Proceedings of the 9th World Congress on Electrical Engineering and Computer Systems and Sciences (EECSS'23)
Brunel University, London, United Kingdom - August 03-05, 2023
Paper No. MVML 116
DOI: 10.11159/mvml23.116

Deep Learning Mobile Algorithms for Detection of Skin Cancer

Iren Valova[1], **Peter Dinh**[2], **Natacha Gueorguieva**[3]
[1]Department of Computer and Information Science, University of Massachusetts Dartmouth
285 Old Westport Rd., Dartmouth, MA, USA
Iren.Valova@umassd.edu
[2]Computer Systems Technology Department, New York College of Technology/City University of New York
300 Jay Street, Brooklyn, NY, USA
Peter.Dinh20@citytech.cuny.edu
[3]Department of Computer Science, College of Staten Island/City University of New York
2800 Victory Blvd, NY, USA
Natacha.Gueorguieva@csi.cuny.edu

Abstract **-** Skin cancer malignant melanoma is the deadliest type of cancer and early detection is important to improve patient prognosis. Recently, Deep Learning Neural Networks (DLNNs) have proven to be a powerful tool in classifying medical images for detecting various diseases and it has become viable to deal with skin cancer detection. In this research we propose a serverless mobile app to assist with skin cancer detection. This mobile app is based on the best performance of five Convolution Neural Network (CNN) models designed from scratch as well as four state-of-the-art architectures used for Transfer Learning (Inception v3, ResNet50v2, DenseNet, and Exception v2). Since the skin cancer dataset is imbalanced, we perform data augmentation. We also use the fine-tuning top layers technique for feature extraction on all models to improve the results. The main novelty of the proposed method is the model deployed as part of mobile app where the classification processes are executed locally on the mobile device. This approach will reduce the latency and improve the privacy of the end users compared with the cloud-based model where user needs to send images to a third-party cloud service. The achieved accuracy of pre-trained Inception v3 model is 99.99%. Therefore, the proposed mobile solution can serve as a reliable tool that can be used for melanoma detection by dermatologists and individual users.

Keywords**:** melanoma, skin lesion, deep learning, machine learning, transfer learning, on-device inference app, mobile app, convolution neural networks.

1. Introduction

Skin cancer is the most common type of cancer in the US. Statistically, 1 out of 5 people will get skin cancer at some point in their life [1]. The vast majority of skin cancer deaths are due to malignant melanoma. Most skin cancers either spread to other parts of the body and or are fatal unless detected and treated early. In its initial stages, malignant melanoma is completely curable with a simple biopsy [2]. Therefore, an early detection is the best solution to improve skin cancer prognostic. However, if the diagnosis and treatment begin in late stages, it usually takes long time to get the results. Unfortunately, many people do not want to go to the doctor during the preliminary stages when it can be prevented. Thus, there is a need for a mHealth app – a self-examination tool for individual user with a standard camera from the smartphones through installation of an app on mobile device that can perform a quick, efficient, and low-cost diagnosis.

Convolution Neural Networks (CNNs) have been applied to many fields as image recognition and object detections as well as in new research domains. Computer-aided systems for skin lesion diagnosis is a growing area of research. Recently, researchers have shown an increasing interest in developing computer-aided diagnosis systems. There are several methods of deep learning but the pre-trained deep learning models and handcrafted methods that are based on a deep learning approach already show promising results with high-precision accuracy for melanoma detection [3].

Mobile health defined as the use of mobile devices with machine learning, is a rapidly growing field in the digital health sector providing healthcare support and intervention. It has the potential to reduce the cost of health care and to improve the well-being in numerous ways [4]. Machine Learning requires a large amount of storage and system processing which is a challenging for mobile applications. Many mHealth applications deploy Machine Learning model on a cloud-based server but this approach has some disadvantages as the time it takes for a cloud-based server to respond to a client request and the

privacy issues which might arise when sending sensitive medical data. Researchers in [5] presented the concept of a system that consists of a smartphone application available to the client (client side) and the server side of the system, consisting of a general-purpose server, computing server, users database, and images database.

In this research we propose an on-device inference app to assist with skin cancer detection. The app will use on-device pre-trained classification model to perform classification of new data. The classification process would use a pre-trained CNN model using 25,331 dermoscopic images generated by the International Skin Imaging Collaboration (ISIC) [6]. As the dataset is small and imbalanced we implemented data augmentation to avoid overfitting.

Our model is deployed on mobile app, where the inference process takes place. When presented with new image all computations are executed locally where the test images remain. This approach will reduce the latency and improve the privacy of the end users compared with the cloud-based model where the user needs to send images to a third-party cloud service. Figure 1 demonstrates the on-device inference for Skin Cancer Detection app process.

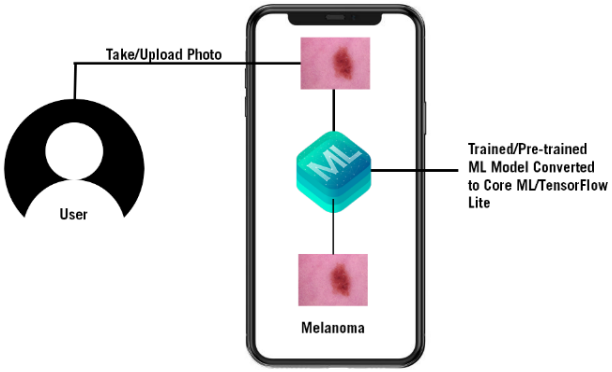

Fig 1. On-device Machine Learning Mobile App for Skin Cancer Detection

2. Application Design and Methodology

We design an on-device Machine Learning mobile app to support the diagnosis of skin cancer using CNN based on the best performance of five CNNs primary based models such as Inception v3, ResNet50v2, DenseNet, and Exception v2. The above models represent pre-trained networks for object classification and detection over a large dataset. Training the models is done on the basis of Transfer Learning using a dataset of 25,331 dermoscopic images generated by the International Skin Imaging Collaboration (ISIC) [6]. We convert the best performance model to Core ML and TensorFlow Lite model and import it to the iOS and Android app using Xcode, Android Studio Integrated Development Environment. This methodology has many benefits namely the computation are performed on the device, the data process is handled on the user's device, and the app does not rely on network connectivity to function.

2.1. Convolutional Neural Networks (CNNs)

The Convolutional Neural Networks (CNNs) is a special type of Deep Learning Network (DNN). The architecture of CNNs is different from other deep learning models. There are two special aspects in the architecture of CNN, i.e., local connections and shared weights. CNNs exploits the local correlation using local connectivity between the neurons of near layers. CNNs use a special architecture which is well-adapted to classify images. This architecture makes convolution network fast to train so we can train many-layer networks which are exceptionally good at image classification [7].

When CNNs process an image, it utilizes convolutional operations to extract useful information from its pixels. The input image, when passed through a convolution layer, produces set of output images, known as output feature maps. The output feature maps detect features such as edges and color composition variation. The next layer will detect slightly more complicated features, such as shapes and other geometric structures. The deeper layers the network has the more convolution layers learn to detect more complicated features. The final layers of the CNNs detect specific things such as cars, horse, and

cats. The output layer of the CNNs provides a table of numerical value which is the probability that a specific object was discovered in the image.

In this research, the dataset contains skin images divided into eight categories (type of skin lesions) labels and the CNN learns the features of respective types. The CNN model process the input with a series of convolution, pooling and sub-sampling layers followed by fully-connected layers. The convolution and pooling layers perform feature extraction by capturing typical characteristics of the images. The fully connected layers assign a probability for the input image according to the given features.

2.2. Transfer Learning

Presented CNN methods that use transfer learning and parameters fine-tuning for skin lesion classification are the most common and best-performing approaches [3]. We also adopted a similar approach using transfer learning to train CNN models. Transfer Learning is a Machine Learning (ML) method where a model developed for one task is reused for another model on different task. In recent years, Transfer Learning has gained a lot of popularity for image classification tasks because of reasonable training time and improved baseline and final performance.

In a typical image classification, the neural networks try to determine the edges in the initial level layers, shapes in the middle level layers and all specific features in the final layers. However, in Transfer Learning, we use the initial and middle level layers and only retrain the final layers [8]. Moreover, we need a lot of data to train the model but most of the time the size of the data is not big enough and it takes more time to train CNNs from scratch. Therefore, it is common to use pre-trained network on a large dataset and then use it as an initialization or a fixed feature extractor for the tasks of interest [9]. The most common ways to use Transfer Learning are feature-extraction and fine-tuning. Feature extraction serves the purpose of assembling information into meaningful observations, like identifying and reducing the presence of certain edges, shapes, or color patterns [10]. A well-tuned feature extraction layers can identify and amplify characteristics that are relevant to the problem. The role of feature extraction and feature interpretation are controlled by convolutional and fully-connected layer.

2.3. On-Device Machine Learning

Typically, Machine Learning models deploy on Cloud-based servers. However, On-Device Machine Learning is when the app performs interference with models directly on a mobile device. The ML model processes input data such as images, text, or sound on the mobile device rather than sending those inputs to a server and all computations are executed there.

As mobile devices become more powerful, running Machine Learning on-device has many benefits as: a) Low latency: the computation has happened on the device; therefore, there is no round trip to the server and no need to wait for the results to come back from the server to the local device. b) Privacy: with on-device Machine Learning, the data process can happen on the user's device. c) Works offline: since the Machine Learning model has been deployed on the mobile device, the app does not rely on network connectivity to function. d) No cost: the app has been deployed on the mobile device and it does not need to host any additional servers or paying for cloud computers. Therefore, the app will only use the processing power on the device.

There are some limitations of on-device Machine Learning because mobile devices are more restricted in terms of storage, memory, power consumption, and compute resources [11]. Apple's Core ML is a framework to help integrate Machine Learning models into the mobile app. It is not only just a runtime environment but also a file format. This format will be integrated into an iOS app. The app uses Core ML APIs and user data to make predictions and to train or fine-tune the model [12]. The framework supports the classification of images, natural language for processing text, speech for converting audio to text, and identifying sounds on audio. Alternatively, we can use Core ML tools to convert other Machine Learning models into Core ML format and once the model deployed on a user's device, we can use Core ML to retrain or fine-tune it on-device, with that user's data.

TensorFlow is one of the Google's deep learning libraries with several components: 1. Tensor is the core framework that every computation in TensorFlow involves tensors, which are the matrices/vectors of the n-dimensions. 2. Each value in a tensor shares an identical datatype or shape. 3. TensorFlow uses a graph framework to gather in all series computations that

it has done through the training and describe them. The graph can run on multiple GPUs, CPUs, or mobile OS. TensorFlow Lite is a mobile library for deploying models on mobile, microcontrollers and other edge devices [13]. We can train a new model or select a pre-trained model and convert the model to a TensorFlow model with the TensorFlow Lite Converter. The compressed *.tflite* file can be deployed into a mobile or embedded device [14].

3. Experiments: Dataset and Image Preprocessing

We build a CNN model from scratch and four Transfer Learning models based on the Inception v3, ResNet50v2, DenseNet, and Exception v2. Since the dataset is imbalanced and small, we apply data augmentation such as random rotation of 180 degrees, random brightness change, and zoom augmentation to create new images from existing ISIC dataset [6]. Moreover, to achieve the best performance of the training model we apply fine-tuning of parameters such as layer freezing, Drop Learning Rate on Plateau. The fully-connected layer is retained in the instantiated pre-trained model.

We use a dataset of 25,331 dermoscopic images was generated by the International Skin Imaging Collaboration (ISIC) [6, 15, 16]. The dataset consists of eight categories of skin lesions: Melanocytic nevi, Melanoma, Basal cell carcinoma, Benign keratosis-like lesions, Actinic keratoses, Squamous cell carcinoma, Dermatofibroma, and Vascular lesions (Table 1). All skin cancer images are in 24-bit JPEG format and EXIF tags within the images were removed. Dimensions of these images are inconsistent ranging from 600 × 450 to 1024 × 1024. Figure 2 shows the samples of dermoscopic images.

Table 1. Dataset Skin Lesions Images

Skin Lession Type	Abbreviation	Total
Melanocytic nevi	NV	12875
Melanoma	MEL	4522
Benign keratosis-like lesions	BKL	2624
Basal cell carcinoma	BCC	3323
Actinic keratoses	AK	867
Squamous cell carcinoma	SCC	622
Dermatofibroma	DF	239
Vascular lesions	VASC	253

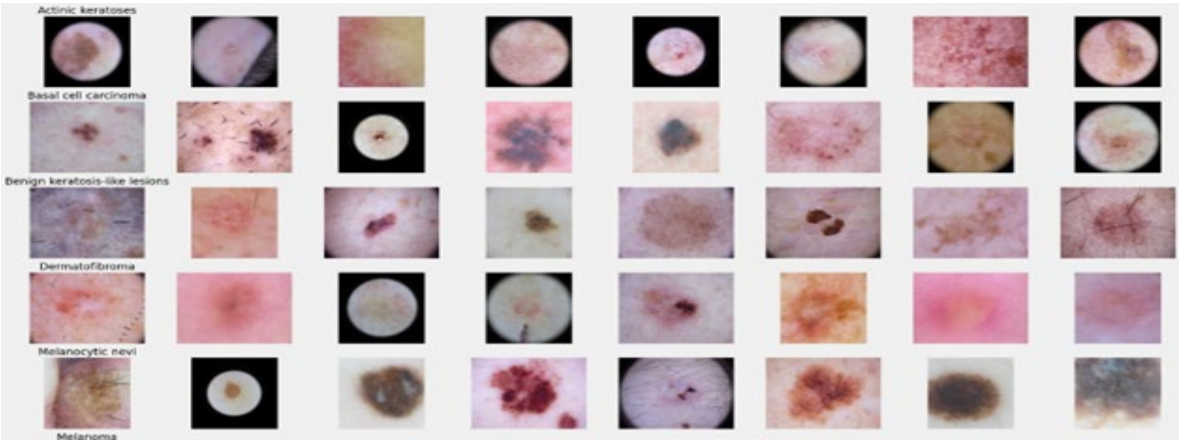

Fig 2. Samples for each type of skin lesion

Each skin lesion image has information in the metadata table, which includes all the fields such as lesion identifier, image, age, site, sex, and cell type. The dataset is imbalanced because of unequal distribution of samples. There are 12875 images in Melanocytic nevi. However, there are only 253 images in Vascular lesions.

Image data augmentation is a technique to artificially create new training data from the existing datasets. It transformed the images in the training dataset that belong to the same class as the original one. The transformation includes a range of image manipulation such as shifts, flips, zoom, and much more. We applied random rotation of 180 degrees, vertical and horizontal shift, vertical and horizontal flip, random brightness change, and zoom augmentation. Furthermore, images were resized to the input size of the model.

3.1. Fine-tune Parameters

Fine-tuning makes small adjustments to a process to achieve the desired performance. The basic pipeline involves replacing the last "classifier" layer of a pre-trained network with a new randomly initialized layer for the target task of interest. The modified network is then fine-tuned with additional passes of appropriately tuned gradient descent on the target training set [17]. We use layer freezing technique to fine-tune the training. When a layer is frozen, its weighs are fixed and it cannot be trained. Therefore, it is common to freeze the convolutional layers of the pre-trained model and to train only the custom fully-connected layers. By doing this, the feature-extracting knowledge is learned and kept in the weights of the convolutional layers stored, but the interpretative fully-connected layers are trained to best interpret the extracted features.

We also utilize the important parameter, "include_top" of the Keras pre-trained models to implement the first method of changing pre-trained model architecture for the training. When the parameter "include_top" is set to True, the fully-connected layer will be retained in the instantiated pre-trained model object. Therefore, the parameter, "include_top" is set to True to all models.

In this research we apply the Drop Learning Rate on Plateau to drop the learning rate by a factor after no change in a monitored metric for a given number of epochs. The "patience" values, which are the number of epochs to wait for a change before dropping the learning rate is 2. We use a default learning rate of 0.01 and drop the learning rate by setting the "factor" to 0.1 for all models.

4. Model Training and Tools

The core model is implemented on Windows OS using Python, Keras, TensorFlow and several other popular Machine Learning tools and frameworks to build the mobile app such as Android Studio, Xcode IDE. The specification of the computer on which trainings and testing took place includes Intel(R) Core (TM) i7-9700F CPU @3.00 GHz, 48 GB installed RAM and NVIDIA GeForce 3060 with 12 GB of memory.

Four state-of-the-art CNN architectures are used for Transfer Learning (Table 2). These models were pre-trained on ImageNet and then fine-tuned on the training set. All models followed the same setup with learning rate of 0.01 excluding the basic CNN model (0.001) for number of epochs - 200. Additionally, all image points outside the boundaries of the inputs are filled from the nearest available pixel of the input. During the training, the two final fully-connected layers of the network are removed, and the new dense layer is added to train the images of all models to fine-tune the model. Also, the class weighted cross-entropy loss is applied during the training to mitigate the data imbalance problem.

Table 2. Five Convolutional Neural Network Models

Model	Input Size	Number of Layers	Number of Parameters
Basic CNN Model	224 x 224	9	11,955,369
Inception v3	299 x 299	314	21,821,225
Xception v2	299 x 299	135	20,879,921
DenseNet201	224 x 224	710	20,242,984
ResNet50 v2	224 x 224	567	58,350,089

4.1. Metrics

One of the main goals of an automatic skin lesion classification application is acquiring specific information and treatment options for a lesion, and achieving this goal depends on correct diagnosis out of multiple categories. We use different models (Table 2) for detecting eight categories of skin lesions (Table 1). The purpose of these experiments is find the best fit for our scenario. The classification report which gives us insight towards precision, recall, and F1 score the first step of the evaluation process. Plotting of training accuracy, validation accuracy, training loss, and validation loss helps us in choosing the final model. Experimental results are shown in Fig. 3.

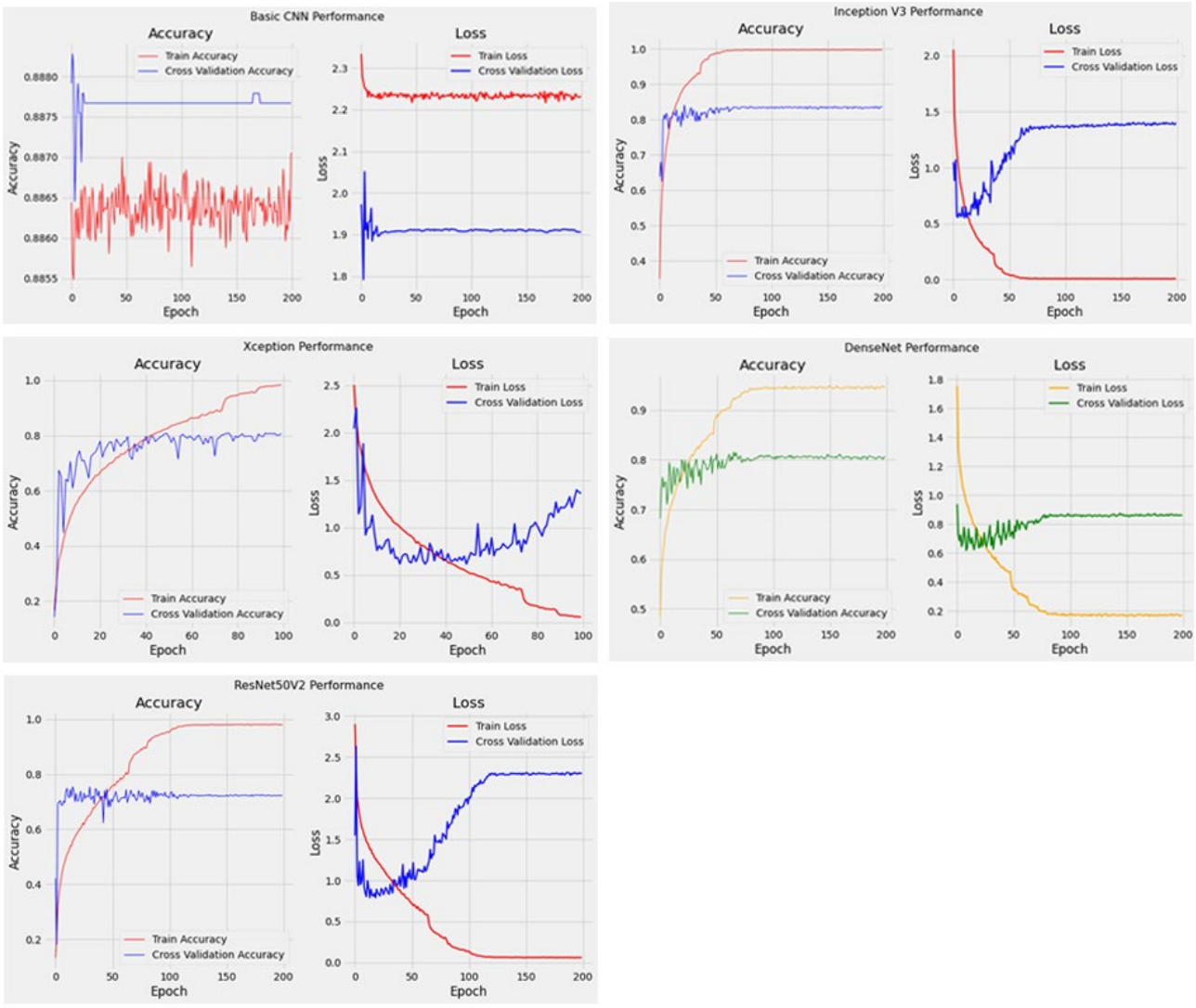

Fig 3. Accuracy and loss for training and validation of five models

5. Model Conversion and Intergrading the Pre-trained Model

Inception v3 model is the best model among utilized, with 99.99% accuracy and 83 for F1 score. Moreover, it achieves 91% of precision and 95% of recall classifying Melanocytic nevi. Therefore, we use this model and convert it to both Core ML and TensorFlow Lite model in order to build an iOS and an Android app.

5.1. Core ML and TensorFlow Lite Conversions

Apple's Core ML is a framework to help integrate Machine Learning models into the mobile app. We can integrate the trained Machine Learning models into iOS app. Figure 4 shows the process of the conversion. The pre-trained model in HDF5 (Hierarchical Data Formats) format will go to a Core ML converter and the output is the Core ML model format (*mlmodel*).

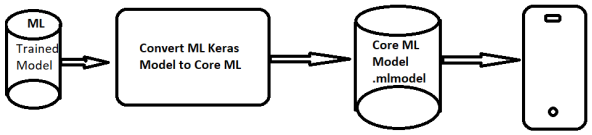

Fig 4. Model Conversion: Keras ".h5" to Apple Core ML ".mlmodel"

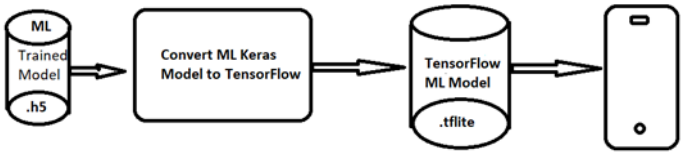

Fig 5. Model Conversion: Keras ".h5" to Google TensorFlow ML ".tflite"

The TensorFlow Lite converter takes a TensorFlow model and generates a TensorFlow Lite model (an optimized FlatBufffer identified by the "*.tflite*" file extension) [18]. Figure 5 demonstrates the conversion process of the Keras model to the TensorFlow model.

6. Skin Cancer Detector app on iOS and Android

We use the model with the best performance from the Table 3 as a base model to convert to both "*mlmodel*" and "*.tflife*" in order to build iOS app and Android app to demonstrate the process of using on-device inference approach, which has number of benefits over cloud-based solutions. A user can either provide the photo of skin image by selecting existing photo or can take a new photo as an input. The photo classified is based on a probability distribution of the output. It matches with the labels which listed all categories of skin lesions and then displays it on the screen.

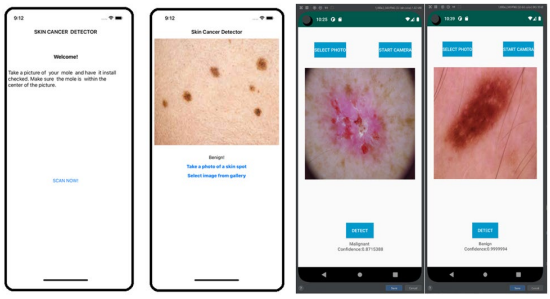

Fig. 6. Demonstration of Skin Cancer Detection on iOS and Android Device

7. Result and Analysis

We trained five different neural networks and their difference is the number of frozen top layers for feature extraction. The results summarized in Table 3 show that model Inception v3 has the best performance while both Xception v2 and ResNet50 v2 performed quite similar. The basic CNN model has the lowest performance. Therefore, Inception v3 is chosen as base model for our skin cancer detection algorithm. This model is converted to Core ML and TensorFlow Lite model to deploy in the mobile app.

Table 3. Results of all models with fine-tuning top layers

Model	Training Accuracy	Training Loss	Validation accuracy	Validation Loss	F1 Score
CNN Model	0.8868	1.954	0.8865	2.051	0.02
Inception v3	0.9999	0.0007	0.8349	1.382	0.83
Xception v2	0.9948	0.017	0.8068	1.362	0.81
Dense Net	0.9836	0.058	0.8068	0.857	0.81
ResNet50 v2	0.9972	0.017	0.7223	2.311	0.72

8. Conclusion

In this research we propose a mobile app to support the diagnostic of skin cancer using CNNs. Building a successful CNN model from scratch requires better architectures and training on large datasets interactively. Instead we utilize the knowledge learned in pre-trained models to design a new model. Since the pre-trained models are trained on large datasets, they have learned the basic features such as corners, edges etc. Therefore, to select a base model, we chose one CNN model from scratch and then use four state-of-the-art architectures for Transfer Learning. These are evaluated based on accuracy, precision, and recall of the model. We selected Inception v3 architecture with the best performance having 91% precision and 95% when recall classifying Melanocytic nevi.

The results obtained from the models show that Inception v3 model achieves a performance of 99.99% accuracy and results comparable with results presented in literature. Our research will be extended by Mask R-CNN to solve instance segmentation of cancer cells in the images in order to improve the results further.

References

[1] A. C. Society, "Cancer Facts & Figures 2018," Atlanta, American Cancer Society, 2018.

[2] American Cancer Society: Survival rates for melanoma skin cancer, by stage (2016), https://www.cancer.org/cancer/melanoma-skin-cancer/detection-diagnosis-staging

[3] M. Kassem, K. Hosny, R. Damaševičius, M. Eltoukhy , "Machine Learning and Deep Learning Methods for Skin Lesion Classification and Diagnosis: A Systematic Review", *Diagnostics* (Basel). 2021 Jul 31;11(8):1390. doi: 10.3390/diagnostics11081390. PMID: 34441324; PMCID: PMC8391467.

[4] Y. Park, "Emerging New Era of Mobile Health Technologies", *Healthcare Informatics Research* vol. 22,4 (2016): pp. 253-254. doi:10.4258/hir.2016.22.4.253

[5] G. Ahsan, I. Addo, S. Ahamed, D. Petereit et all, "Toward an mHealth, Intervention for Smoking Cessation", Proc COMPSAC. 2013:10.1109/COMPSACW.2013.61.

[6] M. Combalia, C. Noel, C. Codella, *et all*: "BCN20000: Dermoscopic Lesions in the Wild", 2019; arXiv:1908.02288.

[7] L. Deng, Y. Dong, "Deep Learning: Methods and Applications", *Foundations and Trends® in Signal Processing*: Vol. 7, 2014: No. 3–4, pp 197-387. http://dx.doi.org/10.1561/2000000039

[8] A. M. Nielsen, "Neural Networks and Deep Learning," *Determination Press*, 2015.

[9] N. Karthikeyan, "Machine Learning Projects for Mobile Application," *Packt*, 2018.

[10] K. Weiss, T., Khoshgoftaar, D.Wang, "A Survey of Transfer Learning", *Journal of Big Data*, 2016.

[11] Y. Andre, "Modern Deep Learning Design and Application Development," Apress, 2022.

[12] On-Device Learning Machine Learning, Google. https://developers.google.com/learn/topics/on-device-ml/

[13] Core ML, Integrate Machine Learning Models into Your App. Apple. https://developer.apple.com/documentation/coreml

[14] Tensorflow. https://www.tensorflow.org/

[15] P. Tschandl, C. Rosendahl, H. Kittler, "The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions", *Sci. Data* 5, 2018, 180161 doi.10.1038/sdata.2018.161.

[16] C. Noel, D. Gutman, et all, "Skin Lesion Analysis Toward Melanoma Detection: A Challenge at the 2017", International Symposium on Biomedical Imaging (ISBI), (ISIC)", 2017; arXiv:1710.05006.

[17] Y. Wang, D. Ramanan, M. Hebert, "Growing a Brain: Fine-Tuning by Increasing Model Capacity", Robotics Institute, Carnegie Mellon University

[18] TensorFlow Lite Converter. https://www.tensorflow.org/lite/convert