Android Teledermatology Application for Skin Lesions Classification Based on Dermoscopic Images

Faris Abdurrahman Gymnastiar[1], Yoyok Heru Prasetyo Isnomo[2*], Ahmad Wilda Yulianto[3]

[1,2,3] Digital Telecommunication Network Study Program, Department of Electrical Engineering, State Polytechnic of Malang, 65141, Indonesia.

[1] faris.a.gymnastiar@gmail.com, [2] yoyok.heru@polinema.ac.id, [3] ahmadwildan@polinema.ac.id

***Abstract*— Skin cancer is a commonly diagnosed cancer worldwide, which is ranked third in Indonesia. While this type of cancer is curable in the early stage, there is an increase in the risk of death in the late stage. Thus, early detection is essential to minimize the spread of cancer. A prototype of an Android-based application was developed in this research to classify skin lesions. Dermoscopy images from the MNIST HAM10000 dataset comprised 10,015 images across seven skin cancer classes. Five pre-trained models from Keras, ResNet50V2, DenseNet121, InceptionV3, Xception, and MobileNet will be utilized for training and testing. During training, each model will be tested for classification performance based on changes in several hyperparameter variables, including learning rate, optimizer, batch size, and dropout.After testing, each model's performance improvement was obtained with adjustments to the training hyperparameters. The Xception model achieved the best accuracy at 92.5 percent. This model was then implemented in an Android-based application. The testing results on Android showed the same accuracy for original dermoscopic images from the test dataset. However, a decrease in accuracy was observed when testing with the camera, which noise or differences in resolution might cause.**

Keywords*— *Convolutional Neural Network, Deep Learning, Hyperparameter Tuning, Pre-trained Model, Skin Lesions.

I. INTRODUCTION

Skin cancer is a commonly diagnosed cancer, with an estimated two to three million cases of non-melanoma cancer and 132,000 cases of melanoma are diagnosed annually based on the data from WHO [1]. Meanwhile, it is ranked third in Indonesia after cervical and breast cancer [2]. Early-stage lesions for skin cancer can be cured when its detected and treated early by minimizing the cancer spread. In Indonesia, healthcare services are provided through numerous community health centers (Puskesmas). However, the distribution of medical specialists, particularly dermatologists, is uneven, primarily concentrated in major cities, leading to disparities in access to quality skin healthcare.

Those healthcare gaps can be filled by using a teledermatology, which is a telemedicine that specializes in treating skin problems. That system allows patients to do a remote consultation through video calls or have their medical assessment by sending skin images to its database. With advancements in technology, there also have been several proofs of the use of Convolutional Neural Network (CNN) to do image classification on healthcare problems. This is possible due to their ability to learn and recognize complex features from images.

In research [4], a system for detecting and classifying skin diseases based on dermoscopic skin images has been designed. The system is designed for web application-based teledermatology. The input provided is a dermoscopy image. The output obtained includes results such as accuracy in classifying the seven types of skin lesions. That study used Inception and MobileNet V1 for skin lesion classification, achieving average accuracies of 58% and 78%. Whereas [5] compared the use of ResNet50, DenseNet121, MobileNet, Xception, InceptionV3, and InceptionResNetV2 for tuberculosis detection, finding DenseNet121 to be the most accurate at 91.57%.

In the study [6], EfficientNets were evaluated for multiclass skin cancer classification, while [7] used VGG-16 to test different models with various hyperparameters such as learning rate, optimizer, and batch size. The different results of each model architecture's performance demonstrate the importance of selecting and adjusting the appropriate value of each hyperparameter.

Based on these references, this research proposes using CNN to classify skin lesions in dermoscopy images. Five high-accuracy pre-trained models, such as ResNet50V2, DenseNet121, InceptionV3, Xception, and MobileNet, were trained using the public HAM10000 dataset on [9], which consists of 10,015 skin lesion images. Each model was tested to determine its impact on their performance when trained on several hyperparameter settings, including learning rate, optimizer, batch size, and dropout. The best-performing model was implemented in an Android-based mobile application to assist either doctors or patients in detecting and classifying skin lesions more quickly and accurately.

II. METHOD

The research method consists of several stages, which include training the pre-trained model and testing various hyperparameters to determine the optimal configuration. This process is followed by the implementation of the best-performing model on the Android platform for practical

*Corresponding author

application and evaluation. The overall workflow of the pre-trained CNN model design is illustrated, as shown at Fig. 1.

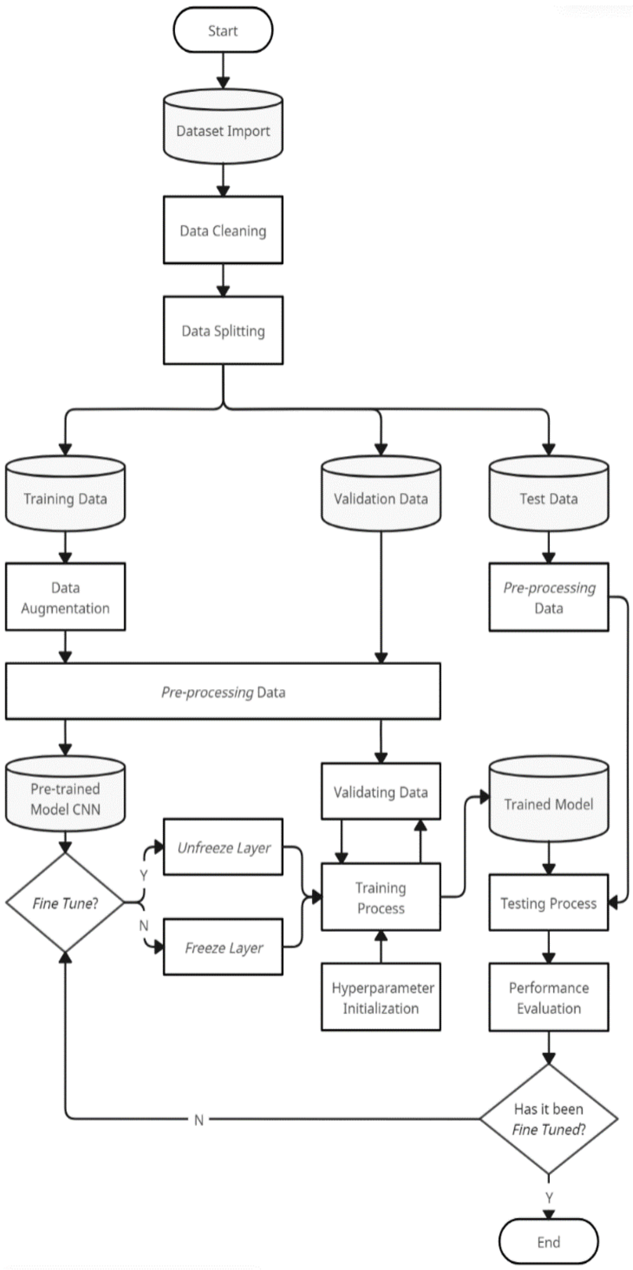

Figure 1. Pre-Trained CNN model design flow diagram

A. Research Data Cleaning

In this research, the HAM 10000 dataset consists of 10.015 images from seven classes of skin lesions. Data cleaning was carried out by separating duplicated data from original data. The dataset used has 4,501 duplicate images.

At the same time, the duplicated image is the result of enlarging the image to be more focused than the original image. Therefore, the duplicated images are separated from the original data to produce 5,514 clean images, which will be processed further.

B. Data Splitting

The clean dataset can then be split into three different parts. Each dataset part would be divided using specific ratios, such as 70:15:15, 80:15:5, or 80:10:10. The model uses training data to learn and recognize patterns within the images and can include unclean or raw data. Validation data is utilized to evaluate the model's performance during the training phase so that the model can adjust its training parameters. The test data only evaluates the overall model performance after the training. Hence, both should get clean data to provide an unbiased assessment.

For instance, in the splitting ratio of 80:10:10, 36% of the 5,514 clean data would be used to let validation and test data get 10% each. Then, it was split in half, each comprising 18% of the clean data or 993 images. Thus, the training data is derived from the remaining 64% of the clean and all the duplicated data.

C. Data Augmentation

An augmentation process was used to modify images from the dataset to create several new versions of images from it. In this research, the distribution of images across different lesion classes is imbalanced. The highest number was in the Nevus (nv) class, which consists of around 67% of the total data, whereas the fewest is in the Dermatofibroma (df) class, which consists only of 1% of the total data. Given this comparatively huge imbalance, data augmentation was employed to help increase the quantity and diversity of data on the training dataset without gathering more original data. Doing so can at least balance the data from each dataset class. In order to minimize the data imbalance gap in the minority class, several techniques were used to balance the data. Some of them include using rotation (up to 180 degrees), horizontal and vertical shifting (up to 10% of image width or height), image scaling (up to 10% enlargement), horizontal and vertical flipping, and the 'nearest' approach for filling image gaps during operations like rotation or shifting. All those techniques were combined to create new variations of the original images from the training dataset.

D. Data Pre-processing

Image preprocessing was done on all the dataset images, including the training set's augmented data. The images in the MNIST HAM10000 dataset are originally colored and have a size of 650x450 pixels. Thus, all those images were resized to 224x224 pixels across all RGB layers. Then, they were normalized into specific values based on their pre-trained model's format.

E. Pre-trained Model

Five of several pre-trained models from the Keras library were stated in this stage. Those models were MobileNet V2, ResNet50 V2, DenseNet121, Inception V3, and Xception. Each of those pre-trained models then received modifications in its network architecture using the same conditions.

In order to allow modification on pre-trained models, the "include_top" section was set to "false", which would remove the existing fully connected layers previously used for predicting 1000 classes of the ImageNet dataset. Hence, these models can receive additional layers, enabling further customization on specific datasets. Thus, this adjustment allows the models to classify seven different classes.After the convolution layers of each pre-trained model, the new layers and line of codes are added in the following order below.

- Global Average Pooling 2D is a pooling layer that reduces the spatial dimensions of the images to produce a single decimal value per channel.
- Batch Normalization was used to stabilize the training process by normalizing the values of input layer.
- Dropout randomly sets some neuron values to zero during training based on a certain percentage, leaving those neurons untrained.
- The dense layer runs as a fully connected layer. Since it is a multiclass classification, the softmax activation function is applied to the last line of those layers.

F. Transfer Learning and Fine Tuning

For each of those five pre-trained models, it was decided whether to receive single-step or two-step training. This was done through the method of freezing and unfreezing the base layers of the model. In the single-step training, each model received fine-tuning right away. The base layers of the model with the additional modified layers were unfrozen. This allows all those layers to update their weights, adjusting them to learn the datasets of the new task.However, the two-step training begins using transfer learning in the first step. During the transfer learning process, the base layers of the model were frozen. Thus, the new layers can learn features from the new images without disrupting the pre-trained weights. After that, the model was fine-tuned in its second step, letting the knowledge from the previous training be used.

G. Hyperparameter Initialization and Testing

When the first training began, each of those hyperparameters had the initial settings values such as optimizer = Adam, batch size = 64, and dropout = 0.05. After that, it would then be tested sequentially in the following order from left to right, as stated in the table below.

The types and values of hyperparameters tested during this process are summarized,as shown at Table I.

TABLE I
HYPERPARAMETER TYPES AND VALUES TESTED

Learning Rate	Optimizer	Batch Size	Dropout
0,001 then 0,001	Adam	16	0,3
0,001 then 0,0001	RMSprop	32	0,4
0,001	SGD	64	0,5
0,0001	Nadam	128	0,6

The test was done by averaging the three best-performing from five training to mitigate the influence of any outlier test results that might happen. The hyperparameter value that generates the average best performance was chosen as a fixed value for the next training.

For example, the first testing was done on the learning rate. If the model generates the best performance when trained using the learning rate value of 0.0001, then the baseline for the optimizer testing would be updated to learning rate = 0.0001, optimizer = Adam, batch size = 64, and dropout = 0.5. This process recurs and continues for each hyperparameter.

H. Model Training

This process was done to retrain the pre-trained model. Several training variables directly related to the model training process were declared.

- Epoch is a single complete pass of the model that has learned all the images in the training dataset. In this research, the total training epoch that was used ranged from 10 to 15.
- Early stopping is a regularization technique that would prevent overfitting by stopping the training process at a particular epoch. When there were no signs of a decrease in the validation loss for three consecutive epochs, the training process was stopped early.
- Adaptive learning rate adjusts the learning rate during the training process. Thus, it can learn faster in the early stages and slow down as it approaches the optimal solution. In this research, it was reduced when there was no improvement in the validation loss for two epochs.
- Checkpoint is a mechanism that periodically saves the model whenever it reaches the highest accuracy.

I. Model Testing and Evaluation

The testing and evaluating process was done using the images from the test dataset.This determines each of the model's ability to recognize new and unseen data, as the test dataset is not used in training. Each of those five models was evaluated using a confusion matrix. This shows their performance based on accuracy and the F1 score of each skin lesion class.

J. Model Implementation on Android

One of the five models that generate the best accuracy and F1-Score performance was implemented as a base model for the Android application. Then, it was converted into a format called '.tflite' and imported into the Android Studio IDE for further development.

III. RESULTS AND DISCUSSION

A. Data Processing

1) Data Splitting: This part investigates the performance impact of different dataset split ratios on one of the models, Xception. It was found that the 80:10:10 split ratio results in better performance, as shown at Table II.

TABLE II
PERFORMANCE RESULTS OF DATASET SPLITTING RATIO TEST

Ratio			Accuracy	F1-Score
Train	Val	Test		
70	15	15	91,673	0,753
80	5	15	92,237	0,773
80	10	10	92,950	0,803

Based on these results, the 80:10:10 split ratio was selected for this study. This means that the validation and test datasets each comprise 18% of the 5,514 clean images, while the remaining 3,528 clean images, along with all 4,501 duplicated images, are included in the training set. The distribution of dataset shares for each class is presented, as shown at TableIII.

TABLE III
NUMBER OF DATASET SHARES

Class Data	Training Data	Validation Data	Test Data
nv	5115	795	795
mel	1030	42	41
bkl	940	79	80
bcc	451	32	31
akiec	273	27	27
vasc	119	11	12
df	101	7	7

2) Data Augmentation: There are 8,029 dermoscopic images in the training set, while the validation and test set each had 993 images. There needs to be a more balanced amount of data on the training set. Therefore, the 'nv' class was used as a representative to ensure a balanced amount of data on each class. Thus, classes with fewer images received augmentation to increase the amount of their data until it reached or approached 5,000 images per class, as shown at Table IV.

TABLE IV
THE AMOUNT OF DATA AFTER AUGMENTATION

Class Data	Training Data	After Augmentation
Melanocytic Nevi (nv)	5115	5115
Melanoma (mel)	1030	4970
Benign Keratoses (bkl)	940	5000
Basal Cell Carcinoma (bcc)	451	4560
Actinic Keratoses (akiec)	273	4618
Vascular Lesions (vasc)	119	3027
Dermatofibroma (df)	101	3433
Total	8029	31123

The augmentation process proved to be highly effective for several skin lesion classes, including 'mel', 'bkl', 'bcc', and 'akiec', as these classes nearly achieved the target number of images set for balanced training. This indicates that the applied augmentation techniques were successful in enriching the training set with diverse variations of images, which helps improve the model's ability to generalize and accurately classify unseen images. On the other hand, the augmentation results for the 'vasc' and 'df' classes did not reach the desired number of images, largely due to their very small original training sets, which contained only 119 and 101 images, respectively. Despite the augmentation efforts, the limited initial data constrained the maximum achievable number of augmented images.

Examples of the augmented images for the 'df' class, demonstrating the diversity introduced by augmentation, are shown at Fig. 2. These examples illustrate how the augmentation process generated new variations while preserving the characteristics of the original lesions, which is essential for improving model robustness.

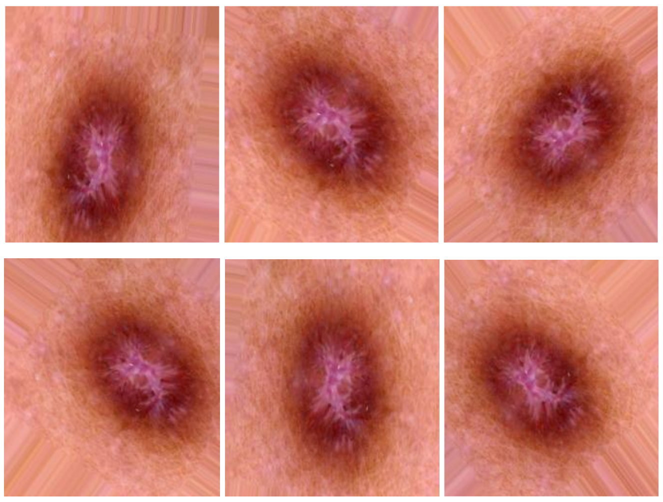
Figure 2. Examples of augmented images from 'df' class

B. Model Training

The performance of each model was assessed based on their accuracy score. The success of the model training is measured by the model's performance in predicting familiar data from the training set and new data from the test set. That was done to ensure that the model does not experience overfitting and underfitting. Overfitting occurs when the model memorizes the images from the training set instead of learning them. Hence, it predicts training data too well but generates poor accuracy on new data. Meanwhile, underfitting happens when the model cannot predict either type of data well.

This research uses an early stopping method to address an overfitting. This method prevents the model from memorizing the training data. The critical variable is patience, which determines the number of waiting epochs before stopping training. This study uses a patience value of three epochs, meaning the model stops training if the validation loss does not decrease over three consecutive epochs.

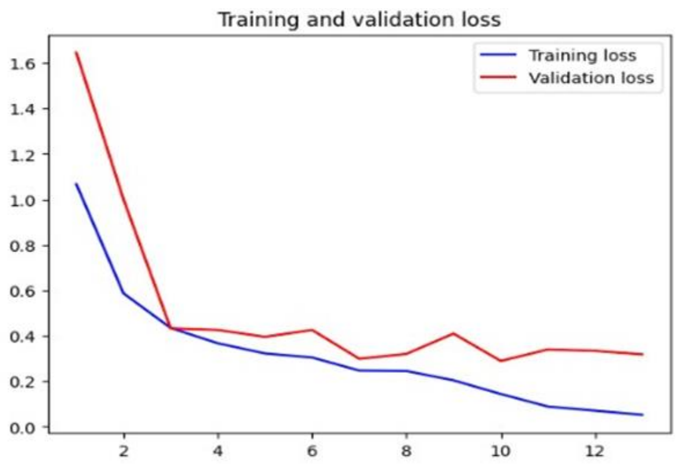

Figure 3. Training graph example

The graph in Fig. 3 illustrates the training of the Xception model. It shows that underfitting occurs when training is stopped before ten epochs, while signs of overfitting appear around epoch 13 and higher, indicated by an increase in validation loss. Hence, selecting the appropriate number of training epochs is crucial, which is effectively handled using early stopping.

C. Hyperparameter and Model Training

This part states each model's training and testing results against several hyperparameters. The test results show the average values from multiple tests. The averages were derived from the three best-performing tests out of five for each test. It was performed to mitigate the impact of any anomalous or outlier test results that might happen. Based on that reason, a more accurate, reliable, and consistent estimate of model performance was obtained by averaging multiple tests.

1) Learning Rate: At this stage, each model is trained using the initial value of each hyperparameter. Then, it tested four scenarios, two of which were done in a single stage of training. It uses the unfreeze layers method, which involves unlocking all weights in the model's layers. Thus, it allows changes and adjustments to the weight of each neuron value in all layers during the training process.

TABLE V
LEARNING RATE TESTING RESULTS WITH SINGLE STAGE TRAINING

Pre-trained	**0,001**		**0,0001**	
	Accuracy	**F1-Score**	**Accuracy**	**F1-Score**
MobileNet V2	78,093	0,580	89,033	0,653
ResNet50 V2	87,773	0,670	90,040	0,733
DenseNet121	90,533	0,717	91,477	0,747
Inception V3	88,053	0,653	91,087	0,737
Xception	92,080	0,773	91,107	0,740

Based on Table V, a learning rate of 0.0001 generally yields better performance, with the lowest top-three accuracy at 89%. The Xception model, however, performs best with a learning rate of 0.001, achieving 92% accuracy and an F1-Score of 0.773. Optimal hyperparameter values significantly affect training by updating neuron weights. To further assess performance, models underwent two-stage training: transfer learning with frozen base layers, followed by fine-tuning with all layers unfrozen.

TABLE VI
LEARNING RATE TESTING RESULTS WITH TWO STAGE TRAINING

Pre-trained	**0,001 / 0,001**		**0,001 / 0,0001**	
	Accuracy %	**F1-Score**	**Accuracy %**	**F1-Score**
MobileNet V2	85,963	0,620	87,363	0,603
ResNet50 V2	87,660	0,663	90,043	0,707
DenseNet121	90,823	0,723	92,327	0,793
Inception V3	90,437	0,683	90,923	0,727
Xception	92,950	0,803	90,973	0,733

By comparing the two-stage training results in Table VI with the single-stage results in Table V, the effect of two-stage training is noticeable across all models except for ResNet50 V2, which produces a relatively similar performance with the single-stage training. In contrast, MobileNet V2 and Inception V3 exhibit both improvements and declines in performance on those two tests. These mixed results suggest that the benefits of two-stage training may be model-specific and not universally applicable.

Though three of those models did not improve when trained in two stages, DenseNet121 and Xception achieved their peak performance when trained in this manner. Both models benefit only from a two-step approach, with increased accuracy and F1 scores likely due to their complexity. As such, in some situations, training a pre-trained model in two stages can improve its performance.

2) Optimizer: Each model is trained with batch size = 64 and dropout = 0.5. The learning rate values used in this training were derived from previous tests that had the best performance. Then, four testing scenarios on optimizers were conducted to observe the impact of different optimizers on each model's performance.

TABLE VII
OPTIMIZER TESTING RESULTS

Pre-trained	**F1-Score**			
	Adam	**RMSprop**	**SGD**	**Nadam**
MobileNet V2	0,653	0,613	0,353	0,643
ResNet50 V2	0,733	0,727	0,393	0,737
DenseNet121	0,793	0,760	0,415	0,763
Inception V3	0,737	0,747	0,415	0,757
Xception	0,803	0,743	0,530	0,730

Based on the results shown in Table VII, all models perform relatively better with the Adam and Nadam optimizers. The comparatively similar F1-Score evidences this on several models, with three models achieving their highest performance with Adam and two models with Nadam. The Xception model trained with Adam achieved the highest performance. This is followed by DenseNet121 and MobileNet V2. The other two models achieved their highest performance when trained using the Nadam optimizer, with Inception V3 and ResNet50 V2. These findings suggest that the choice of optimizer can significantly impact model performance, even among the models with similar architectures.

Meanwhile, testing with RMSprop yielded performance similar to Adam and Nadam but slightly lower. RMSprop serves as the basis for the development of Adam and Nadam optimizers, which sometimes makes RMSprop less effective in comparison. The performance difference between RMSprop and the other two optimizers is less than 1% of its accuracy. This close performance gap highlights the robustness of RMSprop. However, the improvements in Adam and Nadam make them better choices for many deep-learning applications.

The Stochastic Gradient Descent optimization algorithm tends to perform lower than RMSprop, Adam, and Nadam for several reasons. Unlike those three optimizers, SGD lacks the adaptive mechanisms and may need to be adjusted appropriately to the dynamic nature of its gradients. It is also not equipped with advanced features such as momentum, learning rate adaptation, or bias correction. In this way, SGD might not be sufficiently efficient when used to work with complex data and architectures.

3) Batch Size: Each model is trained with dropout = 0.5 at this stage, with the same learning rate used in the previous tests. The optimizer on each model was selected according to its highest performance. Then, four testing scenarios were conducted to observe the impact of different values of batch size on each model's performance.

TABLE VIII
BATCH SIZE TESTING RESULTS

Pre-trained	F1-Score			
	16	32	64	128
MobileNet V2	0,690	0,653	0,653	0,283
ResNet50 V2	0,703	0,753	0,737	0,710
DenseNet121	0,770	0,797	0,793	0,723
Inception V3	0,750	0,750	0,757	0,700
Xception	0,760	0,793	0,803	0,730

From the results in Table VIII, batch size indicates the total number of images trained simultaneously in one iteration, affecting the model's performance based on its architecture and complexity. Across all models, MobileNet V2 consistently produced the lowest performance and reached its highest when trained with a batch size 16. Meanwhile, ResNet50 V2 and DenseNet121 performed best on batch size 32 due to their higher model complexity.

Inception V3 and Xception worked better with a larger batch size of 64, with Xception achieving the highest performance and outperforming every model. However, the performance of Inception was still lower than that of DenseNet121 trained with batch sizes 32 or 64. Therefore, those tests demonstrate that a proper batch size value can affect the performance of a model and the stability of its training.

4) Dropout Rate: At this last hyperparameter test stage, the value of the learning rate and optimizer type was similar to the one used in the previous tests. At the same time, the batch size value was selected based on the one that provided the highest performance. Then, four testing scenarios on dropout rate were conducted to observe its impact on each model's performance.

TABLE IX
DROPOUT RATE TESTING RESULTS

Pre-trained	F1-Score			
	0,3	0,4	0,5	0,6
MobileNet V2	0,680	0,703	0,690	0,670
ResNet50 V2	0,707	0,730	0,753	0,720
DenseNet121	0,783	0,777	0,797	0,760
Inception V3	0,707	0,723	0,757	0,693
Xception	0,767	0,770	0,803	0,753

From the results in Table IX, during an iteration each neuron was selected randomly to change its value to zero. Thus, it does not contribute to the calculation processes, leaving its weight un-updated. All of the neurons were randomized again in the next iteration. Based on the test results, four of the five models tested achieved their highest performance when trained with a dropout rate of 0.5, with only MobileNet V2 performing best at a dropout rate of 0.4. The Xception model achieved the highest performance, while the lowest performance was seen in the MobileNet V2.

The performance of each model improved parallel with the increase in dropout rates from 0.3 to 0.5. After that, at a higher dropout rate, such as 0.6, the performance of each model begins to decline again. This occurs since a low dropout rate only deactivates a small portion of neurons, allowing the model to overlearn, potentially leading to overfitting. On the contrary, a high dropout rate ignores a high number of neurons, reducing the model's learning capability, which could lead to underfitting.

D. Model Analysis

Each model was thoroughly tested and trained using its optimal hyperparameter settings, which had been determined through systematic testing of learning rate, optimizer, batch size, and dropout rate. Based on the final evaluation shown in Fig. 4, it was found that the Xception model achieved the highest average performance, with an accuracy of 92.95% and an F1 score of 0.803. This indicates that Xception was able to generalize well across the training and test datasets, effectively recognizing and classifying skin lesion images. On the opposite end, the MobileNet V2 model produced the lowest average performance, with an accuracy of 90.10% and an F1 score of 0.703. This demonstrates that while MobileNet V2 is computationally lighter and faster, it may struggle with more complex patterns present in dermoscopic images compared to more sophisticated architectures such as Xception.

The classification performance of each model is strongly influenced by two main factors: the complexity of the lesion images and the number of images available in the original training dataset. For example, the class ‘mel’ (melanoma), which is the second largest class after ‘nv’ (melanocytic nevi) with 1,030 images, consistently showed lower F1 scores across all models. This indicates that despite having a relatively large number of images, the intricacy and high variability of melanoma lesions including irregular shapes, colors, and textures made it more difficult for the models to learn distinguishing features effectively.

In contrast, the ‘bkl’ (benign keratoses) class, which has a comparable number of original images at 940, consistently achieved higher F1 scores than the ‘mel’ class. This suggests that image complexity plays a more critical role than dataset size in determining model performance. Even with slightly fewer images, the less complex visual patterns in the ‘bkl’ class allowed the models to learn more efficiently and make more accurate predictions.

A similar pattern was observed between the ‘df’ (dermatofibroma) and ‘vasc’ (vascular lesions) classes. Both classes had the lowest original training dataset sizes, with only 101 images for ‘df’ and 119 for ‘vasc’. Through data augmentation, ‘df’ was increased to 3,433 images, while ‘vasc’ reached 3,027 images. Despite these being smaller numbers than other classes, both achieved relatively higher F1 scores than ‘mel’ or ‘bkl’. This highlights that simpler image features can compensate for smaller datasets, as the models can more easily extract and learn relevant features.

Overall, this analysis underscores that image complexity, not just dataset size, is a critical factor affecting model performance. Classes with high variability and intricate patterns require more sophisticated feature extraction, deeper architectures, or larger augmented datasets to achieve comparable performance to classes with simpler features. Furthermore, these findings suggest that augmenting underrepresented or complex classes and carefully selecting model architectures tailored to image complexity are essential strategies to improve overall classification accuracy and F1 scores across all classes, as shown in Fig. 4.

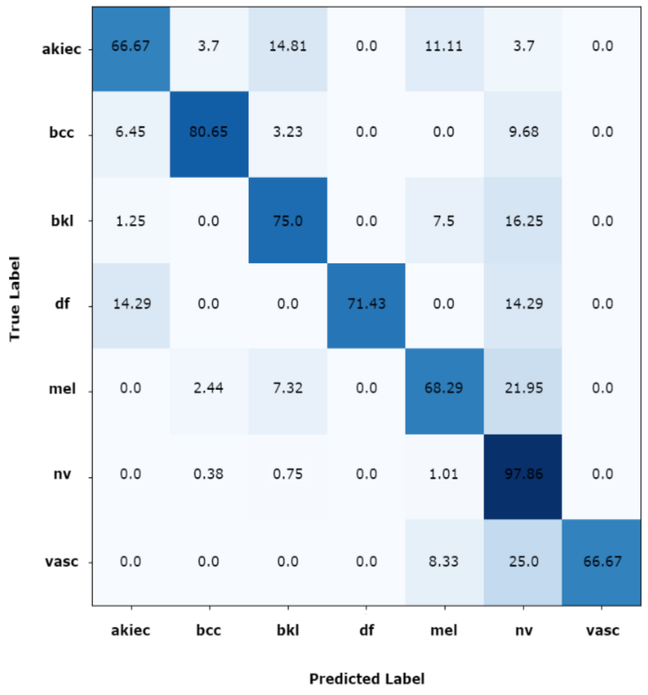

Figure 4. Confusion matrix of the highest performanced model xception

This analysis infers that the complexity of different image classes, along with the data imbalance problem, significantly impacts model performance, especially for minority classes. More learning process in these classes is required for the model to perform well. To improve the overall accuracy, more image data from another source could be added to help reduce the data imbalances in some classes. This approach would expand the original training and test set data.

E. Implementation on Android

The model Xception was chosen and implemented in Android application development since it generates the highest performance of the four models. Then, a couple of tests were conducted to determine the system's accuracy in classifying input images that were taken from the gallery and smartphone camera. These results were then compared with the classification results from Google Colab. In this test, a total of 100 images were randomly selected, with a minimum of seven images from each class, as shown in Table X.

TABLE X
ANDROID PREDICTION ACCURACY RESULTS

Lession	Test Data		Accuracy		
Class	Actual	Android	Actual	Gallery	Camera
akiec	27	8	0,67	0,75	0,63
bcc	31	9	0,81	0,89	0,89
bkl	80	11	0,75	0,64	0,73
df	7	7	0,71	0,71	0,43
mel	41	9	0,68	0,78	0,67
nv	795	48	0,98	0,94	0,71
vasc	12	8	0,67	0,63	0,75

It was found that all predicted images chosen from the phone gallery produced results close to the predictions on Google Colab. This means the model performed well in predicting original images, as there were no significant prediction errors between the original images tested on Google Colab and the Android application. This consistency can be observed in the 'df' class, which showed the same accuracy results on the same amount of test data.

However, predictions for camera-captured images exhibited decreased accuracy for the ‘akiec,’ ‘df,’ ‘mel,’ and ‘nv’ classes. The most significant drop occurred in the ‘nv’ class, where accuracy decreased by 23%, as presented in Table X. This decline is likely caused by variations in lighting, angles, and camera resolution, as well as artifacts from screen-captured images used during testing. Such artifacts may include screen reflections, moiré patterns, or focus inaccuracies, which introduce noise not present in the original dataset.

IV. CONCLUSION

In the development of this teledermatology application for early skin cancer detection, five pre-trained models were tested and trained using the HAM10000 dataset, which includes 10,015 dermoscopic images of skin lesions. Each of the models was also tested on various hyperparameters, such as learning rate, optimizer, batch size, and dropout rate, which was done to determine their impact on their performance. Among these models, the MobileNet V2 model demonstrated the lowest average performance in the top-three testing, with an accuracy of 90.1% and an F1 score of 0.703. Conversely, the Xception showed the highest performance, with an accuracy of 92.95%, an error rate of 7.05%, and an F1 score of 0.803, making it chosen for further Android application development.The testing indicated that the characteristics and complexity of the skin lesions class could affect the model performance in recognizing those classes. For example, the model struggled with the 'mel' class and consistently received the lowest F1 scores. The comparison between the test results from the model before and after its implementation in the Android application shows similar accuracy for images taken from the gallery. However, there is a noticeable drop in accuracy for camera-captured images, with the 'nv' class experiencing the most significant decrease of up to 23%. Those highlight that while the model performs well with original test dataset images, its performance could degrade on a real-world conditions.

REFERENCES

[1] Urban, Katelyn, Sino Mehrmal, Prabhdeep Uppal, Rachel L. Giesey, and Gregory R. Delost. 2021. “The Global Burden of Skin Cancer: A Longitudinal Analysis From the Global Burden of Disease Study, 1990–2017.” JAAD International 2 (March): 98–108.

[2] Wilvestra, Silvia, Sri Lestari, and Ennesta Asri. "Studi retrospektif kanker kulit di poliklinik ilmu kesehatan kulit dan kelamin RS Dr. M. Djamil Padang periode 2015-2017." Jurnal Kesehatan Andalas 7 (2018): 47-49.

[3] Mikołajczyk, Agnieszka, and Michał Grochowski. "Data augmentation for improving deep learning in image classification problem." In 2018 international interdisciplinary PhD workshop (IIPhDW), pp. 117-122. IEEE, 2018.

[4] Purnama, I. Ketut Eddy, Arta Kusuma Hernanda, Anak Agung Putri Ratna, Ingrid Nurtanio, Afif Nurul Hidayati, Mauridhi Hery Purnomo, Supeno Mardi Susiki, and Reza Fuad. "Disease classification based on dermoscopic skin images using convolutional neural network in teledermatology system." In 2019 international conference on computer engineering, network, and intelligent multimedia (CENIM), pp. 1-5. IEEE, 2019.

[5] Rochmawanti, Ovy, Fitri Utaminingrum, and Fitra A. Bachtiar. "Analisis Performa Pre-Trained Model Convolutional Neural Network dalam Mendeteksi Penyakit Tuberkulosis." Jurnal Teknologi Informasi dan Ilmu Komputer 8, no. 4 (2021): 805-814.

[6] Ali, Karar, Zaffar Ahmed Shaikh, Abdullah Ayub Khan, and Asif Ali Laghari. "Multiclass skin cancer classification using EfficientNets–a first step towards preventing skin cancer." Neuroscience Informatics 2, no. 4 (2022): 100034.

[7] Agustina, Regita, Rita Magdalena, And Nor Kumalasari Caecar Pratiwi. "Klasifikasi Kanker Kulit menggunakan Metode Convolutional Neural Network dengan Arsitektur VGG-16." ELKOMIKA: Jurnal Teknik Energi Elektrik, Teknik Telekomunikasi, & Teknik Elektronika 10, no. 2 (2022): 446.

[8] Prastika, Indah Widhi, and Eri Zuliarso. "Deteksi penyakit kulit wajah menggunakan tensorflow dengan metode convolutional neural network." Jurnal Manajemen Informatika dan Sistem Informasi 4, no. 2 (2021): 84-91.

[9] Tschandl, Philipp, Cliff Rosendahl, and Harald Kittler. "The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions." Scientific data 5, no. 1 (2018): 1-9.

[10] Baig, Abdul Rauf, Qaisar Abbas, Riyad Almakki, Mostafa EA Ibrahim, Lulwah AlSuwaidan, and Alaa ES Ahmed. "Light-dermo: A lightweight pretrained convolution neural network for the diagnosis of multiclass skin lesions." Diagnostics 13, no. 3 (2023): 385.

[11] Abiodun, Oludare Isaac, Aman Jantan, Abiodun Esther Omolara, Kemi Victoria Dada, Abubakar Malah Umar, Okafor Uchenwa Linus, Humaira Arshad, Abdullahi Aminu Kazaure, Usman Gana, and Muhammad Ubale Kiru. "Comprehensive review of artificial neural network applications to pattern recognition." IEEE access 7 (2019): 158820-158846.

[12] Purwono, Purwono, Alfian Ma'arif, Wahyu Rahmaniar, Haris Imam Karim Fathurrahman, Aufaclav Zatu Kusuma Frisky, and Qazi Mazhar ul Haq. "Understanding of convolutional neural network (cnn): A review." International Journal of Robotics and Control Systems 2, no. 4 (2022): 739-748.

[13] Putra, Wayan Suartika Eka. "Klasifikasi citra menggunakan convolutional neural network (CNN) pada caltech 101." Jurnal Teknik ITS 5, no. 1 (2016).

[14] Muraina, Ismail. "Ideal dataset splitting ratios in machine learning algorithms: general concerns for data scientists and data analysts." In 7th International Mardin Scientific Research Conference, pp. 496-504. 2022.

[15] Thanapol, Panissara, Kittichai Lavangnananda, Pascal Bouvry, Frédéric Pinel, and Franck Leprévost. "Reducing overfitting and improving generalization in training convolutional neural network (CNN) under limited sample sizes in image recognition." In 2020-5th International Conference on Information Technology (InCIT), pp. 300-305. IEEE, 2020.