Machine Learning on Mobile: An On-device Inference App for Skin Cancer Detection

Xiangfeng Dai
School of Medicine
The University of North Carolina at Chapel Hill, Chapel Hill, USA

Irena Spasić
School of Computer Science & Informatics Cardiff University
Cardiff, UK

Bradley Meyer
Applied Analytics Program
Columbia University
New York, USA

Samuel Chapman
Booz Allen Hamilton Incorportion
Washington DC, USA

Frederic Andres
Digital Content & Media Sciences Research Division
National Institute of Informatics
Tokyo, Japan

***Abstract*— Mobile health (mHealth) is considered one of the most transformative drivers for health informatics delivery of ubiquitous medical applications. Machine learning has proven to be a powerful tool in classifying medical images for detecting various diseases. However, supervised machine learning requires a large amount of data to train the model, whose storage and processing pose considerable system requirements challenges for mobile applications. Therefore, many studies focus on deploying cloud-based machine learning, which takes advantage of the Internet connection to outsource data intensive computing. However, this approach comes with certain drawbacks such as those related to latency and privacy, which need to be considered in the context of sensitive data. To tackle these challenges of mHealth applications, we present an on-device inference App and use a dataset of skin cancer images to demonstrate a proof of concept. We pre-trained a Convolutional Neural Network model using 10,015 skin cancer images. The model is then deployed on a mobile device, where the inference process takes place, i.e. when presented with new test image all computations are executed locally where the test data remains. This approach reduces latency, saves bandwidth and improves privacy.**

Keywords— Machine Learning, mHealth, CNN, Image Classification, Convolutional Neural Networks, Deep Learning, Skin Cancer, Health Informatics

I. Introduction

Over the past decade, machine learning has proven to be a powerful tool in classifying medical images for detecting various diseases [23][24][25][26][27]. Mobile health (mHealth), the confluence of machine learning in health and mobile devices, is rapidly evolving in various medical specialties [9][14][31]. mHealth is considered one of the most transformative drivers for ubiquitous health informatics delivery of medical applications. mHealth applications of machine learning are transforming our lives at an extraordinary pace [30][31][32].

Supervised machine learning requires a large amount of data to train a classification model, whose storage and processing pose considerable system requirements challenges for mobile applications. Because of this, many mHealth applications deploy machine learning models on the cloud. For example, Ruiz-Zafra *et al*. [10] presented a platform for developing cloud-based mHealth apps, which are used for patient monitoring, information collecting and remote diagnosing. Gatsios *et al*. [11] combined machine learning algorithms, mobile technology and cloud-based approaches to self-manage the conditions of Parkinson patients. A similar study by Pan *et al*. developed a cloud-based mHealth app to collect quantitative information and monitor symptoms of Parkinson patients [13]. Melillo and Scala [12] designed a cloud-based mHealth platform to provide remote monitoring and clinical decision for hypertensive patients. Similarly, Ahsan *et al*. [33] implemented an mHealth app that uses machine learning algorithms to analyze patients' smoking behavior and interacts with their cloud-based data store to support smoking cessation.

These cloud-based approaches offload inference execution to the cloud, but this comes with multiple drawbacks related to:

- **Latency**: It takes time for a cloud-based service to respond to a client request.
- **Privacy**: Privacy issues might arise by sending sensitive data to the cloud, especially in the context of medical and health applications.
- **Cost**: Cloud-based approaches incur financial costs from the cloud service providers.
- **Connectivity**: Network connection is essential to run the cloud-based app, but cloud-based service may not always be available.
- **Customization**: In general, the cloud services provide generic models based on their common datasets. These models may not be appropriate or customizable for specific health problems.

In this study, we propose an on-device Inference App, where the classification model is pre-trained and stored on a mobile device, where it is used to perform classification of new data, which, consequently, does not need to be shared externally. This approach reduces latency, saves bandwidth and improves privacy. We demonstrate the basic principles of our approach including its evaluation using a case study, which focuses on skin cancer - one of the most common human malignancies [5][6][7][8].

II. Application Design and Methodology

A. *On-device Inference*

Supervised machine learning has two phases: training and inference. The challenge of performing machine learning on a mobile device is that training can often be computationally heavy and can take days or even months. In this study, we solved this problem by performing the training phase on a powerful computer and the inference phase on a mobile device (Figure 1).

Having trained a model, we then deployed it on a mobile device. Our App runs all the inference computations locally. Efficiency is the primary reason to perform inference directly on a mobile device. Not having to communicate with the cloud server, the App is able to classify new data almost instantaneously. Moreover, patients do not need to send their data including any Protected Health Information (PHI) over to the cloud server, thus eliminating any risk of a privacy breach.

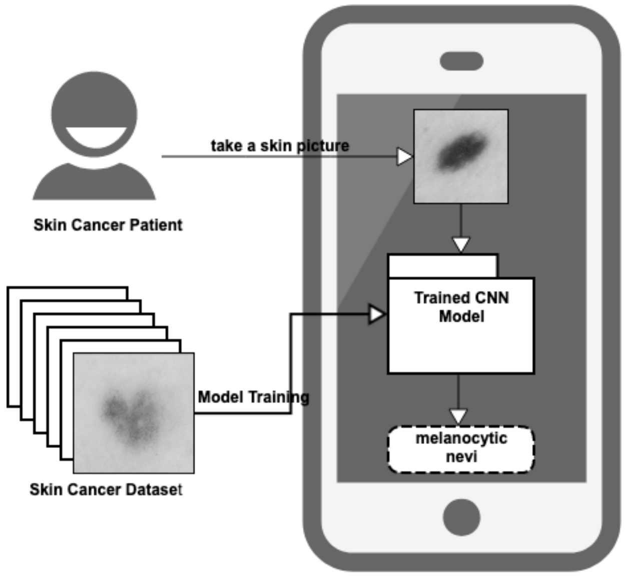

Fig. 1. On-device Inference App for Skin Cancer Detection

In our case study, the training dataset includes images of skin lesions, which have been labelled with respect to their type. The classification model has been trained to differentiate between various types of skin lesions. In the inference phase, new skin image, possibly taken by a mobile device, is presented to the model, which is then used to classify it with respect to its type.

B. *Convolutional Neural Networks*

Convolutional Neural Networks (CNN) are a subclass of neural networks with at least one convolution layer, which combines two mathematical functions to produce a third. They are most commonly used for image classification [28][29]. Early in their development, LeCun *et al.* [3] used a single convolution layer to present the concept of CNNs. Since then, researchers have made continual improvements of the original method, such as with the creation of AlexNet [19].

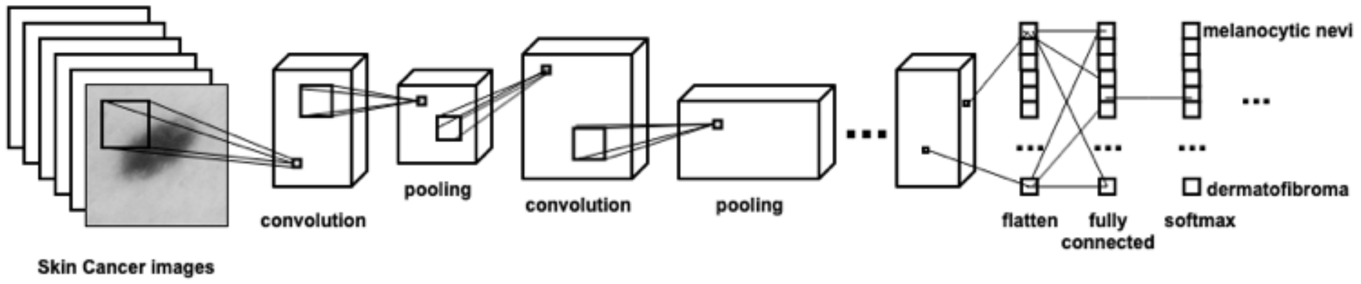

Fig. 2. Architecture of Convolutional Neural Networks for Skin Cancer Detection

The main advantage of a CNN is its ability to extract pertinent features automatically. In our dataset, skin images are divided into seven categories and the CNN learns the features of each category without supervision. Figure 2 shows the proposed CNN architecture, where input is processed by a series of convolution, pooling and sub-sampling layers followed by fully-connected layers. The convolution and pooling layers perform feature extraction by capturing general characteristics of the images. The fully connected layers assign a probability for the input image according to the given features.

III. Experiments and Implementation

A. *Dataset*

We use a dataset of skin cancer images from [2] to demonstrate a proof of concept. The dataset consists of 10,015 dermatoscopic images from different populations separated into seven categories (types of skin lesions): melanocytic nevi, melanoma, benign keratosis-like lesions, basal cell carcinoma, actinic keratoses, vascular lesions, and dermatofibroma (Table 1).

TABLE I. Dataset of Skin Cancer Images

Skin lesion type	Abbreviation	Total
melanocytic nevi	nv	6705
melanoma	mel	1113
benign keratosis	bkl	1099
basal cell carcinoma	bcc	514
actinic keratoses	akiec	327
vascular lesions	vasc	142
dermatofibroma	df	115

Each image has a record in the metadata table, which includes fields such as lesion identifier, image identifier, diagnosis, diagnosis type, age, sex, and localization. Table 2 shows examples from the metadata table.

Upon inspection of the dataset we identified 57 records with null or missing values and subsequently removed them from consideration. The remaining images were then resized to 120*90 for practical reasons as the original size, 600*450, was too large to train in TensorFlow [17].

B. *Model Training*

The core algorithms were implemented on macOS using Python [16], Scikit-learn [20], TensorFlow [17], and Keras [18]. Scikit-learn is a library of common machine learning algorithms in Python. TensorFlow is an open source library for creating deep learning models. Keras runs on top of TensorFlow and enables efficient experimentation.We used the architecture shown in Figure 2, which uses convolutional land pooling layers. Pooling is a form of non-linear down-sampling and it is common to add pooling layers after convolutional layers to remove unnecessary features and reduce the number of parameters during the training in a quest to prevent overfitting. Specifically, we use max pooling (MaxPool2D). After the convolutional and max pooling layers, we use the flattening and fully connected (FC) layers, which flatten the multidimensional array into a two-dimensional one. The output layer is based on the softmax function, which calculates the probabilities of each class (i.e. type of skin lesion) in order to propose the most likely classification. Mathematically, the softmax function is shown below:

$$\sigma(Z)_j = \frac{e^{Z_j}}{\sum_{k=1}^{K} e^{Z_k}} \quad \text{for} \quad j = 1, \ldots, \mathrm{K}$$

where z is a vector of the inputs to the output layer and j indexes the output units.

One of the difficulties of automatically processing skin images is that an image may be taken under a variety of conditions (*e.g*., brightness, angle, focal distance), which makes the comparison of images and identification of pertinent features difficult. To minimize the impact of image parameters on classification model, we incorporated data augmentation, which in our case produces new images by randomly rotating, zooming, shifting, and cropping from the center of existing images. These additional images can then be used to boost training.

We split the dataset into three subsets to be used for training (64%), validation (16%) and testing (20%) respectively. The validation data were used for fine-tuning the hyperparameters during training phase. Table 3 shows that the best results we achieved.

TABLE II. Samples of Metadata

Lesion Identifier	Image Identifier	Diagnosis	Diagnosis Type	Age	Sex	Localization
HAM_0001480	ISIC_0026835	bkl	histo	70	M	abdomen
HAM_0005388	ISIC_0027815	bkl	histo	80	M	chest
HAM_0002129	ISIC_0025903	df	consensus	60	M	abdomen
…	…	…	…	…	…	…
HAM_0004607	ISIC_0026150	mel	histo	50	F	back
HAM_0006092	ISIC_0029241	mel	histo	70	M	face
HAM_0006047	ISIC_0026321	bcc	histo	65	F	scalp

TABLE III. MODEL EVALUATION

Data Augmentation				Performance	
Rotation Degree Range	**Zoom Range**	**Width Shift Range**	**Height Shift Range**	**Loss**	**Accuracy**
10	0.1	0.15	0.15	0.71	75.2%

The pre-trained model is the result of applying the CNN to the given dataset. The format of the model we trained is HDF5, a file format [34] suitable for storing large collections of multidimensional numeric arrays (e.g. models, data files).

C. Model Conversion and Integrating Pre-trained Model into the App

Core ML, a machine learning framework created by Apple (Figure 3), lets us integrate trained machine learning models into iOS app [36]. The integration process brings the machine learning model from the cloud onto the mobile devices.

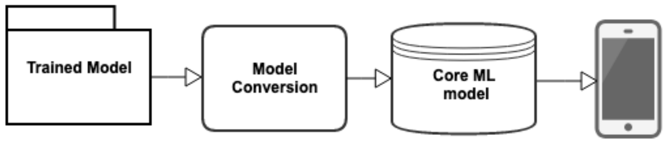

Fig. 3. Model Conversion and Integrating Pre-trained Model into the App

However, Core ML only supports a very limited number of model types [35]. In most of the cases, we need to convert the pre-trained model into one suitable for integrating into the mobile app. In this study, we used the Python library Core ML Tools [4] to convert our pre-trained model (.hdf5 format) into the Core ML model format (.mlmodel).

D. Demonstration

Figure 4 demonstrates the app's interface designed for the Apple iOS system. A user provides a skin image as input. This image is then warped into a probability distribution over the considered categories of skin lesions. The most probable classification (e.g., melanoma) is then displayed on the screen. The inference process is near real-time.

IV. CONCLUSIONS

mHealth is considered one of the most transformative drivers for ubiquitous health informatics delivery of medical applications. The combination of machine learning and mobile health technology has a great potential to transform the detection and prevention of various diseases, such as skin cancer. Many studies about machine learning on mobile devices have been the focus on cloud-based solutions because memory and computational power is relatively limited. However, cloud-based approaches come with drawbacks such as latency and privacy, which need to be considered in the context of medical applications. In this study, we present an on-device inference approach, which has and number of benefits over cloud-based solutions. These include lower latency, improved privacy, lower costs and higher availability. As the pre-trained model is deployed on the mobile device, it is not necessary to invoke a cloud service to classify the category of skin lesions. This on-device inference approach substantially reduces latency. Moreover, it improves privacy because it does not require a patient to send images to a third-party cloud service. Lastly, it eliminates the overhead and cost of running and maintaining cloud services.

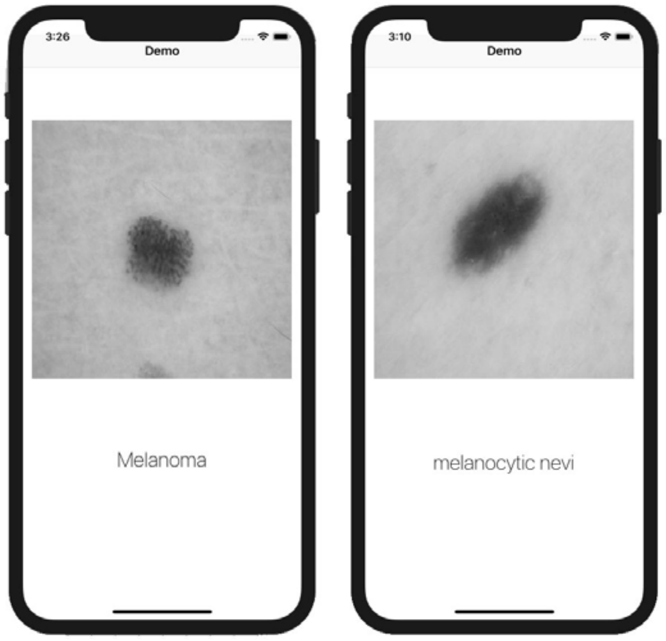

Fig. 4. Demonstration of Skin Cancer Detection

The main limitation of our approach is that the pre-trained model is not easily updated because the offline model is integrated in the app. Thus, if we would like to improve the classification performance of the app by re-training its model using additional skin images, it would need to be update the up by downloading and installing the new version, which may be time-consuming and inconvenient for the user.

REFERENCES

[1] Khan, Sameer, and Suet-Peng Yong. "A deep learning architecture for classifying medical images of anatomy object." Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC), 2017. IEEE, 2017

[2] Tschandl, Philipp, Cliff Rosendahl, and Harald Kittler. "The HAM10000 Dataset: A Large Collection of Multi-Source Dermatoscopic Images of Common Pigmented Skin Lesions." arXiv preprint arXiv:1803.10417 (2018).

[3] LeCun, Yann, et al. "Gradient-based learning applied to document recognition." Proceedings of the IEEE 86.11 (1998): 2278-2324.

[4] Core ML Community Tools. https://github.com/apple/coremltools (Accessed: 11 Feb.2019)

[5] Rogers, Howard W., et al. "Incidence estimate of nonmelanoma skin cancer (keratinocyte carcinomas) in the US population, 2012." JAMA dermatology 151.10 (2015): 1081-1086.

[6] Stern, Robert S. "Prevalence of a history of skin cancer in 2007: results of an incidence-based model." Archives of dermatology 146.3 (2010): 279-282.

[7] Armstrong, April W., et al. "Text-message reminders to improve sunscreen use: a randomized, controlled trial using electronic monitoring." Archives of dermatology 145.11 (2009): 1230-1236.

[8] Youl, Philippa H., et al. "Can skin cancer prevention and early detection be improved via mobile phone text messaging? A randomised, attention control trial." Preventive medicine 71 (2015): 50-56.

[9] Banos, Oresti, et al. "mHealthDroid: a novel framework for agile development of mobile health applications." International Workshop on Ambient Assisted Living. Springer, Cham, 2014.

[10] Ruiz-Zafra, Ángel, et al. "Zappa: An open mobile platform to build cloud-based m-health systems." Ambient intelligence-software and applications. Springer, Heidelberg, 2013. 87-94.

[11] Gatsios, Dimitrios, et al. "Mhealth platform for Parkinson's disease management." International Conference on Biomedical and Health Informatics. Springer, Singapore, 2015.

[12] Melillo, Paolo, et al. "Cloud-based remote processing and data-mining platform for automatic risk assessment in hypertensive patients." International Workshop on Ambient Assisted Living. Springer, Cham, 2014.

[13] Pan, Di, et al. "A mobile cloud-based Parkinson's disease assessment system for home-based monitoring." JMIR mHealth and uHealth 3.1 (2015).

[14] Callejas, Zoraida, et al. "A virtual coach for active ageing based on sentient computing and m-health." International Workshop on Ambient Assisted Living. Springer, Cham, 2014.

[15] Caffe. http://caffe.berkeleyvision.org/ (Accessed: 7 Feb.2019)

[16] Python. https://www.python.org/ (Accessed: 10 Feb.2019)

[17] Tensorflow - An open source machine learning framework for everyone. https://www.tensorflow.org/ (Accessed: 8 Feb.2019)

[18] Keras: The Python Deep Learning library https://keras.io/ (Accessed: 9 Feb.2019)

[19] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems. 2012.

[20] scikit-learn: Machine Learning in Python https://scikit-learn.org/stable/ (Accessed: 12 Feb.2019)

[21] Convert MXNet models into Apple CoreML format https://github.com/apache/incubator-mxnet/tree/master/tools/coreml (Accessed: 12 Feb.2019)

[22] TensorFlow (TF) to CoreML Converter https://github.com/tf-coreml/tf-coreml (Accessed: 5 Feb.2019)

[23] Shin, Hoo-Chang, et al. "Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning." IEEE transactions on medical imaging 35.5 (2016): 1285-1298.

[24] Kamnitsas, Konstantinos, et al. "Efficient multi-scale 3D CNN with fully connected CRF for accurate brain lesion segmentation." Medical image analysis 36 (2017): 61-78.

[25] Tajbakhsh, Nima, et al. "Convolutional neural networks for medical image analysis: Full training or fine tuning?." IEEE transactions on medical imaging 35.5 (2016): 1299-1312.

[26] Nie, Dong, et al. "Estimating CT image from MRI data using 3D fully convolutional networks." Deep Learning and Data Labeling for Medical Applications. Springer, Cham, 2016. 170-178.

[27] Ledig, Christian, et al. "Photo-realistic single image super-resolution using a generative adversarial network." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

[28] Moeskops, Pim, et al. "Automatic segmentation of MR brain images with a convolutional neural network." IEEE transactions on medical imaging 35.5 (2016): 1252-1261.

[29] Bar, Yaniv, et al. "Deep learning with non-medical training used for chest pathology identification." Medical Imaging 2015: Computer-Aided Diagnosis. Vol. 9414. International Society for Optics and Photonics, 2015.

[30] Marcolino, Milena Soriano, et al. "The impact of mHealth interventions: systematic review of systematic reviews." JMIR mHealth and uHealth 6.1 (2018).

[31] Dai, Xiangfeng, and Marwan Bikdash. "Trend analysis of fragmented time series for mHealth apps: Hypothesis testing based adaptive spline filtering method with importance weighting." IEEE Access 5 (2017): 27767-27776.

[32] Longacre, Meghan, et al. "Clinical Adoption of mHealth Technology to Support Pediatric Cystic Fibrosis Care in Sweden: Qualitative Case Study." JMIR Pediatrics and Parenting 1.2 (2018): e11080.

[33] Ahsan, GM Tanimul, et al. "Toward an mHealth intervention for smoking cessation." 2013 IEEE 37th Annual Computer Software and Applications Conference Workshops. IEEE, 2013.

[34] How can I save a Keras model? https://keras.io/getting-started/faq/#how-can-i-save-a-keras-model (Accessed: 10 May.2019)

[35] Converting Trained Models to Core ML, https://developer.apple.com/documentation/coreml/converting_trained_models_to_core_ml (Accessed: 11 May.2019)

[36] Core ML- Integrate machine learning models into your app. https://developer.apple.com/documentation/coreml (Accessed: 11 May.2019)