, C., KhamisyFarah, R., Converti, M., Wu, J.: Epidemiological trends and clinical features of the ongoing monkeypox epidemic: A preliminary pooled data analysis and literature review. Journal of Medical Virology (2022). https://doi.org/10.1002/jmv.27931
4. Taylor, L.: Monkeypox: Who declares a public health emergency of international concern. BMJ, 1874 (2022). https://doi.org/10.1136/bmj.o1874
5. Zumla, A., Valdoleiros, S.R., Haider, N., Asogun, D., Ntoumi, F., Petersen, E., Kock, R.: Monkeypox outbreaks outside endemic regions: scientific and social priorities. The Lancet Infectious Diseases 22, 929–931 (2022). https://doi.org/10.1016/S1473-3099(22)00354-1
6. Ali, S.N., Ahmed, M.T., Paul, J., Jahan, T., Sani, S.M.S., Noor, N., Hasan, T.: Monkeypox skin lesion detection using deep learning models: A preliminary feasibility study. arXiv preprint arXiv:2207.03342 (2022)
7. Ding, C., Tao, D.: Robust face recognition via multimodal deep face representation. IEEE Transactions on Multimedia **17**, 2049–2058 (2015). https://doi.org/10.1109/TMM.2015.2477042
8. Liu, J., Gu, Y., Kamijo, S.: Customer behavior classification using surveillance camera for marketing. Multimedia Tools and Applications **76**, 6595–6622 (2017). https://doi.org/10.1007/s11042-016-3342-1
9. Ozbay, F.A., Alatas, B.: Fake news detection within online social media using supervised artificial intelligence algorithms. Physica A: Statistical Mechanics and its Applications **540**, 123174 (2020). https://doi.org/10.1016/j.physa.2019.123174
10. Yan, K., Wang, X., Lu, L., Summers, R.M.: Deeplesion: automated mining of large-scale lesion annotations and universal lesion detection with deep learning. Journal of Medical Imaging 5, 1 (2018). https://doi.org/10.1117/1.JMI.5.3.036501
11. Oztel, I., Yolcu, G., Ersoy, I., White, T.A., Bunyak, F.: Deep learning approaches in electron microscopy imaging for mitochondria segmentation. International Journal of Data Mining and Bioinformatics **21**, 91 (2018). https://doi.org/10.1504/IJDMB.2018.096398
12. Sahin VH, Oztel I (2021) Developing a message broadcasting system for natural disasters. International Journal of Engineering Research and Development **13**:13–21, https://doi.org/10.29137/umagd.664730
13. Gonzlez, D., Patricio, M.A., Berlanga, A., Molina, J.M.: A super resolution enhancement of uav images based on a convolutional neural network for mobile devices. Personal and Ubiquitous Computing **26**, 1193–1204 (2022). https://doi.org/10.1007/s00779-019-01355-5
14. Joshi, R., Joseph, A., Mihandoust, S., Madathil, K.C., Cotten, S.R.: A mobile application-based home assessment tool for patients undergoing joint replacement surgery: A qualitative feasibility study. Applied Ergonomics **103**, 103796 (2022). https://doi.org/10.1016/j.apergo.2022.103796
15. Buono, F.D., Lalloo, C., Larkin, K., Zempsky, W.T., Ball, S., Grau, L.E., Pham, Q., Stinson, J.: Innovation in the treatment of persistent pain in adults with neurofibromatosis type 1 (nf1): Implementation of the icancope mobile application. Contemporary Clinical Trials Communications **25**, 100883 (2022). https://doi.org/10.1016/j.conctc.2021.100883
16. Hung, S.-W., Chang, C.-W., Ma, Y.-C.: A new reality: Exploring continuance intention to use mobile augmented reality for entertainment purposes. Technology in Society **67,** 101757 (2021). https://doi.org/10.1016/j.techsoc.2021.101757
17. Amazon: Kindle for Android. 2022. https://www.amazon.com/dp/B004DLPXAO, Accessed 4 Aug 2022
18. Coca, L.-G., Cusmuliuc, C.G., Iftene, A.: Automatic tarmac crack identificationapplication. Procedia Computer Science **192**, 478–486 (2021). https://doi.org/10.1016/j.procs.2021.08.049
19. Dammak, B., Turki, M., Cheikhrouhou, S., Baklouti, M., Mars, R., Dhahbi, A.: Lorachaincare: An iot architecture integrating blockchain and lora network for personal health care data monitoring. Sensors **22**(4) (2022). https://doi.org/10.3390/s22041497
20. Breman, J.G., Kalisa-Ruti, Steniowski, M.V., Zanotto, E., Gromyko, A.I., Arita, I.: Human monkeypox, 1970-79. Bulletin of the World Health Organization **58**, 165–82 (1980)
21. Ladnyj, I.D., Ziegler, P., Kima, E.: A human infection caused by monkeypox virus in basankusu territory, democratic republic of the congo. Bulletin of the World Health Organization **46**, 593–7 (1972)
22. Heymann, D.L., Szczeniowski, M., Esteves, K.: Re-emergence of monkeypox in africa: a review of the past six years. British Medical Bulletin **54**, 693–702 (1998)
23. Wilson, M.E., Hughes, J.M., McCollum, A.M., Damon, I.K.: Human monkeypox. Clinical Infectious Diseases **58**, 260–267 (2014). https://doi.org/10.1093/cid/cit703
24. Ahsan, M.M., Uddin, M.R., Farjana, M., Sakib, A.N., Momin, K.A., Luna, S.A.: Image data collection and implementation of deep learning-based model in detecting monkeypox disease using modified vgg16 (2022). http://arxiv.org/abs/2206.01862
25. Watts, P., Breedon, P., Nduka, C., Neville, C., Venables, V., Clarke, S.: Cloud computing mobile application for remote monitoring of bells palsy. Journal of Medical Systems **44**, 149 (2020). https://doi.org/10.1007/s10916-020-01605-7
26. Mamoun, R., Nasor, M., Abulikailik, S.H.: Design and development of mobile healthcare application prototype using flutter. In: 2020 International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE), pp. 1–6 (2021). https://doi.org/10.1109/ICCCEEE49695.2021.9429595
27. Cho, N.-B., Cho, S.-R., Choi, S.H., You, H., Nam, S.I., Kim, H.: Short-term and long-term efficacy of oropharyngolaryngeal strengthening training on voice using a mobile healthcare application in elderly women. Communication Sciences & Disorders **26**, 219–230 (2021). https://doi.org/10.12963/csd.21799
28. Berger-Groch, J., Keitsch, M., Reiter, A., Weiss, S., Frosch, K., Priemel, M.: The use of mobile applications for the diagnosis and treatment of tumors in orthopaedic oncology a systematic review. Journal of Medical Systems **45**, 99 (2021).10.1007/s10916-021-01774-z
29. A mobile augmented reality application for supporting real-time skin lesion analysis based on deep learning. Journal of Real-Time Image Processing **18**, 1247–1259 (2021). https://doi.org/10.1007/s11554-021-01109-8
30. Krohling, B., Castro, P.B.C., Pacheco, A.G.C., Krohling, R.A.: A smartphone based application for skin cancer classification using deep learning with clinical images and lesion information (2021)
31. Doukas, C., Stagkopoulos, P., Kiranoudis, C.T., Maglogiannis, I.: Automated skin lesion assessment using mobile technologies and

cloud platforms, pp. 2444–2447 (2012). https://doi.org/10.1109/EMBC.2012.6346458
32. Vasefi, F., MacKinnon, N.B., Horita, T., Shi, K., Munia, T.T.K., Tavakolian, K., Alhashim, M., Fazel-Rezai, R.: A smartphone application for psoriasis segmentation and classification (conference presentation), p. 52 (2017). https://doi.org/10.1117/12.2261505
33. Abadi, M., Agarwal, A., Barham, P., et al: TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems (2015). https://doi.org/10.5281/zenodo.4724125, https://www.tensorflow.org/
34. TensorFlow Lite: TensorFlow Lite ML for Mobile and Edge Devices. https://www.tensorflow.org/lite, Accessed 20 Jul 2022
35. Mayya, V., Pai, R.M., Pai, M.M.M.: Automatic facial expression recognition using dcnn. Procedia Computer Science 93, 453–461 (2016). https://doi.org/10.1016/j.procs.2016.07.233
36. Xie, S., Hu, H.: Facial expression recognition with frr-cnn. Electronics Letters**53**, 235–237 (2017). https://doi.org/10.1049/el.2016.4328
37. Jaderberg, M., Simonyan, K., Vedaldi, A., Zisserman, A.: Deep structured output learning for unconstrained text recognition (2014)
38. Wang, P., Wang, P., Fan, E.: Violence detection and face recognition based on deep learning. Pattern Recognition Letters 142, 20–24 (2021). https://doi.org/10.1016/j.patrec.2020.11.018
39. Duan, M., Li, K., Yang, C., Li, K.: A hybrid deep learning cnnelm for age and gender classification. Neurocomputing 275, 448–461 (2018). https://doi.org/10.1016/j.neucom.2017.08.062
40. Aydogdu, M.F., Celik, V., Demirci, M.F.: Comparison of Three Different CNN Architectures for Age Classification. In: 2017 IEEE 11th International Conference on Semantic Computing (ICSC), pp. 372–377 (2017). https://doi.org/10.1109/ICSC.2017.61
41. Gkioxari, G., Girshick, R., Malik, J.: Contextual action recognition with r*cnn. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV) (2015)
42. Al-Milaji, Z., Ersoy, I., Hafiane, A., Palaniappan, K., Bunyak, F.: Integrating segmentation with deep learning for enhanced classification of epithelial and stromal tissues in h & e images. Pattern Recognition Letters **119**, 214–221 (2019). https://doi.org/10.1016/j.patrec.2017.09.015
43. Bao, R., Al-Shakarji, N.M., Bunyak, F., Palaniappan, K.: Dmnet: Dualstream marker guided deep network for dense cell segmentation and lineage tracking. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops, pp. 3361–3370 (2021)
44. Hamad, A., Ersoy, I., Bunyak, F.: Improving nuclei classification performance in h&e stained tissue images using fully convolutional regression network and convolutional neural network. In: 2018 IEEE Applied Imagery Pattern Recognition Workshop (AIPR), pp. 1–6 (2018). https://doi.org/10.1109/AIPR.2018.8707397
45. Shuvo, M.M.H., Kassim, Y.M., Bunyak, F., Glinskii, O.V., Xie, L., Glinsky, V.V., Huxley, V.H., Thakkar, M.M., Palaniappan, K.: Multi-focus image fusion for confocal microscopy using u-net regression map. In: 2020 25th International Conference on Pattern Recognition (ICPR), pp. 4317–4323 (2021). https://doi.org/10.1109/ICPR48806.2021.9412122
46. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press, USA (2016). http://www.deeplearningbook.org
47. Matlab: crossentropy. https://www.mathworks.com/help/deeplearning/ref/dlarray.crossentropy.html, Accessed 20 Jul 2022
48. TensorFlow Guide: Transfer learning and fine-tuning. Accessed 20 July2022. https://www.tensorflow.org/guide/keras/transfer_learning, Accessed 20 Jul 2022
49. Google: Android OS. https://www.android.com, Accessed 4 Aug 2022
50. Google: Mobile App Developer Tools - Android Developers. https://developer.android.com, Accessed 4 Aug 2022
51. Platforms, M.: React Native https://reactnative.dev, Accessed 4 Aug 2022
52. Google: React Native. https://flutter.dev, Accessed 4 Aug 2022
53. JetBrains: Kotlin Multiplatform Mobile. https://kotlinlang.org/docs/multiplatform-mobile-getting-started.html, Accessed 4 Aug 2022
54. Mozilla: Progressive web apps (PWAs) MDN. https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps, Accessed 4 Aug 2022
55. NumPy: NumPy Homepage. https://numpy.org, Accessed 20 Jul 2022
56. Kaggle: Monkeypox Skin Lesion Dataset. https://www.kaggle.com/datasets/nafin59/monkeypox-skin-lesion-dataset, Accessed 20 Jul 2022
57. Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., Fei-Fei, L.: ImageNet: A Large-Scale Hierarchical Image Database. In: CVPR09 (2009)
58. TensorFlow: TensorFlow Examples Repository. https://github.com/tensorflow/examples, Accessed 20 Jul 2022
59. The Apache Software Foundation: Apache License Version 2.0. https://apache.org/licenses/LICENSE-2.0, Accessed 20 Jul 2022