ADVANCED INTELLIGENT SYSTEMS Open Access
www.advintellsyst.com

Deep Learning-Based Skin Diseases Classification using Smartphones

Ismail Oztel, Gozde Yolcu Oztel, and Veysel Harun Sahin*

Skin disease recognition is one of the essential topics in the medical industry. Detecting skin disease from appearance can be difficult due to the similar appearance of skin lesions. In some cases, such as the monkeypox virus, the illness must be quickly determined, and the patients must be isolated to reduce the spreading of the disease. This study aims to create a deep learning-based automated intelligent mobile application to detect skin disease. First, different small-size pretrained networks are trained for skin lesion image classification. Then, the most suitable network from the viewpoint of both performance and mobile compatibility is transformed into the TensorFlow Lite format. Finally, a mobile application is created on the Android platform that utilizes the smartphone's camera to obtain images and uses TensorFlow Lite to make predictions. The proposed system produces 74.27% classification accuracy for seven classes on a combined dataset. It produces comparable/better results compared to the literature. Owing to the proposed system, the patients can make a preliminary diagnosis of their lesions using their smartphones. Thus, risky patients can be encouraged to visit the hospital for a definitive diagnosis. In addition, the mobile application can avoid undue stress and false alarms.

1. Introduction

Early detection can help control a disease's progress, reduce symptoms, and improve the life quality for numerous illnesses. Owing to technological development, artificial intelligence and machine learning approaches can assist the authorities in the early diagnosis of disease.[1,2]

Skin diseases may significantly reduce life quality. Mainly due to expanding contamination, malnutrition, and viruses. The number of skin-related patients increases at a quicker rate.[3] Skin diseases include skin infections, which may differ according to skin disease type.[4] Since the 1970s, skin cancer has been one of the most prevalent diseases globally. According to Skin Cancer Foundation statistics, one out of every five people in the United States will have skin cancer during their lifetime.[5] Skin cancer contains two main categories, nonmelanoma and melanoma.[6] Dermatologists frequently diagnose the lesions as nonmelanoma or melanoma by examining different features of skin such as color, texture, and shape with the naked eye.[4] According to Massone et al.,[7] using an additional device can provide a more accurate diagnosis.

Recently, to classify skin cancers, image analysis with artificial intelligence techniques has been actively studied by researchers. In contrast, a skin lesion can also be an indicator of a skin disease other than skin cancer, like monkeypox. Also, a prompt diagnosis is vital for the monkeypox. Thus, the infected people can quickly be isolated, and the spread rate of the disease can be reduced.

Today we can also utilize smartphones and their cameras to classify skin diseases with the help of deep learning techniques. The usage of smartphones has risen extensively in the last decade. There are two main reasons for the popularity of smartphones: connectivity and sensors. We can use smartphones to connect to the internet, wireless networks, and other devices. In addition, we can transfer different kinds of data. With the help of connectivity, smartphones have become more than just mobile phones that we use only for talking and messaging. They connected us to the world in many different ways. We can also use smartphones to interact with real life. Unlike traditional computer systems, smartphones include various types of sensors like image sensors, accelerometers, and barometers. Both connectivity and sensors made smartphones a part of our daily lives. Today smartphones are used in many different areas like healthcare,[8–12] sports,[13] disasters,[14,15] agriculture,[16,17] and special needs.[18]

This study presents a mobile system for diagnosis and monitoring of skin diseases. The proposed system is reasonably quick and straightforward for the public to use. The system focuses on classifying various skin diseases with the help of deep learning and smartphones. In the context of this study, firstly, different skin lesion datasets have been used to form a combined dataset.

I. Oztel
Department of Computer Engineering
Sakarya University
54050 Sakarya, Turkiye
E-mail: ioztel@sakarya.edu.tr

G. Yolcu Oztel, V. H. Sahin
Department of Software Engineering
Sakarya University
54050 Sakarya, Turkiye

The ORCID identification number(s) for the author(s) of this article can be found under https://doi.org/10.1002/aisy.202300211.

DOI: 10.1002/aisy.202300211

The datasets that have been combined in this step are the PAD-UFES-20[19,20] and the Monkeypox Skin Lesion Dataset (MSLD).[21,22] The newly formed dataset contains images of various skin diseases, including monkeypox. Second, data balancing has been applied to prevent the model from becoming biased toward any particular class. Then, different pretrained convolutional neural network (CNN) models were trained with the newly created dataset, and network results were compared. To be able to utilize the deep learning models on mobile devices, TensorFlow[23,24] has been used. TensorFlow is a machine learning (ML) library that helps to create ML models. In this study, the most suitable network from the viewpoint of both performance and mobile compatibility has been created on TensorFlow and transformed into a TensorFlow Lite model[25] to be used in the mobile system. TensorFlow Lite is a mobile framework that is used to run ML models on portable systems such as smartphones and Internet of Things (IoT) devices. Eventually, a mobile application that runs on Android smartphones has been generated in this study. The mobile application gathers skin images through the Android smartphone's camera and performs classification utilizing the TensorFlow Lite framework.

The contributions of this study are given in the following items: 1) Contribution in terms of healthcare: An essential, affordable, and noninvasive disease diagnostic mobile application has been created. Owing to the proposed system, contagious and requiring urgent detection of diseases such as monkeypox will able to be prediagnosed by patients' mobile phones. Patients can be motivated for an actual diagnosis through an expert. Therefore, the transmission rate of the disease can be reduced with the help of the proposed system; 2) Contribution in terms of computer science: A low-modified ResNet-18 model has been introduced. The model was trained to classify skin lesion images and transformed for mobile applications; and 3) Contribution in terms of data science: Two different skin lesion datasets have been combined and a new dataset that includes monkeypox images has been used for seven-class classification problem.

2. Related Works

2.1. Skin Disease Image Analysis

Skin disease image analysis is a challenging and popular task in the computer vision field. Nida et al.[26] studied melanoma lesion detection and segmentation using deep region-based convolutional neural network and fuzzy C-means clustering. Karthik et al.[27] classified the skin images as acne, actinic keratosis, melanoma, and psoriasis. They used CNN-based approach. In the study of Anand et al.,[4] a transfer learning-based model has been developed with the help of a pretrained Xception model. In the study of Zheng et al.,[28] skin disease health detection of college students has been done. Three skin lesions were identified and classified: actinic keratosis, melanocytic nevus, and vascular lesions. Srinivasu et al.[29] proposed a study that classifies skin disease using MobileNet V2 and long short-term memory. They used the HAM10000 dataset and compared the performance of the study against the state-of-the-art deep learning models. Their shared results show that the proposed method is faster than the conventional MobileNet. Along with this, they also have developed a mobile detection application. Medhat et al. and Alyami et al.[30,31] selected two cancer types from PAD-UFES-20 dataset (in which we used six classes in it) and applied a comparative study. Chen et al.[32] used clinical data and visual data to classify skin lesions.

Due to the monkeypox pandemic, monkeypox detection studies have recently been popular in the literature. In the study of Ahsan et al.,[33] monkeypox-infected skin images were collected from Google and analyzed using deep learning approaches. Ali et al.[22] created a human monkeypox image dataset. Then, they classified the images using VGG16, ResNet50, InceptionV3, and Ensemble networks. Sahin et al.[34] performed a binary image classification as monkeypox versus nonmonkeypox.

2.2. Mobile Applications for Healthcare Improvement

Owing to recent technological developments, smartphones can be used to monitor healthcare. Therefore, people can rapidly learn about their health conditions and be monitored by their doctors without additional device costs.

Mamoun et al.[35] developed a healthcare mobile application prototype. The application provides diseases to be diagnosed online by medical specialists. Moreover, it simplifies the ordering of medicines using online payment. Cho et al.[36] proposed a mobile healthcare application for investigating the effectiveness of oral and pharyngolaryngeal strength training on voice in older women. Watts et al.[37] developed The Facial Remote Activity Monitoring Eyewear (FRAME) mobile application. This application provides nearly real-time, safe remote access for therapists monitoring their patients. Berger-Groch et al.[38] reviewed mobile applications used for diagnosing and treating tumors in orthopedic oncology.

3. Methodology

The proposed study focuses on classifying skin diseases using smartphones. The proposed system concentrates on distinguishing several skin lesions, including monkeypox. **Figure 1** shows the general system pipeline. As seen in the figure, first, a merged skin lesion image dataset has been created and some data augmentation techniques have been applied to these images. Then, using the transfer learning approach, different small-size pretrained networks were trained with these images. The pretrained network models have been compared, and the most suitable network from the viewpoint of both performance and mobile compatibility has been converted to the TensorFlow Lite model. Finally, a mobile application has been developed for the skin lesion classification task.

3.1. Deep Transfer Learning for Skin Image Classification

Advances in deep learning started to enable accurate image analysis. It shows successful results in the literature.[39–42] CNN is a deep learning model that includes multiple layers to extract higher-level features from the raw input.

CNNs are created with different combinations of convolution, pooling, activation, dropout, fully connected, and some other

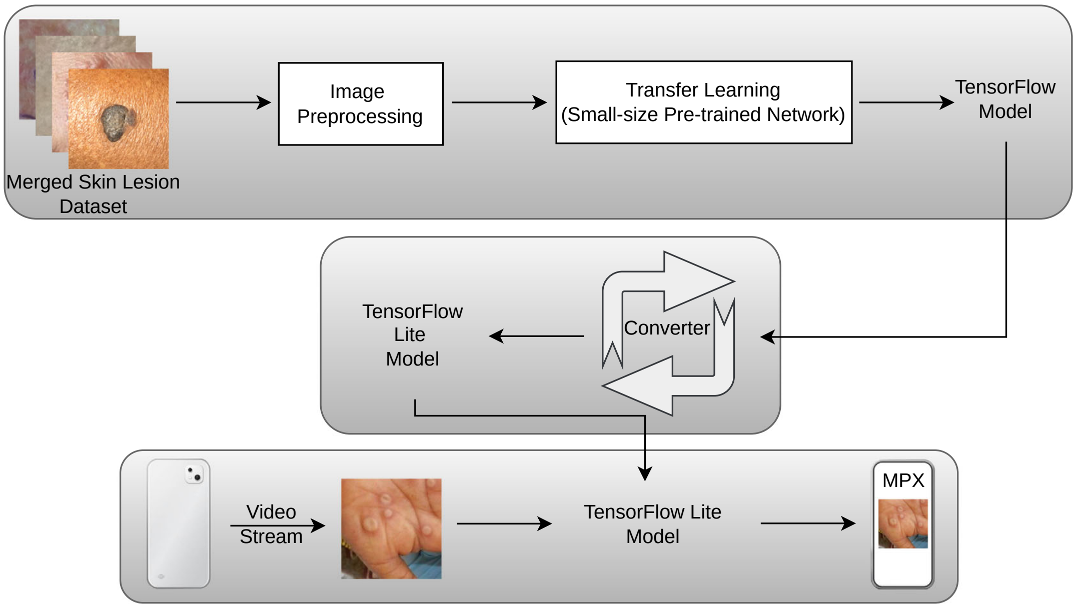

Figure 1. General system pipeline.

layers. The last layer is the classification layer for image classification problems. In the convolution layers, filters are convolved with the input data, and features are extracted from training samples. The mathematical definition of convolution is shown in Equation (1).[43]

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

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

Pooling operations can transform multiple cells into one cell. The pooling layer is used to reduce the input size in width and height. As an activation function, the rectified linear unit (ReLU) is mostly used. It regularizes the CNN with a reduction of the model parameters because it ignores negative values. Theoretically, these values are not activated regardless.

Dropout randomly removes some output features of a layer during training. Thus, it can prevent the model from overfitting. The fully connected layer connects to all nodes in the previous layer. It is used just before the classification layer.

Training data for specific problems such as medical image analysis is often limited because of costly acquisition and limited accessibility. Also, the training operation usually requires high computing costs. A common solution to these problems is using transfer learning during CNN model training. Transfer learning carries features learned on one problem to a new problem.[44]

In this study, different pretrained networks have been used for the skin lesion classification task. The list of the networks and their characteristics can be seen in **Table 1**. These networks have been selected because of their small size. Since the developed application will be used on mobile devices with resource constraints, we made our choice in this way. The sizes reported in the table are of approximate values. They may vary depending on different model formats such as onxx, tf, and tflite.

A unique fully connected layer with seven outputs has been created as a substitute for the fully connected layers of the pretrained networks. Thus, the networks have been made ready for the seven-class skin disease classification.

3.2. Mobile Application for Skin Lesion Detection

To utilize smartphones for skin lesion detection, first, a mobile platform should be chosen. In this first step, the authors decided to use the Android operating system (OS).[45] Android OS supports a wide range of device types such as tablets, smartphones, smartwatches, and TVs.

An Android application can be developed using different approaches. The first approach is to create a native Android application that compiles to and runs on only Android OS. We can apply this approach using the mobile application tools provided in the Android ecosystem.[46] The second approach is using cross-platform/multiplatform frameworks like Flutter,[47]

Table 1. The pretrained networks' details.

Network name	Depth	Size [MB]	Parameters [millions]	Input size
EfficientNetb0	82	20	5.3	224 × 224
ResNet-18	18	44	11.7	224 × 224
MobileNetv2	53	13	3.5	224 × 224
GoogleNet	22	27	7.0	224 × 224
NasnetMobile	*	20	5.3	224 × 224
ShuffleNet	50	5.4	1.4	224 × 224
EfficientNetb3	107	75	12	300 × 300

*NasnetMobile does not have a linear sequence.

Kotlin Multiplatform Mobile,[48] and React Native.[49] These frameworks' main aim is to provide a platform to create an application using a single code base that can run on multiple computing platforms. The last approach is utilizing Progressive Web Applications (PWAs).[50] PWAs are web applications that can be installed on devices and work offline.

In this study, the authors decided to use the first approach. In line with this approach, Android Studio has been chosen. The Android 12 standard development kit (SDK) and hence Android application programming interface (API) level 31 was used with the Android Studio. The development was performed using Kotlin programming language. Skin lesion detection using a smartphone requires gathering skin images using the mobile device's camera. Currently, there are two different camera APIs in the Android platform. The first one is Camera2 API, and the second one is CameraX API. In this study, CameraX API was used.

An ML framework supporting mobile systems is needed to classify skin lesion images with smartphones. For this need, TensorFlow Lite has been chosen. With the help of TensorFlow Lite, we can run ML models on portable devices like smartphones and edge devices. One of the main aspects of this framework is that it is optimized to perform ML tasks on mobile devices. It provides high performance. It can use hardware accelerators like graphics processing units (GPUs) and digital signal processors (DSPs). It can also use optimized models.

Executing the models on TensorFlow Lite run-time and making predictions is called inference. TensorFlow Lite offers different ways to run inference, such as TensorFlow Lite Support Library, TensorFlow Lite Task Library, and TensorFlow Lite Interpreter API. The authors have chosen the TensorFlow Lite Interpreter API. It is a low-level API that can be used on multiple platforms and languages.

The technologies used during the mobile system development are given in **Table 2**.

TensorFlow Lite has its own model format called the TensorFlow Lite model. Therefore, the network training and the model creation have been done using TensorFlow. Before deploying to the Android application, the TensorFlow model was transformed into the TensorFlow Lite format.

Table 2. Technologies used during the development of Android application.

Type	Name
Mobile platform	Android OS
Android SDK	Android 12 SDK
Camera API	CameraX API
Programming language	Kotlin
ML framework	TensorFlow Lite
Inference API/Library	TensorFlow Lite Interpreter API

A typical process of machine learning tasks on Android smartphones using TensorFlow Lite is illustrated in **Figure 2**. The input data are first obtained using the sensors of the device. In this study, the input data are the image frames from the camera. In the second step, the input data are converted to a tensor. Tensors are the inputs of the TensorFlow Lite run-time. After getting the input, the TensorFlow Lite run-time runs inference (makes a prediction) on the data and outputs its result again as a tensor. In the last step, output tensors are interpreted, and prediction is gathered.

CameraX API provides four use cases: 1) *preview use case* that helps display the camera stream on the smartphone screen; 2) *image analysis use case* that helps the processing of the image frames; 3) *image capture use case* that enables capturing photos; and 4) *video capture use case* that enables the capturing video and audio. In this study, the preview and the image analysis use cases were used. In light of this information, the workflow of the developed application is shown in **Figure 3** and is explained in detail below.

In the **first step**, with the help of the preview use case (preview object), the camera preview stream is connected to the user interface (UI) surface of the application for displaying the stream on the screen of the device. In the **second step**, with the help of the image analysis use case (ImageAnalysis object), the image frames of the device's camera are delivered to the application for processing and prediction purposes.

In this study, to use the ImageAnalysis object, three parameters needed to be set: operating mode, image format, and aspect ratio. The configured parameters are given in **Table 3** and explained here. The ImageAnalysis object has two operating modes: nonblocking and blocking. The application uses **nonblocking mode**. In this mode, during the analysis of an image frame, only the newest arriving image is cached in the image buffer. This mode is enabled by setting back the pressure strategy to STRATEGY_KEEP_ONLY_LATEST option. Because RGB image format is needed for ML tasks, **RGBA image format** was set as the output color space with the OUTPUT_IMAGE_FORMAT_RGBA_8888 option. In addition, **4:3 aspect ratio** was set with the RATIO_4_3 option.

After getting the image frames with the help of the ImageAnalysis object, in the **third step**, the RGB bits of the image are copied to a bitmap buffer. In the **fourth step**, the image is processed with the help of the ImageProcessor object of TensorFlow Lite. During this phase, the image is cropped, resized, rotated, and normalized. Then, a TensorImage is created, which is the input of the TensorFlow Lite run-time. In the **fifth step**, the TensorImage is given to the TensorFlow Lite Interpreter object as input, and inference is run. The TensorFlow Lite returns the classification result as an output tensor. In the **last step**, the prediction value of each class is

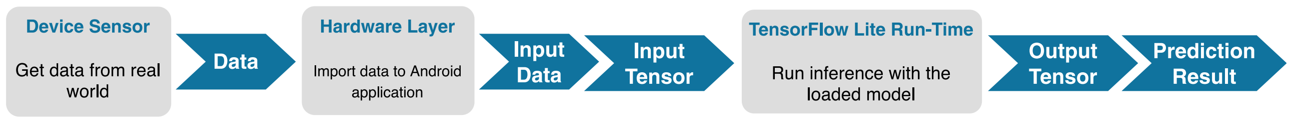

Figure 2. The typical process of machine learning tasks on Android smartphones using TensorFlow Lite.

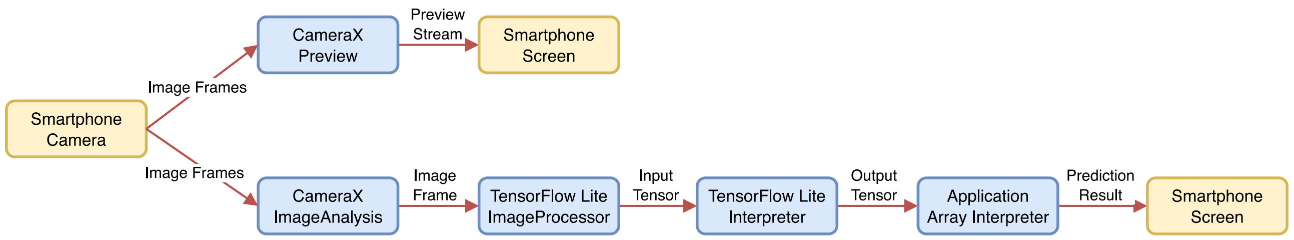

Figure 3. The workflow of the developed smartphone application.

Table 3. The image analysis uses case settings used in the application.

Parameter type	Selected value	Used option
Operating mode	Nonblocking	STRATEGY_KEEP_ONLY_LATEST
Image format	RGBA	OUTPUT_IMAGE_FORMAT_RGBA_8888
Aspect ratio	4:3	RATIO_4_3

extracted from the output tensor, and the maximum prediction value is shown to the user as a percentage value together with the corresponding class name.

4. Experimental Section

4.1. Dataset

The dataset used in this study includes seven classes. It has been created using two skin lesion datasets: PAD-UFES-20,[19,20] and MSLD.[21,22] PAD-UFES-20 dataset includes six class image lesions. These are actinic keratosis (ACK), basal cell carcinoma (BCC), melanoma (MEL), nevus (NEV), squamous cell carcinoma (SCC), and seborrheic keratosis (SEK). All of these lesions and images have been included in the combined dataset. MSLD has two classes: Monkeypox (MPX) and others. Label "others" includes different skin lesions other than monkeypox without a specific disease label. Therefore, the "others" class has been excluded from this study. Because of the monkeypox pandemic,[51] fast and correct classification of monkeypox images has been crucial recently. Therefore, the authors included the monkeypox (MPX) class in the newly formed dataset. As a result, 2,298 images which include 1,641 skin lesions from 1,373 patients have been used from PAD-UFES-20 and 102 images have been used from MSLD. **Table 4** shows the details of the datasets.

After that, the merged dataset was divided into two subsets: the training set (80%), and the testing set (20%). Also, to ensure a balanced dataset, the training subset has been augmented using reflection, rotation, scale, and translation approaches. The imbalanced and balanced versions of the dataset are shown in **Figure 4**. The values on the y-axis represent the number of samples.

4.2. Experiments on Skin Lesion Image Classifications

In this study, small-sized networks are especially preferred in terms of integration into mobile applications. Thus, six small-sized pretrained networks have been utilized. These networks had originally been trained using ImageNet dataset[52] and can classify images into 1,000 categories. The fully-connected layers of the networks were modified to classify skin lesion images into seven classes. The parameters of the training can be found in **Table 5**.

The classification results using the combined dataset are given in **Table 6**. As seen in the table, data balancing improved the result for all networks and the best accuracy has been obtained from EfficientNetb3 with balanced data. EfficientNetb3 was also trained and tested in this study because, as shown in **Table 7**, this network produced successful results in another study. However, we decided to use the ResNet-18 model for our application because of its mobile applicability from resource usage and performance viewpoints. The file sizes of our EfficientNetb3 and ResNet-18 TensorFlow Lite models are approximately 42.8, and 8.9 MB, respectively. As seen, these are different values from the mentioned sizes in Table 1. TensorFlow Lite converts the models into a smaller, more efficient machine learning (ML) model format.[53] In our preliminary mobile experiments, using

Table 4. Details of the datasets.

Class	Counts	Country	Job	Age	Body area
ACK	730	Different cities in Espírito Santo state, Brazil.	Patients are or have been farm workers with many hours of sun exposure per day	Patient average age is approximately 60 years old, but it may vary according to the diagnostic	Face, scalp, nose, lips, ears, neck, chest, abdomen, back, arm, forearm, hand, thigh, shin, and foot
BCC	845				
MEL	52				
NEV	244				
SCC	192				
SEK	235				
MPX	1428	N.A.	N.A.	N.A.	N.A.

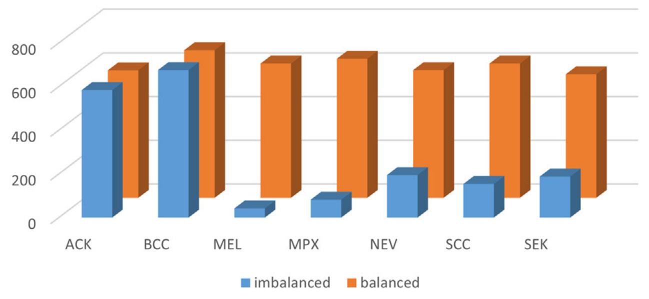

Figure 4. Comparison of the number of balanced and imbalanced data.

Table 5. The parameters used in the training phase.

Name	Value
Optimizer	Sgdm
Learning rate	0.001
Mini-batch size	16
Shuffle	every-epoch
Learning rate drop factor	0.1
Learning rate drop period	5
Momentum	0.9

Table 6. Performances of the pretrained networks for skin lesion classification.

Network name	Imbalanced data	Balanced data
	Accuracy [%]	Accuracy [%]
NasnetMobile	66.60	73.24
GoogleNet	68.06	72.20
MobileNetv2	67.43	71.99
ShuffleNet	69.73	71.16
EfficientNetb0	65.97	73.44
ResNet-18	72.03	74.27
EfficientNetb3	69.73	75.88

our mobile application with our models, we noticed a significant performance difference between the EfficientNetb3 and ResNet-18 models regarding inference times and frames per second (FPS) values. For example, in one of our devices, with no acceleration setting, we observed 401.49 and 91.63 ms inference times for EfficientNetb3 and ResNet-18 models, respectively. We also observed a 2.44 FPS value for the EfficientNetb3 model and a 10.16 FPS value for the ResNet-18 model. In addition, we did not notice any performance improvement in the XNNPack library acceleration setting with the EfficientNetb3 model. Lastly, we got an error and could not run our application when we used NNAPI delegate acceleration with EfficientNetb3.

Figure 5 shows the confusion matrix of the ResNet-18. The system successfully distinguishes diseases in general. Also, the performance rate of the Monkeypox class (indicated as 4 in the matrix) is quite high when compared to other classes. In the figure, the white squares represent 0, meaning no false prediction exists in the relevant cells. **Figure 6** shows some visual results on the confusion matrix of the ResNet-18.

The proposed system has also been compared to other studies that used the same PAD-UFES-20 dataset. Table 7 shows the comparison results in terms of precision, recall, F1-score, Jaccard, and accuracy index. For a more fair comparison, ResNet-18 has also been trained and tested using only six classes on PAD-UFES-20 dataset images. For six-class classification, the proposed system showed better results than Chen et al. and Haritha et al.[32,54] in terms of accuracy. It showed a comparable result with Pacheco et al. and Khan et al.[55,56] It also produced better and comparable results than other studies, despite the seven-class comparison, which included monkeypox images, unlike the others. ResNet18, ResNet50, and DenseNet121 are examples of deep networks class. ResNet18 has fewer parameters, while ResNet50 and DenseNet121 are deeper networks with more parameters. Having more parameters, generally, makes the model more complex and flexible. However, having more parameters also has disadvantages like more memory usage, slower training time, and more data requirements. Therefore, a scenario where ResNet18 outperforms ResNet50 and DenseNet121 might depend on the complexity and size of the dataset. Although ResNet18 has fewer parameters, it may have enough capacity to make a better generalization over the dataset. The information in Table 7 is an example of this situation.

In the literature, there are some other studies, such as Medhat et al. and Alyami et al.,[30,31] which used the same dataset. But, these studies selected some classes of the dataset and studied with fewer classes; some of them[32] merged visual data with clinical data. To ensure a fair comparison, these are not included in the comparison table.

4.3. Experiments with Android Application

The application has been delivered to four Android smartphones for the on-device experiments. The devices' information is shown in **Table 8**. It has been successfully run on each device. Some sample screenshots of the application are shown in **Figure 7**. The class of the skin lesion with the highest probability is shown at the top left corner of the screen, together with the probability percentage value. At the bottom of that information, the inference time and the FPS values are seen. Inference time is the time spent predicting a single image frame. FPS represents the number of image frames processed and predicted in a second.

For the quantitative evaluation of the mobile application, inference time and FPS performances have been observed. The performances were measured using different accelerators. TensorFlow Lite framework provides two main mechanisms for acceleration. These are the XNNPack library and delegates.

XNNPack library provides optimized implementations for floating-point neural network operators and, therefore, utilizes CPUs of smartphones for better performance. For example, it includes optimized convolution and fully connected layer operators for ARM processors that support ARM NEON extension. It also provides an operator fusion feature that combines possible operators into a single operator, resulting in improved performance.

Table 7. Comparative results for skin lesion classification task.

Study	Method	Dataset	#Class	Precision [%]	Recall [%]	F1-score [%]	Jaccard [%]	Accuracy [%]
[56]	EfficientNetB3 + noisy student	PAD-UFES-20	6	81.00	82.00	74.00	–	76.00
	EfficientNetB3	PAD-UFES-20	6	78.00	80.00	79.00	–	74.00
[32]	VGGNet19	PAD-UFES-20	6	54.72	53.11	50.00	–	55.75
	ResNet50	PAD-UFES-20	6	67.66	63.00	64.40	–	70.29
	DenseNet121	PAD-UFES-20	6	66.82	63.65	64.85	–	71.60
	InceptionV3	PAD-UFES-20	6	60.73	59.39	59.78	–	67.70
[55]	EfficientNetB4	PAD-UFES-20	6	–	–	–	–	74.40
	Densenet121	PAD-UFES-20	6	–	–	–	–	74.50
	Mobilenetv2	PAD-UFES-20	6	–	–	–	–	70.00
	Resnet50	PAD-UFES-20	6	–	–	–	–	73.20
	Vggnet13	PAD-UFES-20	6	–	–	–	–	74.90
[54]	Vggnet16 + SVM	PAD-UFES-20	6	–	–	–	–	53.24
	Vggnet16 + SVM	PAD-UFES-20	6	–	–	–	–	53.24
	Vggnet16 + MLP	PAD-UFES-20	6	–	–	–	–	61.00
	Mobilenetv2 + SVM	PAD-UFES-20	6	–	–	–	–	66.00
	Nasnetmobile+SVM	PAD-UFES-20	6	–			–	61.30
	Mobilenetv2 + MLP	PAD-UFES-20	6	–	–	–	–	61.00
Proposed study	ResNet-18	PAD-UFES-20	6	75.17	62.77	65.90	50.81	74.62
Proposed study	ResNet-18	MSLD* + PAD-UFES-20	7	76.78	71.40	73.63	59.68	74.27

*Only monkeypox-labeled images have been used from MSLD.

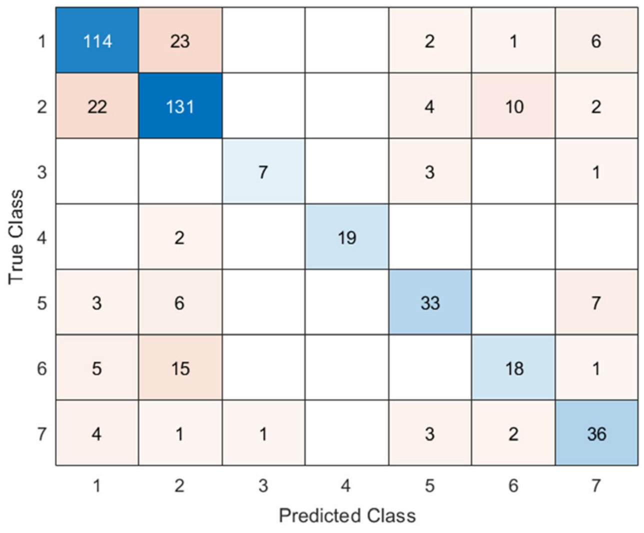

Figure 5. The confusion matrix of the ResNet-18 produced the best accuracy. (1:ACK, 2:BCC, 3:MEL, 4:MPX, 5:NEV, 6:SCC, 7:SEK).

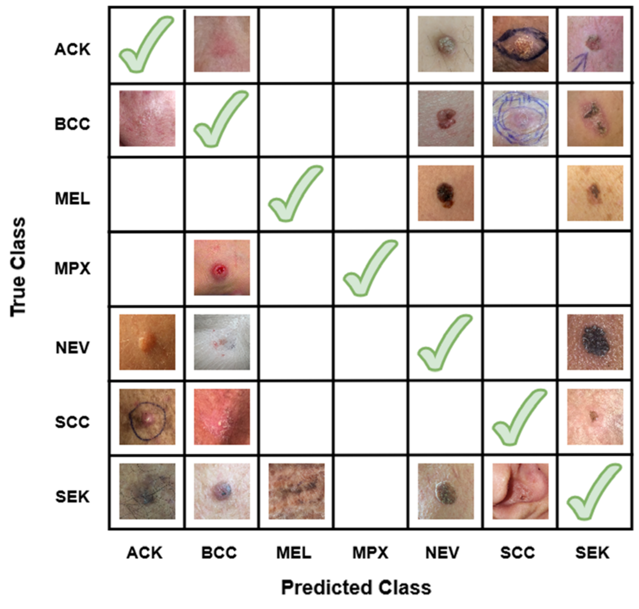

Figure 6. Some visual examples on the confusion matrix of the ResNet-18 produced the best performance.

Delegates allow the usage of hardware accelerators of smartphones during ML operations. TensorFlow Lite currently provides four delegates: 1) GPU delegate for Android and iOS platforms; 2) Hexagon delegate for Android platform; 3) Android neural networks API (NNAPI) delegate for Android platform; and 4) Core ML delegate for iOS platform. GPU delegate and Hexagon delegate allow the utilization of GPU and Qualcomm Hexagon DSP, respectively. NNAPI delegate enables the usage of device's GPU, DSP, and neural processing unit (NPU) for acceleration on Android platforms.

Table 8. Android devices used in the experiments.

Device	Android API	Android OS	RAM	ISA	CPU
Device-1	33	13	8 GB	ARMv8.2-A	Octa-core (4 × 1.8 GHz and 3 × 2.42 GHz and 1 × 2.84 GHz)
Device-2	29	10	4 GB	ARMv8.0-A	Octa-core (4 × 2.2 GHz and 4 × 1.8 GHz)
Device-3	33	13	4 GB	ARMv8.2-A	Octa-core (4 × 2.0 GHz and 4 × 2.0 GHz)
Device-4	25	7.1.1	3 GB	ARMv8.0-A	Octa-core (4 × 1.5 GHz and 4 × 2.0 GHz)

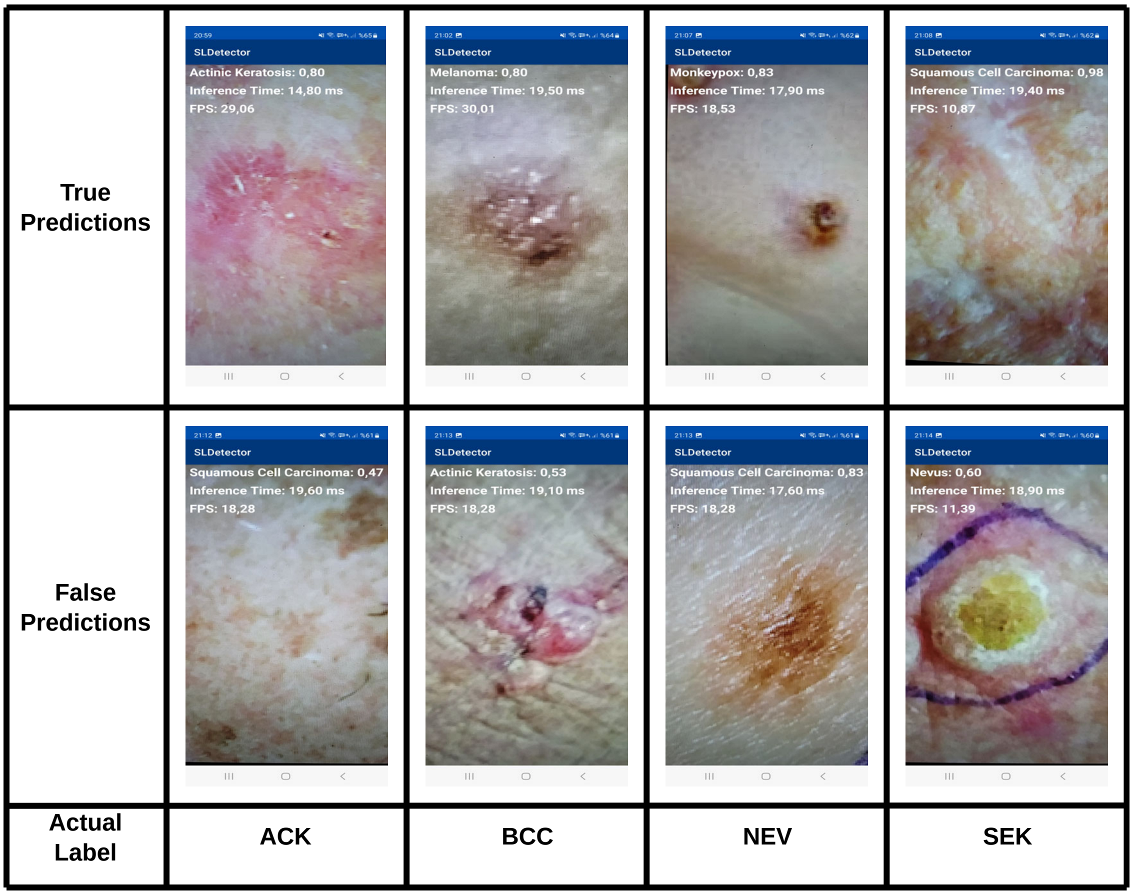

Figure 7. Some sample screenshots from the application.

Core ML delegate utilizes the neural engine of iOS platforms. During the experiments, both the inference time and FPS performances have been observed, with no acceleration, XNNPack library acceleration, and NNAPI delegate acceleration options on all smartphones.

The inference times have been measured to assess the single-frame prediction performances. They are shared in **Table 9** in milliseconds (ms). During the experiments, 1,000 inferences for each acceleration option and device have been measured, then the average inference time has been calculated. In the table, NA means not applicable, which states that the accelerator can not be used for that platform. According to these results, for Device-1, XNNPack provided 16.71%, and NNAPI provided a 40.48% increase in performance compared to no acceleration. NNAPI provided a 28.54% increase in performance compared to XNNPack. For Device-2, Device-3, and Device 4, XNNPack provided a 26.88%, 7.44%, and 23.77% increase in performance, respectively, compared to no acceleration.

The FPS values have been observed to get an idea of the application performance from the viewpoint of real-life experience.

Table 9. Inference time performances.

Device	No acceleration [ms]	XNNPack [ms]	NNAPI [ms]
Device-1	26.63	22.18	15.85
Device-2	91.63	67.00	NA
Device-3	135.68	125.58	NA
Device-4	249.91	190.50	NA

During the experiments, the time spent for 1,000 frames for each setting and device was measured, and then the average FPS was calculated. The FPS values are shared in **Table 10**. There are two types of FPS values: provided FPS and analyzed FPS. Provided FPS represents the number of frames delivered to the image analysis use case per second by the platform. Analyzed FPS represents the number of image frames processed and predicted per second. NA means not applicable, which states that the accelerator can not be used for that platform.

For Device-1, the provided FPS value and the analyzed FPS values for each acceleration option are approximately 29. No improvement is seen between different acceleration settings. This is because all of the provided image frames are processed and predicted even without any acceleration. The application can keep up with the provided image frames. For Device-2, the analyzed FPS values are lower than the provided FPS values. The application can not keep up with the provided image frames. When no acceleration option is selected, 48.97% of the image frames are dropped, and when XNNPack acceleration is set, 33.25% of the image frames are dropped. XNNPack resulted in a 30.80% increase in FPS performance compared to no acceleration. For Device-3, the analyzed FPS values are approximately 75.42% lower than the provided FPS values. Analyzed FPS values are approximately 7 with no acceleration and XNNPack acceleration options. Recall that XNNPack provided a 7.44% increase in inference time performance for this device. This increase did not result in significant improvement in FPS performance in our application. For Device-4, the analyzed FPS values are again lower than those provided. When no acceleration option is selected, 85.12% of the image frames are dropped, and when XNNPack acceleration is set, 69.86% of the image frames are dropped. XNNPack resulted in a 102.56% increase in FPS performance compared to no acceleration.

These results show that accelerators of the TensorFlow Lite improve inference time performance, which may also result in improved FPS performance. In addition, with modern, powerful smartphones, we can get real-time prediction performance.

To examine the effect of optimization approaches on accuracy, the samples in the test set were tested in all three optimization techniques. When the results were examined, the same label values were obtained with the same probabilities for many images in all three approaches. In a few examples, although the labels were the same, a 1%–2% difference was observed between the probability values, but in this case, the accuracy value still did not change. The results of some samples from the test set on the three approaches are given in **Table 11**.

Table 10. FPS performances.

Device	Provided FPS	Analyzed FPS		
		No acceleration	XNNPack	NNAPI
Device-1	29.50	29.46	29.50	29.59
Device-2	19.91	10.16	13.29	NA
Device-3	28.49	6.82	7.13	NA
Device-4	28.90	4.30	8.71	NA

5. Conclusion

This article presents a mobile system for skin disease classification. One of the focuses of the study is to distinguish monkeypox lesions from other lesions and quickly determine monkeypox lesions. For this purpose, a combined dataset has been created using two different datasets. The combined dataset consists of seven types of skin lesions, including monkeypox. Then, using this dataset, different pretrained networks were trained based on the transfer learning approach. In this step, small-size pretrained networks have been chosen to better adapt mobile applications. Then the most suitable network from the viewpoint of both performance and mobile compatibility has been transformed into the TensorFlow Lite format. Finally, a mobile application for Android platforms has been developed that uses the TensorFlow Lite ML library to classify skin lesion images. The development has been carried out using Kotlin. The CameraX API was used to gather the images through the smartphone's camera. The mobile application has been run successfully on different smartphones, and the performance results have been collected. Improved performance in inference times has been observed when accelerators are used. The proposed system allows quick preliminary diagnosis of the lesions of patients using their smartphones. The system produced 74.27% classification accuracy for seven-class. In addition, the same system has been also trained and tested with just PAD-UFES-20 dataset images for a more fair comparison with the literature. The system produced 74.62% accuracy for six-class PAD-UFES-20 dataset images. It yielded comparable/better results compared to the literature.

To supplement the testing with some real clinical data collected from hospitals is goal of future work. Also, it aims to cooperate with dermatologists and radiologists to receive feedback on both testing with new patient data and increasing the adoption and usefulness of the system. In addition, the authors are investigating multiplatform/cross-platform mobile development technologies including PWAs, especially from the viewpoint of ML framework integration. In the future, it is aimed to create mobile AI solutions that span a wide range of platforms using a single code base with the help of these technologies.

Table 11. The effect of optimization techniques on accuracy.

Sample	Actual label	Predicted label	Probability for no optimization	Probability for XNNPack	Probability for NNAPI
Sample 1	ACK	ACK	0.84	0.84	0.84
Sample 2	BCC	BCC	0.90	0.90	0.90
Sample 3	ACK	BCC	0.64	0.64	0.65
Sample 4	MEL	NEV	0.50	0.50	0.50

Acknowledgements

This study was supported by The Scientific and Technological Research Council of Turkiye - Turkish Academic Network and Information Center (TUBITAK-ULAKBIM) in the publication process.

Conflict of Interest

The authors declare no conflict of interest.

Data Availability Statement

The data that support the findings of this study are openly available in Mendeley and Kaggle. The data are derived from the following resources available in the public domain: https://data.mendeley.com/datasets/zr7vgbcyr2/1; https://www.kaggle.com/datasets/nafin59/monkeypox-skin-lesion-dataset.

Keywords

deep learning, mobile applications, monkeypox, skin lesions, smartphones

Received: April 25, 2023
Revised: August 24, 2023
Published online: September 19, 2023

[1] B. Wang, W. Wei, Z. Shao, Q. Qin, Z. Wang, Y. Jia, J. Guo, Y. Pang, L. Jiang, G. Jin, C. Mao, S. Liu, *Adv. Intell. Syst.* **2021**, *3*, 2100033.
[2] J. Yao, Z. Lei, W. Yue, B. Feng, W. Li, D. Ou, N. Feng, Y. Lu, J. Xu, W. Chen, C. Yang, L. Wang, L. Wang, J. Liu, P. Wei, H. Xu, D. Xu, *Adv. Intell. Syst.* **2022**, *4*, 2200100.
[3] S. Bhadula, S. Sharma, P. Juyal, C. Kulshrestha, *Int. J. Innov. Technol. Explor. Eng.* **2019**, *9*, 4044.
[4] V. Anand, S. Gupta, D. Koundal, S. R. Nayak, J. Nayak, S. Vimal, *Int. J. Artif. Intell. Tools* **2022**, *31*, 2250029.
[5] V. R. Allugunti, *Int. J. Comput. Program. Database Manag.* **2022**, *3*, 141.
[6] V. Anand, S. Gupta, S. R. Nayak, D. Koundal, D. Prakash, K. D. Verma, *Multimed. Tools Appl.* **2022**, *81*, 37379.
[7] C. Massone, A. D. Stefani, H. P. Soyer, *Curr. Opin. Oncol.* **2005**, *17*, 147.
[8] B. Dammak, M. Turki, S. Cheikhrouhou, M. Baklouti, R. Mars, A. Dhahbi, *Sensors* **2022**, *22*, 4.
[9] E. Kaur, P. Delir Haghighi, F. M. Cicuttini, D. M. Urquhart, *Sensors* **2022**, *22*, 18.
[10] D. Leskur, J. Bozic, D. Rusic, A. Seselja Perisin, T. Cohadzic, S. Pranic, D. Modun, J. Bukic, *Int. J. Med. Inf.* **2022**, *168*, 104895.
[11] M. Valero, J. Clemente, F. Li, W. Song, *Pervasive Mob. Comput.* **2021**, *75*, 101422.
[12] M. Zhong, J. Wen, P. Hu, J. Indulska, *Pervasive Mob. Comput.* **2017**, *38*, 60.
[13] M. Breen, T. Reed, H. M. Breen, C. T. Osborne, M. S. Breen, *Sensors* **2022**, *22*, 16.
[14] V. H. Sahin, I. Oztel, *Int. J. Eng. Res. Dev.* **2021**, *13*, 13.
[15] J. Lee, H. Ho, S. Yoo, Y. Kwon, H.-G. Sohn, *Int. J. Appl. Earth Obs. Geoinf.* **2022**, *111*, 102820.
[16] N. Khasawneh, E. Faouri, M. Fraiwan, *Appl. Sci.* **2022**, *12*, 17.
[17] R. Swetha, P. Bende, K. Singh, S. Gorthi, A. Biswas, B. Li, D. C. Weindorf, S. Chakraborty, *Geoderma* **2020**, *376*, 114562.
[18] T. D. Pham, Y. W. Lee, C. Park, K. R. Park, *Mathematics* **2022**, *10*, 9.
[19] A. G. Pacheco, G. R. Lima, A. S. Salomão, B. Krohling, I. P. Biral, G. G. de Angelo, F. C. Alves Jr, J. G. Esgario, A. C. Simora, P. B. Castro, F. B. Rodrigues, P. H. Frasson, R. A. Krohling, H. Knidel, M. C. Santos, R. B. do Espírito Santo, T. L. Macedo, T. R. Canuto, L. F. de Barros, *Data Br.* **2020**, *32*, 106221.
[20] A. G. Pacheco, R. A. Krohling, *Comput. Biol. Med.* **2020**, *116*, 103545.
[21] Kaggle, Monkeypox Skin Lesion Dataset, https://www.kaggle.com/datasets/nafin59/monkeypox-skin-lesion-dataset (accessed: 5 October 2022).
[22] S. N. Ali, M. T. Ahmed, J. Paul, T. Jahan, S. M. S. Sani, N. Noor, T. Hasan, arXiv:2207.03342, **2022**.
[23] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, et al., *Tensorflow: Large-scale machine learning on heterogeneous systems*, **2016**, https://doi.org/10.48550/arXiv.1603.04467.
[24] TensorFlow, TensorFlow Web Site, https://www.tensorflow.org (accessed: 5 October 2022).
[25] TensorFlow Lite, TensorFlow Lite ML for Mobile and Edge Devices, https://www.tensorflow.org/lite (accessed: 5 October 2022).
[26] N. Nida, A. Irtaza, A. Javed, M. H. Yousaf, M. T. Mahmood, *Int. J. Med. Inf.* **2019**, *124*, 37.
[27] R. Karthik, T. S. Vaichole, S. K. Kulkarni, O. Yadav, F. Khan, *Biomed. Signal Process. Control* **2022**, *73*, 103406.
[28] G. Zheng, M. Li, F. Zhang, B. Wang, Y. Ji, *J. Phys.: Conf. Ser.* **2022**, *2289*, 012027.
[29] P. N. Srinivasu, J. G. SivaSai, M. F. Ijaz, A. K. Bhoi, W. Kim, J. J. Kang, *Sensors* **2021**, *21*, 2852.
[30] S. Medhat, H. Abdel-Galil, A. E. Aboutabl, H. Saleh, *J. Radiat. Res. Appl. Sci.* **2022**, *15*, 262.
[31] J. Alyami, A. Rehman, T. Sadad, M. Alruwaythi, T. Saba, S. A. Bahaj, *Microsc. Res. Tech.* **2022**, *85*, 3600.
[32] Q. Chen, M. Li, C. Chen, P. Zhou, X. Lv, C. Chen, *J. Cancer Res. Clin. Oncol.* **2022**, *149*, 3287.
[33] M. M. Ahsan, M. R. Uddin, M. Farjana, A. N. Sakib, K. A. Momin, S. A. Luna, arXiv:2206.01862, **2022**.
[34] V. H. Sahin, I. Oztel, G. Y. Oztel, *J. Med. Syst.* **2022**, *46*, 79.
[35] R. Mamoun, M. Nasor, S. H. Abulikailik, in *2020 Int. Conf. on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE)*, ISBN 978-1-7281-9111-9, Khartoum, Sudan **2021**, pp. 139–144.
[36] N.-B. Cho, S.-R. Cho, S. H. Choi, H. You, S. I. Nam, H. Kim, *Commun. Sci. Disord.* **2021**, *26*, 219.
[37] P. Watts, P. Breedon, C. Nduka, C. Neville, V. Venables, S. Clarke, *J. Med. Syst.* **2020**, *44*, 149.
[38] J. Berger-Groch, M. Keitsch, A. Reiter, S. Weiss, K. Frosch, M. Priemel, *J. Med. Syst.* **2021**, *45*, 99.
[39] R. Bao, N. M. Al-Shakarji, F. Bunyak, K. Palaniappan, in *Proc. of the IEEE/CVF Int. Conf. on Computer Vision (ICCV) Workshops*, Montreal, QC, Canada **2021**, pp. 3361–3370.
[40] Z. Al-Milaji, I. Ersoy, A. Hafiane, K. Palaniappan, F. Bunyak, *Pattern Recognit. Lett.* **2019**, *119*, 214.
[41] A. Hamad, I. Ersoy, F. Bunyak, in *2018 IEEE Applied Imagery Pattern Recognition Workshop (AIPR)*, ISBN 978-1-5386-9306-3, Washington, DC, USA **2018**, pp. 166–171.
[42] M. M. H. Shuvo, Y. M. Kassim, F. Bunyak, O. V. Glinskii, L. Xie, V. V. Glinsky, V. H. Huxley, M. M. Thakkar, K. Palaniappan, in *2020 25th Int. Conf. on Pattern Recognition (ICPR)*, ISBN 978-1-7281-8808-9, Virtual Event/Milan, Italy **2021**, pp. 4317–4323.
[43] I. Goodfellow, Y. Bengio, A. Courville, *Deep Learning*, MIT Press, USA **2016**, http://www.deeplearningbook.org

[44] TensorFlow Guide, Transfer learning and fine-tuning, https://www.tensorflow.org/guide/keras/transfer_learning (accessed: 20 July 2022).
[45] Google, Android OS, https://www.android.com (accessed: 5 October 2022).
[46] Google, Mobile App Developer Tools - Android Developers, https://developer.android.com (accessed: 5 October 2022).
[47] Google, React Native, https://flutter.dev (accessed: 5 October 2022).
[48] JetBrains, Kotlin Multiplatform Mobile, https://kotlinlang.org/docs/multiplatform-mobile-getting-started.html (accessed: 5 October 2022).
[49] Meta Platforms, React native, https://reactnative.dev (accessed: 5 October 2022).
[50] Mozilla, Progressive web apps (PWAs) MDN, https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps (accessed: 5 October 2022).
[51] WHO, Multi-country monkeypox outbreak in non-endemic countries: Update, 29 May **2022**, https://www.who.int/emergencies/disease-outbreak-news/item/2022-DON388 (accessed: 20 July 2022).
[52] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, in *2009 IEEE Conf. on Computer Vision and Pattern Recognition*, Miami, FL, USA **2009**, pp. 248–255.
[53] TensorFlow, Tensorflow lite models, https://www.tensorflow.org/lite/models?hl=en.
[54] D. Haritha, B. Sandhya, in *Proc. of the 2022 9th Int. Conf. on Computing for Sustainable Global Development, INDIACom 2022*, Institute of Electrical and Electronics Engineers Inc., ISBN 9789380544441, New Delhi, India **2022**, pp. 500–505.
[55] A. G. C. Pacheco, R. A. Krohling, *IEEE J. Biomed. Health Inform.* **2021**, *25*, 3554.
[56] I. U. Khan, N. Aslam, T. Anwar, S. S. Aljameel, M. Ullah, R. Khan, A. Rehman, N. Akhtar, *Complexity* **2021**, *2021*, 5591614.