TIPTEKNO'24 MEDICAL TECHNOLOGIES CONGRESS
10-12 October 2024
Yasmin Bodrum Resort, Muğla, Türkiye

Deep Learning Based Application Design for Detection of Skin Lesion Types

Veronika Selin Erdem
Dept. of Biomedical Engineering
Izmir Katip Celebi University
Izmir, TURKEY
veronikaselin12@gmail.com

Ozlem Karabiber Cura
Dept. of Biomedical Engineering
Izmir Katip Celebi University
Izmir, TURKEY
ozlem.karabiber@ikcu.edu.tr

Fatma Gunseli Yasar Ciklacandir
Dept. of Computer Engineering
Izmir Katip Celebi University
Izmir, TURKEY
fatmagunseli.yasar@ikcu.edu.tr

Samet Ciklacandir
Dept. of Biomedical Engineering
Izmir Katip Celebi University
Izmir, TURKEY
samet.cikla@gmail.com

***Abstract*—This project aims to develop a deep-learning model capable of classifying skin lesions. The HAM10000 dataset, consisting of 10015 dermatoscopic images representing seven common classes of skin lesions, was used to train different models. Among these, MobileNet and InceptionV3 gave the best results with 91.14 percent and 89.66 percent validation accuracy, respectively. Building upon these results, we designed a custom model to further investigate its potential for skin lesion classification. The custom model achieved 82.25 percent validation accuracy. To enhance accessibility and usability, the trained model was integrated into a mobile application. This application allows users to upload photos of skin lesions and receive real-time predictions along with class information, thereby providing a convenient tool for preliminary skin health assessment. This research highlights the potential of deep learning approaches in dermatology and provides a comparative analysis of different model architectures for skin lesion classification.**

Keywords—Deep learning, skin lesions, skin cancer, mobile application.

I. Introduction

One of the most prevalent cancers in the world is skin cancer, and early detection is essential for both successful treatment and better patient outcomes [1]. Dermatoscopic images, which provide detailed views of skin lesions, are invaluable in the diagnostic process. However, accurately interpreting these images requires significant expertise and experience. Artificial intelligence's deep learning subgroup has demonstrated promise in automating the processing of medical images, potentially helping dermatologists and enhancing diagnostic precision [2]. In this study, we explore the application of deep learning techniques to classify skin lesions using the HAM10000 dataset, a comprehensive collection of dermatoscopic images representing seven different classes of skin conditions, including both benign and malignant lesions. Each image's diagnosis was confirmed by clinical experts, providing reliable annotations for model training. However, the HAM10000 dataset presents a significant class imbalance challenge, with some lesion types being underrepresented [3]. Our goal is to create a strong model that can automatically and effectively categorize skin lesions using Convolutional Neural Networks (CNNs), assisting in early diagnosis and treatment. In this study, we first explored the performance of pre-trained models such as MobileNet, InceptionV3, VGG19, and ResNet50 which are known for their efficiency and accuracy in image classification tasks. Building on these findings, a custom model capable of differentiating skin lesion types is developed. In the following sections, we detail our methodology, including the training process, the architecture of our custom model, and the techniques employed to enhance model performance and generalization. Additionally, the trained model is integrated into a mobile application, which allows users to upload photos of their skin lesions and receive immediate predictions along with class information. Screenshots of the mobile app, demonstrating its interface and functionality, are included in the results section.

II. Materials and Methods

A. Choice of Programming Language and Libraries

At the start of the project, MATLAB was used for model development and training but faced performance issues due to its CPU-only processing, leading to slow training times. To address this, we transitioned to Python, using TensorFlow and Keras libraries within the Kaggle environment. Kaggle's free access to high-performance GPUs enabled efficient model training without extra costs. TensorFlow and Keras were chosen for their powerful features and ease of use.

B. Addressing Class Imbalance

Class imbalance, in which certain classes are underrepresented compared to others, is a prevalent problem in machine learning classification problems. Biased models that perform well in majority classes but badly in minority classes may result from this imbalance. We used class weighting, a method that gives minority classes greater weights during training, to remedy this. By using this method, the model's overall performance and fairness are improved since it lays greater emphasis on accurately categorizing underrepresented classes [4]. The class distribution in the HAM10000 dataset is summarized in Table 1, along with the class weights allocated to correct imbalances in the distribution of classes. To ensure that

979-8-3315-2981-9/24/$31.00 ©2024 IEEE

minority classes receive greater weights to balance their influence during model training, the allocated weights were determined by dividing the total number of samples in the dataset by the number of samples in each class.

TABLE I. CLASS WEIGHTS

Class	Number of Samples	Assigned Weight
Actinic Keratosis	327	30
Basal Cell Carcinoma	514	20
Benign Keratosis	1099	9
Dermatofibromas	115	87
Melanoma	1113	9
Melanocytic Nevi	6705	2
Vascular Skin Lesion	142	70

To address the class imbalance, we have also used upsampling and downsampling approaches (Table II). Upsampling is the process of creating new samples by data augmentation, hence increasing the number of samples in the minority classes. The dataset has been expanded to 33220 training images by applying rotation, translation, flipping, and brightness modifications. Upsampling ensures that minority classes are well-represented during the training process, enabling the model to learn more effectively from these classes. On the other hand, downsampling entails lowering the number of samples in the majority classes to the same amount as in the minority classes. By lessening the dominance of majority classes, this strategy contributes to the distribution of classes being balanced [5].

TABLE II. UPSAMPLING AND DOWNSAMPLING

Class	Number of samples	
	Upsampling	Downsampling
Actinic Keratosis	4364	282
Basal Cell Carcinoma	4890	461
Benign Keratosis	4946	967
Dermatofibromas	3697	103
Melanoma	4946	1044
Melanocytic Nevi	5954	1380
Vascular Skin Lesion	4423	123

C. Hyperparameters Tuning

In the development and training of our deep learning models, several hyperparameters were carefully tuned to optimize performance [6]. The following hyperparameters were adjusted and tested to determine the best configuration for achieving high accuracy:

- **Mini Batch Size**: We experimented with different mini-batch sizes to find an optimal value that balances training efficiency and model performance. The mini-batch size determines the number of samples processed before the model's internal parameters are updated.
- **Max Epochs**: The maximum number of epochs was varied to ensure that the model had sufficient time to learn from the training data without overfitting. An epoch refers to one complete pass through the entire training dataset.
- **Categorical Focal Cross Entropy**: To address the class imbalance, the categorical focal cross-entropy loss function was employed. This loss function gives more weight to hard-to-classify examples, helping the model to focus on underrepresented classes.
- **L2 Regularization**: To prevent overfitting, L2 regularization was applied to the model's weights. This technique penalizes large weights, encouraging the model to maintain simpler, more generalizable solutions.
- **Adam Solver**: The Adam optimization algorithm was employed, which integrates the benefits of the AdaGrad and RMSProp algorithms. Adam is renowned for its ability to compute efficiently and fit situations with big datasets and parameters.
- **Learning Rate**: Step size during gradient descent is controlled by a critical hyperparameter called learning rate. Finding a learning rate that permits the model to converge well without fluctuating or diverging required testing across a range of values.

By methodically changing each parameter and assessing the model's performance on the validation set, hyperparameter tuning was carried out. We found the ideal configurations that increased validation accuracy by iteratively experimenting with various pairings.

D. Model Evaluation

Several important measures that offer an extensive overview of the model's efficacy in accurately classifying images were employed to assess the performance of our deep-learning models. Accuracy (1), precision (2), recall (3), and F1-score (4) performance metrics are employed [7].

$$Accuracy = \frac{TP+TN}{TP+TN+FN+FP} \quad (1)$$

$$Precision = \frac{TP}{TP+FP} \quad (2)$$

$$Recall = \frac{TP}{TP+FN} \quad (3)$$

$$F1-score = \frac{2\ x\ Precision\ x\ Recall}{Precision+Recall} \quad (4)$$

The definition of terms used in the formulas are given below:

- **TP** (True Positives): The number of positive cases that were accurately predicted.
- **TN** (True Negatives): The number of accurately predicted negative cases.

- **FP** (False Positives): The number of positive cases that were not correctly predicted.
- **FN** (False Negatives): The number of negative cases that were mistakenly predicted.

E. Original Model Development

After implementing and evaluating pre-trained models, we aimed to develop an original model tailored specifically to our classification task. The architecture of our original model is presented in Fig. 1.

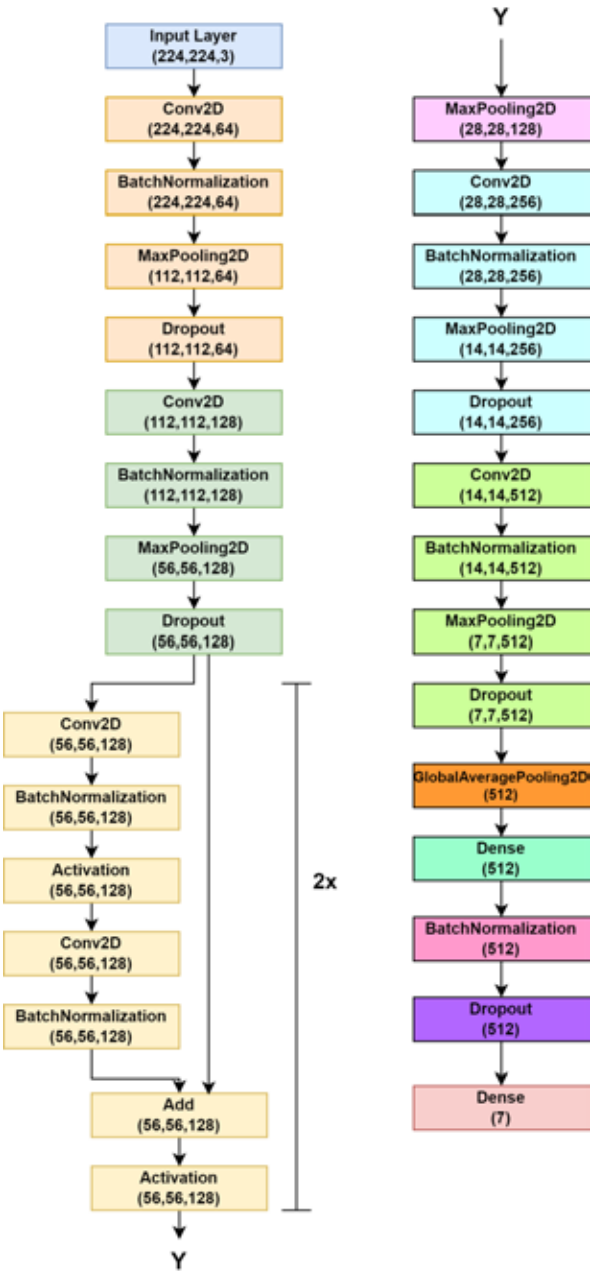


Fig. 1. Architecture of original model tailored specifically to the classification of HAM10000 dataset.

F. Application Development

In addition to developing a deep learning model, we aimed to create a user-friendly mobile application to make our model's capabilities accessible to end-users. For this purpose, we used the Kivy library within the PyCharm Integrated Development Environment (IDE) to develop our application. The application was designed with a simple and intuitive user interface to ensure ease of use.

The workflow of our application is shown below;

1. **User Registration and Login**: The application starts with a welcome screen where users can either register for a new account or log in to an existing one. This step ensures that user data and preferences can be securely stored and accessed.
2. **Image Selection**: Once logged in, users are directed to a screen to select an image of a skin lesion from their device's gallery. This image is then displayed on a new screen within the app.
3. **Prediction**: Users click the "Predict" button after selecting an image. The application integrates the pre-trained deep learning model to classify the selected image. The model processes the image and predicts the class of the skin lesion.
4. **Result Display**: The predicted class of the skin lesion is displayed on a new screen. This result informs the user about the potential type of skin lesion based on the model's classification.
5. **Detailed Information**: To provide more value to the users, the application includes a "Get Information" button. By clicking this button, users are directed to another screen that provides detailed information about the predicted lesion type. This information includes descriptions, risk factors, and other relevant details.

Integrating the trained model into the Kivy application involved several key steps:

1. **Model Loading**: The trained model, developed using TensorFlow and Keras, was exported and then loaded into the application.
2. **Image Preprocessing**: The selected image undergoes preprocessing to match the input requirements of the model, including resizing and normalization.
3. **Prediction Execution**: The preprocessed image is fed into the model, which then outputs the prediction. This prediction is mapped to the corresponding class label and displayed to the user.

Screenshots of the application's user interface are provided in the Results section (Fig. 2) to illustrate the user flow and the application's functionalities.

III. RESULTS

A thorough exploration of hyperparameters was conducted to optimize the performance of our deep-learning models. Table III summarizes the selected hyperparameters along with their corresponding performance values.

TABLE III. HYPERPARAMETERS AND CORRESPONDING PERFORMANCE

Model	Accuracy	Loss	Loss Function	Learning Rate	Mini Batch Size	Max Epochs	L2 Reg.	Class Weights	Time
MobileNet	0.9114	0.1098	CFE	0.0001	50	50	0.001	30 20 9 87 9 2 70	32 min
InceptionV3	0.8966	0.0666	CFE	0.0001	50	50	0.001	30 20 9 87 9 2 70	46 min
MobileNet	0.8946	0.1081	CFE	0.0001	50	70	0.001	30 20 9 87 9 2 70	42 min
MobileNet	0.8942	0.4245	CE	0.0001	50	50	none	4 2 1 12 1 1 10	20 min
MobileNet	0.8817	0.4666	CE	0.001	50	60	0.001	30 20 9 87 9 2 70	30 min
MobileNet	0.8807	0.1045	CFE	0.0001	50	50	0.001	30 20 9 87 9 2 70	34 min
InceptionV3	0.8762	0.6118	CE	0.0001	100	50	0.001	4 2 1 12 1 1 10	60 min
InceptionV3	0.8613	0.5839	CE	0.0001	150	50	0.001	30 20 9 87 9 2 70	45 min
MobileNet	0.8487	0.5994	CE	0.0001	150	50	none	30 20 9 87 9 2 70	32 min
VGG19	0.8319	0.1452	CFE	0.0001	50	50	0.001	30 20 9 87 9 2 70	74 min
MobileNet	0.8261	0.7318	CE	0.0001	150	50	0.01	30 20 9 87 9 2 70	32 min
Original Model	0.8225	0.2111	CFE	0.0001	50	50	0.001	30 20 9 87 9 2 70	40 min
ResNet50	0.6614	0.1633	CFE	0.0001	40	50	0.001	30 20 9 87 9 2 70	36 min

*CFE: Cross focal entropy, CE: Cross entropy
** Adam optimizer is utilized for each model as a solver.

The best results are achieved by MobileNet and InceptionV3 models, scoring 91.14% and 89.66% validation accuracy, respectively. Our original model scored 82.25% validation accuracy, showcasing its competitive performance compared to the established architectures. Screenshots of the developed mobile application are presented below (Fig. 2), showcasing the user interface and the functionality of the application.

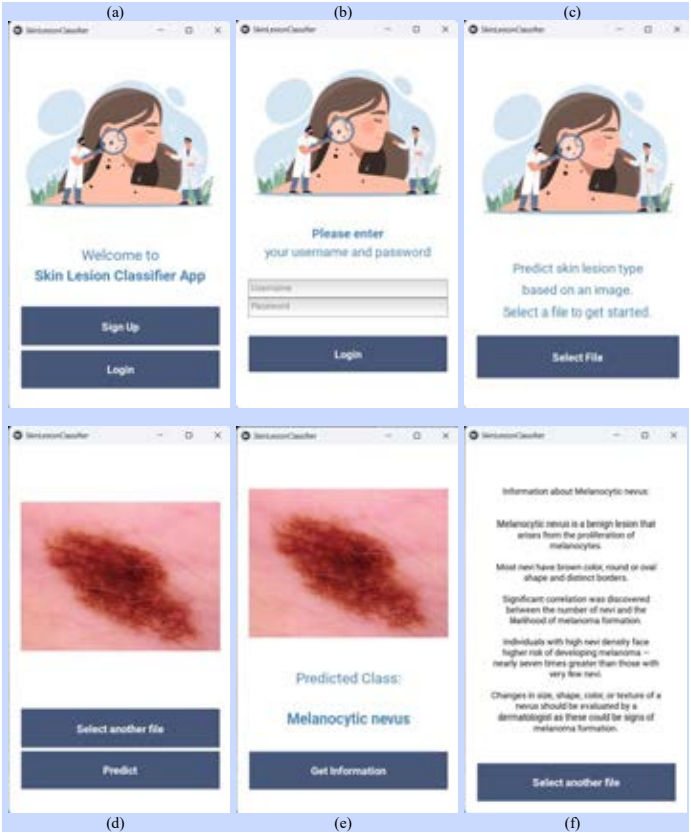


Fig. 2. This figure illustrates the key interfaces of the mobile application with integrated deep learning model developed for skin lesion classification.

IV. DISCUSSION

Early detection is critically important in dermatology, particularly when it comes to skin cancer. Early diagnosis can significantly improve the prognosis and reduce the need for more invasive treatments. The worst type of skin cancer, melanoma, for example, has a high survival percentage when caught early but can swiftly spread to other parts of the body and become life-threatening. Our application supports early detection by enabling users to assess skin lesions quickly and easily, facilitating timely medical consultations and interventions. Also, our application provides detailed information about the predicted lesion type, enhancing user awareness and educational value. Looking forward, there are several promising directions for advancing deep learning-based dermatological applications, which can significantly contribute to early diagnosis, better patient outcomes, and more personalized treatment plans.

REFERENCES

[1] G.Atış, İ. K. Altunay, G. Tukenmez Demırcı, E. Aydın, D. Mammadov, "The most common skin cancers and the risk factors in geriatric patients: A hospital based-controlled study," Journal of Experimental and Clinical Medicine, 32(4), 165-170, 2016.

[2] Z. Guo, X. Li, H. Huang, N. Guo, and Q. Li, "Deep Learning-based Image Segmentation on Multimodal Medical Imaging," IEEE transactions on radiation and plasma medical sciences, 3(2), 162–169, 2019.

[3] P. Tschandl, C. Rosendahl, and H. Kittler, "The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions," Scientific data, 5, 180161, 2018.

[4] C. Bao, "Image Classification Under Class-Imbalanced Situation," Highlights in Science, Engineering and Technology, 39:465-474, 2023. doi: 10.54097/hset.v39i.6570

[5] V. Werner de Vargas, J.A. Schneider Aranda, R. Dos Santos Costa, R., P.R. da Silva Pereira, and J.L. Victória Barbosa, "Imbalanced data preprocessing techniques for machine learning: a systematic mapping study," Knowledge and information systems, 65(1), 31–57, 2023.

[6] T. Yu, and H. Zhu. "Hyper-parameter optimization: A review of algorithms and applications," *arXiv preprint arXiv:2003.05689*, 2020.

[7] M. Hossin, and M.N. Sulaiman. "A Review on Evaluation Metrics for Data Classification Evaluations," International Journal of Data Mining & Knowledge Management Process. 5. 01-11, 2015. 10.5121/ijdkp.2015.5201.