AI-Based Skin Disease Detection Using Convolutional Neural Network

Gungun Verma*[1], Vishakha Sharma[1], Yashika Yashika[1], Md. Shahid[1]

[1]Department of CSE, MIET, Meerut, India

*gungunverma.344@gmail.com, vishakha24606@gmail.com, yashikapal4544@gmail.com, md.shahid@miet.ac.in

Abstract. Skin diseases are a problem for millions of people all around the world. It is really important to find out what is wrong with someone's skin away so they can get the help they need. Usually doctors who are experts in skin problems are the ones who can tell what is wrong. Sometimes these doctors are not available to help people who live in places with limited resources. This paper is about a system that uses computers to help find skin diseases. This system was trained using a lot of pictures of skin problems called the HAM10000 dataset. There are 10,015 pictures, in this dataset. They show seven different types of skin diseases: Actinic Keratosis, Basal Cell Carcinoma, Benign Keratosis, Dermatofibroma, Melanoma, Melanocytic Nevus and Vascular Lesion. Three convolutional neural network (CNN) architectures were implemented and compared: a custom-built CNN, MobileNetV2 (transfer learning), and EfficientNet-B0 (transfer learning). The models were trained with class-weighted loss functions to address the significant class imbalance present in the dataset. The custom CNN achieved the highest validation accuracy of 54.02% and an F1-score of 0.3602 after 10 epochs. MobileNetV2 reached a best validation accuracy of 50.57%, while EfficientNet-B0 showed poor convergence under the given training configuration. Results highlight the challenges of multi-class skin lesion classification on imbalanced dermoscopy data and suggest that extended fine-tuning, data augmentation, and architecture improvements can significantly boost performance.

Keywords: Skin disease detection, CNN, HAM10000, MobileNetV2, EfficientNet, transfer learning, deep learning.

1. Introduction

Serving as the body's protective layer the skin is the biggest organ. It protects systems from many external threats. These threats include bacteria, viruses, ultraviolet radiation, and pollutants. The skin has an important role in protecting us but the skin is also very susceptible to many different kinds of diseases, like melanoma, which is a serious disease. Globally, skin diseases are among the most frequently encountered medical conditions, with an estimated 900 million people affected at any given time. Early detection of skin diseases is really important because it helps patients get better if they have skin cancer.

Doctors usually find skin diseases by looking at the skin and using a tool called dermoscopy. There are not enough skin doctors, especially in areas where people don't have much money or live far away. This makes it hard for people to get checked early.

© The Author(s) 2026
M. Singh et al. (eds.), *Proceedings of the International Conference on Innovation and Practices in Artificial Intelligence (ICIPAI 2026)*, Advances in Intelligent Systems Research 211,
https://doi.org/10.2991/978-94-6239-789-7_19

Computer systems that use intelligence can help with this problem. They can look at skin pictures. Help find diseases quickly and at a low cost. These systems use a type of intelligence called deep learning and convolutional neural networks or CNNs for short. They can help check for skin diseases fast and make it easier to find doctors. Skin cancer and other diseases can be found early with the help of these computer systems.

In the few years CNNs have done a great job with medical image analysis tasks. They are usually as good as or even better than people at doing these tasks. We have collections of dermoscopy images like HAM10000 that are labelled correctly. This has helped us make a lot of progress in this area. HAM10000 has a lot of dermoscopy images. 10,015 To be exact. These images show seven kinds of skin problems. This is really helpful for making and testing systems that can find skin lesions. CNNs are getting better and better at looking at images and HAM10000 is a great tool for this.

This paper is about a system that uses intelligence to detect skin diseases. The system looks at three ways to do this using something called CNN architectures on a big set of data called the HAM10000 dataset.

- It uses a multi-layer system that was built from nothing.
- It uses a system that is based on something called MobileNetV2.
- It uses a system that is based on something called EfficientNet-B0.

The skin disease detection system trains these models using a kind of loss to deal with the fact that there are a lot more examples of one type of skin disease called Melanocytic Nevus, which is about 66.9 percent of all the examples in the dataset. The system is made to be used on the web so people can upload pictures of skin lesions. Get a prediction of what disease they might have and this happens in real time using something called the Flask framework.

The remainder of this paper is structured as follows:

- Section II looks at what other people have written about this topic
- Section III explains the method we are using
- Section IV talks about the computer systems we used for this
- Section V shows the information we used to test these systems
- Section VI shows what we found out and what it means
- Section VII sums up what we did and what we think we should do next

People have been using deep learning to try to classify skin diseases for about ten years now.

2. Literature Review

The application of learning to skin disease classification has been very popular. There have been a lot of studies, on this topic that have helped us get to where we are today with skin disease classification. In a seminal contribution to the field, Esteva et al. [1] showed that a CNN trained on a large clinical image corpus could achieve diagnostic accuracy on par with board-certified dermatologists across two binary classification tasks: separation of keratinocyte carcinomas from benign seborrheic keratoses, and the identification of malignant melanomas against benign nevi. This work served as a foundational proof-of-concept for applying end-to-end deep learning to automated skin disease diagnosis.

Codella et al. [8] introduced the HAM10000 dataset as part of the ISIC 2018 challenge. They provided a dataset of skin lesion images that belonged to seven different categories. This dataset helped fill a gap in skin lesion research by giving researchers standardized images to train and test their classification models.

Rathod et al. [3] created a system to classify skin diseases using a type of intelligence called CNNs. They used tools called TensorFlow and Keras to build their system. It was able to identify skin diseases about 70% of the time across five disease classes. Their work showed that it is possible to use CNNs to classify skin diseases.

Ge et al. [4] suggested an approach, to recognizing skin diseases. They used a combination of dermoscopy images and clinical images. Their method used features to identify skin diseases. They found that it can be hard to tell some skin lesions. They thought that using than one type of data might help improve skin disease recognition.

Tschandl et al. [2] looked at performance of deep neural networks on the HAM10000 dataset. They found out that the problem of class imbalance with Melanocytic Nevus being overrepresented really affects how well the networks can classify things. To make this better they suggested using class weighting and oversampling strategies.

Recent studies have looked at using versions of CNN architectures like MobileNet and EfficientNet for use on mobile devices and at the edge. MobileNetV2, which was introduced by Sandler et al. [5] uses kinds of convolutions and inverted residuals to get good results without using too much computing power. EfficientNet, which was proposed by Tan and Le [6] uses a method to scale the network in a way which means it can get the best results on the ImageNet dataset.

Afroz Rimi et al. [7] made a framework called Derm-NN using a kind of neural network that they trained on the Dermnet dataset, which has 500 pictures of five different kinds of skin diseases. They were able to get 73% accuracy, which's pretty good. Their work showed that using custom networks can be a good way to classify skin diseases even when you do not have a lot of pictures to train the network on

This study looks at this body of work [1]– [8] that has been done far and compares three different CNN architectures. A custom CNN, MobileNetV2 [5] and EfficientNet-B0 [6]. Using the HAM10000 dataset [2][8], it also deals with the problem of class imbalance by using class-weighted training.

3. Proposed Methodology

The system we are proposing follows a process that includes loading data preprocessing it handling class imbalance training the model evaluating it and deploying it. The whole workflow is described in the following subsections.

A. System Overview

Our system looks at pictures of skin from the HAM10000 dataset. It also uses a metadata CSV file that has information about each picture and what disease it really is. It loaded, resized, normalized the images and then it divide them into two groups: one group is for training and the other group is for seeing how well the system is working. We use three models that look at pictures and we train and check each one to see which one is the best. The best one gets. We use it to make a web application, with Flask that can look at new pictures and tell us what it thinks is going on in real time.

The pipeline flows top-down through eight stages:

Input → Preprocessing → Class imbalance handling → Model training (3 parallel architectures) → Evaluation → Best model selection → Flask deployment → Output

The three CNN architectures work together at the time when the system is being trained. They all use the data that has been cleaned up and the same loss that is based on the class weights. When this is all done the best model, which is the Custom CNN with an accuracy of 54.02 percent is. Made available through the Flask web app.

The notes on the side show the two things that are put into the pipeline. The HAM10000 dataset is put into the part where the data is cleaned up. The class weights that are calculated which range from 0.21 for Melanocytic Nevus to 12.44, for Dermatofibroma are put into the part that handles the imbalance. The CNN architectures and the class weights are important for the system to work properly. The Custom CNN and the HAM10000 dataset are parts of the whole process.

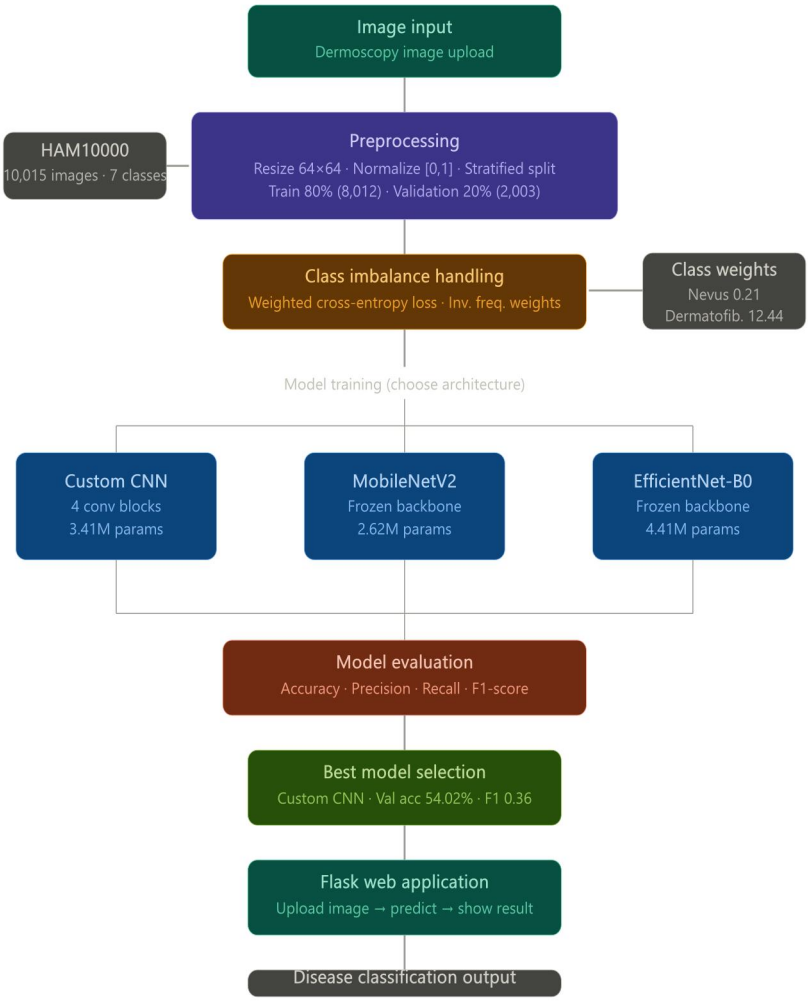


Fig. 1. Proposed System Architecture for AI-Based Skin Disease Detection.

B. Data Preprocessing

All the images are made the size, which is 64 x 64 pixels. This is done so that all the images in the dataset are the same. The color values of the pixels are also changed to be between 0 and 1. This is done by dividing the color values by 255. Labels are encoded as integer class indices corresponding to the seven disease categories. The dataset is distributed into two parts: 80% training (8,012 images) and 20% validation (2,003 images). This was done using sampling to keep the class proportions the same. The custom CNN and the other models were trained on the training part of the dataset.

C. Class Imbalance Handling

The HAM10000 dataset has a lot of class imbalance. Melanocytic Nevus makes up 66.9% of the samples in the HAM10000 dataset. To address this, class weights are computed as inversely proportional to class frequency and applied during model training via the class-weight parameter. This helps the system learn about diseases that're not common. And try to models to learn meaningful representations for under-represented conditions such as Dermatofibroma (1.1%) and Vascular Lesion (1.4%). This means the system will get better at recognizing these diseases.

D. Model Training Configuration

The Adam optimizer was used to train all three models and make the system better. The learning rate was set to 0.001 at the start. The models were trained using cross-entropy as the loss function. Each model was trained for 10 epochs. The mini-batch size was 16 samples. The model training process needed a help to not get stuck. So, a learning rate scheduler was added. This scheduler halved the learning rate whenever the validation loss did not get, over three epochs. The model training process also used a checkpointing mechanism. This mechanism saved the weights of the model when it had the validation accuracy. This way the best model weights were always retained throughout the training of the models. The models were trained to be the best they could be.

4. CNN Architectures

A. Background of CNN

Convolutional Neural Networks are a kind of deep neural network. They are really good at handling data that has a structure to it. The way they work is by using the filters in different places to look for things that are near each other. This helps them find things even if they are not in the spot every time. Because of this Convolutional Neural Networks need a lot things to learn about compared to other kinds of networks. A basic Convolutional Neural Network has three parts. Convolutional Neural Networks have layers that help find important features. They also have pooling layers that make the picture smaller. They have fully connected layers that figure out what something is, at the very end.

The convolutional layers look at the images and find patterns. They do this by using filters that can be changed as the computer learns. These filters help the computer see what is in the images. The batch normalization helps the computer learn faster. It does this by making sure that the numbers the computer is working with are not too big or too small. The dropout layers help the computer not get too good at recognizing the images in the dataset. This is done by turning off some of the parts of the computers brain during

training. The final fully connected layers aggregate extracted features and produce class probability distributions via the softmax activation function.

B. Custom CNN Architecture

The custom CNN is a deep sequential architecture. It structured around four convolutional processing blocks that transition into two classification-oriented dense layers. Each block made up of two Conv2D layers (with 3x3 kernels and ReLU activation), two Batch Normalization layers, a MaxPooling2D layer (2x2), and a Dropout layer (rate = 0.25). The filter depths across the four blocks are 32, 64, 128, and 256, respectively. The flattened feature maps are then processed through dense layers of 512 and 256 units, each regularized with Batch Normalization and Dropout, before the final softmax layer produces a 7-class probability distribution. The total trainable parameter count is approximately 3.41 million (13.01 MB).

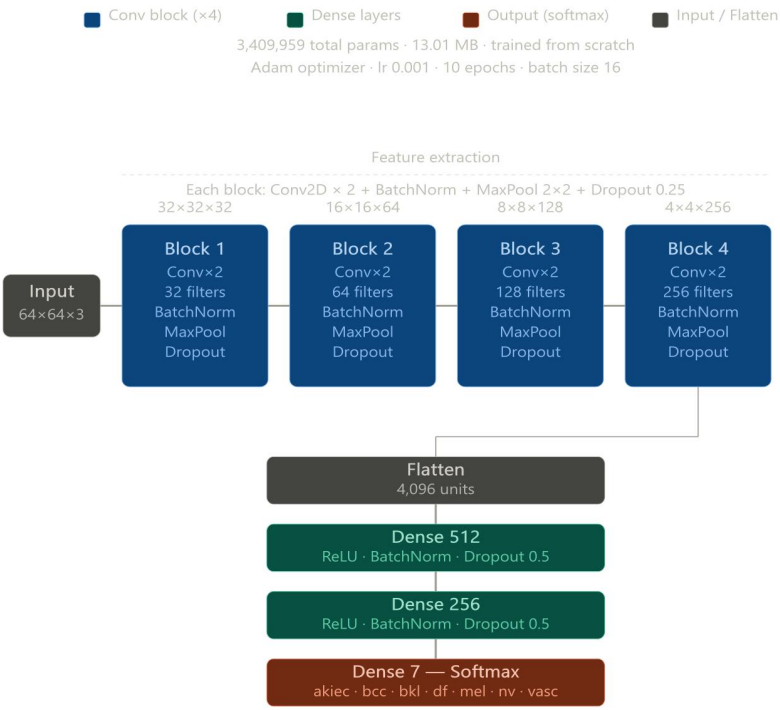


Fig. 2. Custom CNN Architecture Diagram.

C. MobileNetV2 Architecture

MobileNetV2 is a lightweight CNN architecture. These blocks are made up of depthwise convolutions and inverted residual blocks with linear bottlenecks. We use a -trained MobileNetV2 model that was trained on a huge set of pictures called ImageNet. We take out the layers of this model and use the rest as a fixed feature extractor. The features we get from this model are then passed through a Global Average Pooling layer. After that we use some layers with 256 and 128 units and we also use Batch Normalization and Dropout to help the model work better. Finally we use a 7-class softmax output to get the result. Only the added classification head layers are trainable (362,503 parameters), while the MobileNetV2 backbone weights (2,257,984 parameters) remain frozen.

D. EfficientNet-B0 Architecture

The EfficientNet-B0 is the model of the EfficientNet family. It is special because it can scale up or down in terms of how deep wide and detailed it's using something called a compound coefficient. We use a -trained EfficientNet-B0 model as a frozen feature extractor, which means we do not change its weights. Then we pass the features through a Global Average Pooling layer followed by some layers with 256 and 128 units and Batch Normalization and Dropout. Finally we use a 7-class softmax output to get the result like, with the MobileNetV2 model. The architecture has 4,412,842 total parameters, of which only 362,503 are trainable.

5. Image Dataset

A. HAM10000 Image Dataset

The HAM10000 dataset is a collection of skin pictures that people can use for free. It was put together by the International Skin Imaging Collaboration or ISIC for short. The HAM10000 dataset has 10,015 pictures of skin problems. It also comes with a file that has information about each picture like what is wrong with the skin and who the patient is. The HAM10000 dataset was first made public as part of a challenge called the ISIC 2018 Skin Lesion Analysis Challenge. Now the HAM10000 dataset is widely used to test how well computers can tell what is wrong with skin.

All the pictures in the HAM10000 dataset were taken from groups of people at many different places. The HAM10000 dataset has a lot of variety with types of skin and different kinds of skin problems. Each picture in the HAM10000 dataset is labelled with one of seven types of skin problems so computers can practice figuring out what is wrong, with the skin.

B. Disease Categories

The dataset includes seven types of skin diseases:

(A) Actinic Keratosis (akiec): These are pre-cancerous lesions. They are caused by much exposure to the suns UV rays. There is a risk that they can turn into cell carcinoma.

(B) Basal Cell Carcinoma (bcc): It begins in the lower part of the epidermis, which is the outer layer of skin. It appears on sun-exposed areas of skin.

(C) Benign Keratosis (bkl): A group of non-cancerous skin growths. They include keratoses and solar lentigines.

(D) Dermatofibroma (df): These are benign fibrous nodules that are usually found on the lower legs.

(E) Melanocytic Nevus (nv): These are moles. They are the common type of skin lesion and make up the largest part of the dataset.

(F) Vascular Lesion (vasc): They are abnormal growths, clusters of blood vessels (capillaries, veins, or arteries) that appear on the skin of the body.

(G) Melanoma (mel): It develops from the melanin, producing cells known as melanocytes.

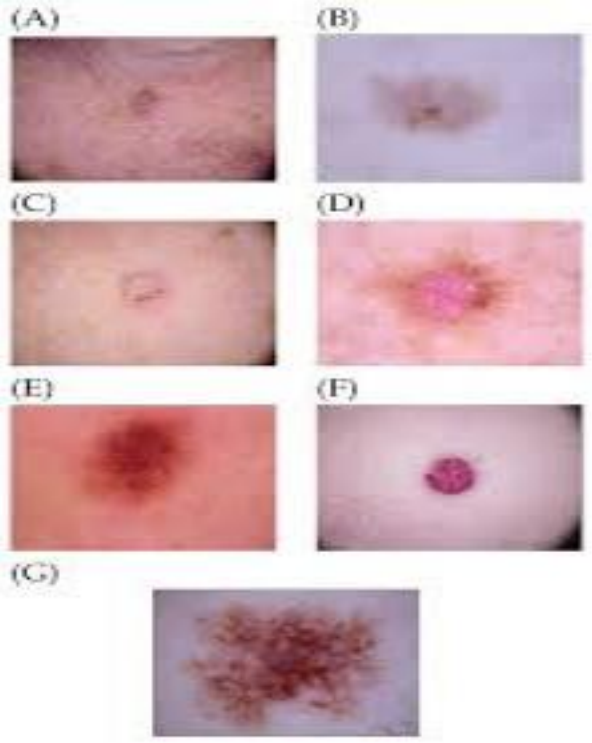


Fig. 3. HAM10000 Image Dataset

C. Class Distribution and Imbalance

As illustrated in Table I, the dataset is not balanced. Melanocytic Nevus makes up 66.9% of it with 6,705 samples while Dermatofibroma only makes up 1.1% (115 samples). To deal with this issue we used something called class-weighted training. This means we gave penalty to the classes that do not have a lot of samples so the model does not favor the class with the most samples.

Table 1. HAM10000 Class Distribution and Computed Class Weights

Class	Count	Percentage	Class Weight
Actinic Keratosis	327	3.3%	4.3686
Basal Cell Carcinoma	514	5.1%	2.7848
Benign Keratosis	1099	11.0%	1.3021
Dermatofibroma	115	1.1%	12.4410
Melanoma	1113	11.1%	1.2860
Melanocytic Nevus	6705	66.9%	0.2134
Vascular Lesion	142	1.4%	10.0401

Source: HAM10000 Dataset Metadata

D. Data Split

The HAM10000 dataset has a total of 10,015 images. We divided these images into two groups. The first group is the training set, which has 8,012 images. This is 80 % of the number of images, while the other group is the validation set having 2,003 images which is 20 % of the number of images. We made sure that the two groups have the balance of classes as the whole dataset by using a method called random splitting. This way no single class has many or too few images, in either group. We did not set aside a test set because we used the validation set to check how well the models were doing after each epoch. We tested all three CNN models, which're the Custom CNN, MobileNetV2 and EfficientNet-B0 on the same groups of images. This allows us to compare the Melanocytic Nevus and Dermatofibroma results of the three CNN models fairly.

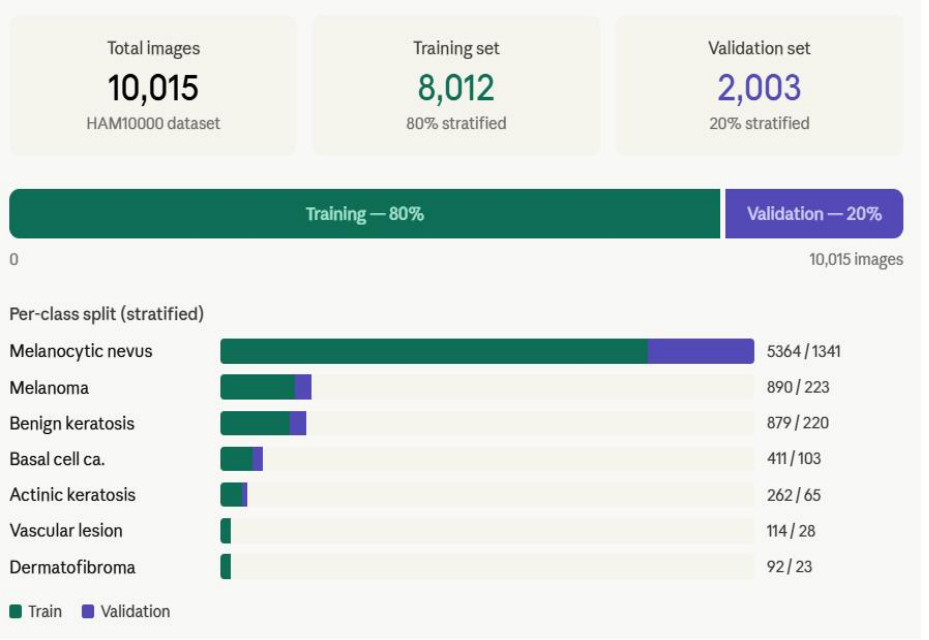


Fig. 4. HAM10000 Class Distribution (Training + Validation).

6. Results and Discussion

A. Training Performance

The training pipeline for all three architectures consisted of 10 epochs, with Adam selected as the optimization algorithm at a learning rate of 0.001. The custom CNN model got better and better at recognizing things as it trained. It started with an accuracy of 21.52% at Epoch 1. By Epoch 10 the custom CNN model was correct 46.34% of the time when it was training and 54.02% of the time when we tested it. The loss was 1.1546 at the end. However, the custom CNN model did its best at Epoch 9 with of 49.13% accuracy.

The MobileNetV2 model was more stable. It started with an accuracy of 19.27% at Epoch 1. Ended with 44.81% at Epoch 10. It did its best at Epoch 5 with an accuracy of 50.57% and a loss of 1.3023. The ReduceLROnPlateau callback triggered a learning rate reduction to 0.0005 at Epoch 10.

The EfficientNet-B0 model did not do well. It had trouble recognizing things. Its accuracy was, below 14% when it was training. When we tested it the accuracy was very low, 1-5%. This is probably because the model was not adapted to the dermoscopy domain and the pictures were too small 64x64 pixels. The EfficientNet-B0 model works best with pictures 224x224 pixels.



Fig. 5. Training performance over 10 Epochs for all Three Models.

B. Model Comparison

Table 2 shows what happened with the three models after they were trained for 10 epochs and then tested on the validation set.

Table 2. Model Performance Comparison on HAM10000 Validation Set

Model	Accuracy	Precision	Recall	F1-Score
Custom CNN	0.3110	0.4700	0.3110	0.3602
MobileNetV2	0.2776	0.4674	0.2776	0.3396
EfficientNet (B0)	0.0120	0.0066	0.0120	0.0012

Evaluation of 2,003 validation images after training of 10 epochs.

The CNN model did better than the two models the MobileNetV2 and the EfficientNet-B0 models when it came to accuracy (31.10%), precision (0.4700), and F1-score (0.3602) during the final evaluation. The custom CNN model did better than the two models the MobileNetV2 and the EfficientNet-B0 models when it came to accuracy (27.76%), F1-score (0.3396), while The EfficientNet-B0 model did not do very well with an accuracy of only 1.20 % and an F1-score of 0.0012, when we looked at the evaluation metrics at the end of the training of the EfficientNet-B0 model and the other two models.

The relatively low evaluation accuracies compared to the in-training validation accuracies may be partly attributable to class imbalance: in a heavily imbalanced dataset, a model that predominantly predicts the majority class (Melanocytic Nevus) will achieve moderate epoch-level accuracy but poor macro-averaged precision, recall, and F1-score. The use of class weights during training partially addresses this but is insufficient to overcome the extreme imbalance within 10 epochs.

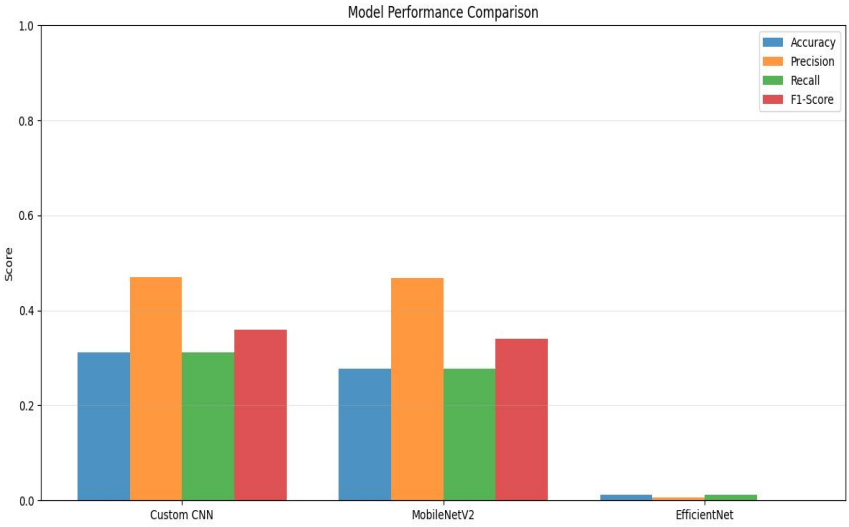


Fig. 6. Model Comparison

7. Discussion

Several observations emerge from the experimental results. First, the custom CNN trained from scratch shows competitive performance compared to transfer learning models when the backbone is frozen and the input size is small (64x64). Transfer learning is most beneficial when the backbone is fine-tuned on domain-specific data; with frozen weights and a mismatched input resolution, models like EfficientNet-B0 underperform.

Second, class imbalance remains the dominant challenge in HAM10000 classification. Even though we used class-weighted training the Melanocytic Nevus class, which makes up 66.9 percent of the images still has an influence on the predictions especially when the models are not trained very well. If we use some techniques like SMOTE or generate images by changing the ones we have like flipping or rotating them or making them brighter or darker it will probably help the models do better on the classes that do not have as many images.

Also if we train the models for epochs and make some changes, to the pre-trained parts of the models and use bigger images like 224x224 it will probably make a big difference, especially for the MobileNetV2 and EfficientNet-B0 models, which were designed to work with higher-resolution inputs.

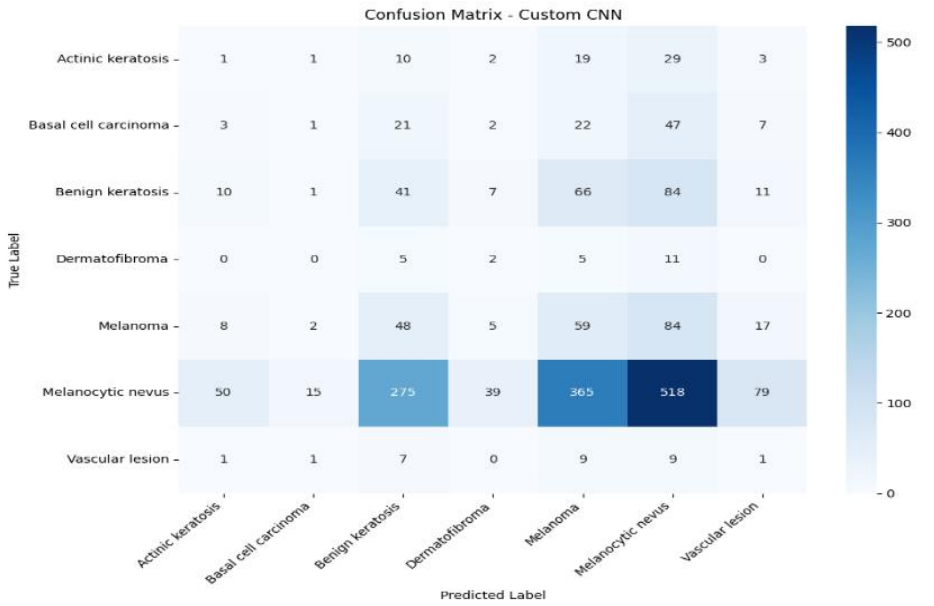


Fig. 7. Normalized Confusion Matrix for Custom CNN on Validation Set.

8. Conclusion

This paper is about a system that uses intelligence to detect skin diseases. The system was trained on the HAM10000 benchmark dataset. It used three kinds of convolutional neural network architectures. These are a custom CNN, MobileNetV2 and EfficientNet-B0. The goal is to look at pictures of skin and figure out what disease it is. There are seven kinds of skin diseases it tries to identify including some really bad ones, like Melanoma and Basal Cell Carcinoma.

The CNN model was able to get the results with a validation accuracy of 54.02% and an F1-score of 0.3602 after the CNN was trained for 10 epochs with class weights. MobileNetV2 showed comparable results, while EfficientNet-B0 failed to converge meaningfully under the current training configuration. The results show that it is really hard to classify skin diseases from pictures especially when there are not pictures of some diseases. It is also hard when the pictures are not very clear.

Future work will focus on: (1) making the pre-trained backbone layers work better for our specific needs; (2) using pictures that are 224x224; (3) trying different ways to change the pictures like rotating them flipping them making them bigger or smaller and changing the colours; (4) finding ways to deal with the fact that some types of skin disease are more common than others in our pictures; (5) combining multiple models to make the system more reliable; and (6) testing the Flask web application in a real world setting with doctors and patients. With these improvements, the system has a lot of potential to be a tool for doctors to help them find skin disease, low-cost clinical decision support tool for early skin disease detection.

REFERENCES

1. Esteva, A., Kuprel, B., Novoa, R. A., Ko, J., Swetter, S. M., Blau, H. M., & Thrun, S. (2017). Dermatologist-level classification of skin cancer with deep neural networks. Nature, 542(7639), 115-118.
2. Tschandl, P., Rosendahl, C., & Kittler, H. (2018). The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions. Scientific Data, 5, 180161.
3. Rathod, J., Wazhmode, V., Sodha, A., & Bhavathankar, P. (2018). Diagnosis of skin diseases using convolutional neural networks. In 2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA), pp. 1048-1051. IEEE.
4. Ge, Z., Demyanov, S., Chakravorty, R., Bowling, A., & Garnavi, R. (2017). Skin disease recognition using deep saliency features and multimodal learning. In International Conference on Medical Image Computing and Computer-Assisted Intervention, pp. 250-258.

5. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). MobileNetV2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4510-4520.
6. Tan, M., & Le, Q. V. (2019). EfficientNet: Rethinking model scaling for convolutional neural networks. In International Conference on Machine Learning (ICML), pp. 6105-6114.
7. Afroz Rimi, T., Sultana, N., & Ahmed Foysal, M. F. (2020). Derm-NN: Skin diseases detection using convolutional neural network. In Proceedings of the International Conference on Intelligent Computing and Control Systems (ICICCS 2020), pp. 1205-1209. IEEE.
8. Codella, N. C. F., Gutman, D., Celebi, M. E., Helba, B., Marchetti, M. A., Dusza, S., et al. (2018). Skin lesion analysis toward melanoma detection: ISIC 2017 challenge. In 2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI), pp. 168-172.

Open Access This chapter is licensed under the terms of the Creative Commons Attribution-NonCommercial 4.0 International License (http://creativecommons.org/licenses/by-nc/4.0/), which permits any noncommercial use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.

