DERMATOLOGIST-AIDED SKIN TYPE DETECTION USING CONVOLUTIONAL NEURAL NETWORK.

ABSTRACT

Detection of skin types is an important/critical step in diagnosing dermatological issues and treatment planning, this influences the choice of treatment and for recommending products, even though self-assessment is not 100% reliable. This research work explores the development of a computer-aided system for skin type detection using Convolutional Neural Network (CNNs). This CNN model will be trained on the skin type dataset (normal, dry, oily) this dataset is collected by dermatologists worldwide, The ResNet-18 and a custom CNN model I named MyCNN models was used to extract and learn complex features from the skin type dataset and train it to enable accurate skin type classifications. Standard metrics was used for model's performance evaluation also to demonstrate significant improvement over existing methods. MyCNN model performed better overall with 40% accuracy against ResNet-18 33% accuracy. The results obtained shows that dermatologist-aided CNNs offer a better promising solution for automated skin type detection, it has potential applications in epidemiological studies, The process of integration of Artificial Intelligence and expert insights presents a step forward in dermatological diagnostics, enhancing both precision and the efficiency of skin types and their classification.

Keywords—Skin type, Artificial Intelligence, Convolutional Neural Network (CNN), ResNet-18

II. INTRODUCTION

Our skins as humans adapt and readjust to the weather, we find ourselves, from being dry and flaky during intense harsh weathers like winter to being very oily during humid summers, we lose our skin protecting barriers during this season and hence it affects our looks and even has an effect on our overall lifestyle. we experienced this moving to the UK to study from my previous country, looking for a solution to my skin problems led me online to look for products that can help me, but not being able to Identify the skin type I have was an issue to buying the right product and that is what this paper addresses, I created a simple classification model that can be incorporated into an app or web, that can easily detect the skin type of a customer and proceed to offer suggestions for the kind of product that would work for them, or maybe a dermatologist that is trying to know the skin type of their patient without a physical contact can easily pass several images of the patient through the model and the model would predict to a certain percentage what skin type it is and offer further assistance in prescribing required drugs or lotion for the patients, the method of physical assessment can be time consuming and subjective.

we used a custom convoluted neural network (myCNN) and a pre- trained CNN architecture called the Resnet-18, we used both to propose a deep learning approach for skin type classification after which we compare the performance of both models in detecting and classifying new skin types. The input to our algorithm is an image, we then use both myCNN model and ResNet-18 model to predicted skin type.

Aim: - To train and develop two CNN models that is able to classify skin types and then compare their performance and choose the best for classification.

Objective: - develop and train CNN models for classifying skin types from images dataset, evaluate and compare the performance of the two models.

III. RELATED WORK

Skin type classification using CNN models have gained more interest recently and at the invention of artificial intelligence this is because of its application in making medical processes much easier we will be comparing existing research categorizing them and comparing them to our work using custom CNN and ResNet-18.

1. Human skin type classification using image processing and deep learning approaches [1] Sirawit Saiwaeo et al. (2023)

This method utilizes CNN architectures like MobileNet-V2, EfficientNet-V2, InceptionV2 and ResNet-V1.

Strength and weakness

- It used different pretrained models (up to 4) to compare and contrast and get the best score, it is already pretrained so it will reduce computational cost and there will also be an improvement in performance compared to training from scratch.
- There is possible occurrence of overfitting and black box effect which can affect interpretability

2. Cosmetic Skin Type Classification Using CNN With Product Recommendation [2]

Arya Kothari et al. (2021)

Strength and weakness

- It used different classification algorithms like CNN and SVM
- it did not focus on explaining the deep learning processes or used facial images did not compare different models to know which performed better also SVM require large memory and computational time to deal with large dataset.

3. Modelling of Human Skin by the Use of Deep Learning [7]. Xin Xiong et al. (2021)

Strength and weakness

- it uses ISVM instead of SVM to reduce training and updating time also accuracy.
- It only focused on a specific dataset and might have a problem with generalizing.

V. DESCRIBE THE DATASET

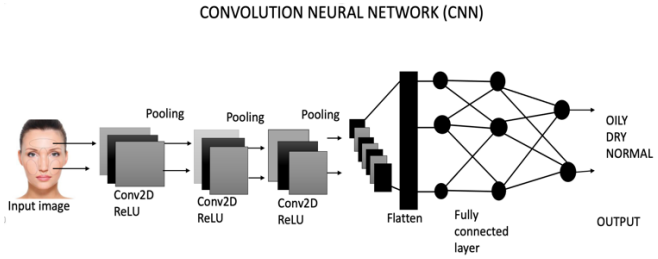
The dataset is a dataset is a dataset by Shakya Dissanayake. (2021). Oily, Dry, and Normal Skin Types Dataset we got from Kaggle [6], it is made up of images of skin types namely normal, oily and dry. It has a total of 2,890 images of different skin types from people of different colour and race, it has a resolution of 640 x 640 and a JPEG image format. For the oily skin type we have a total of 1,120 images, for the dry skin type we have a total of 758 images and for the normal skin type we have a total of 1274 images. This is a sample from the dataset showing skin types:

Splitting the dataset: -
I split my dataset into training, test and validation. We used 2480 for train, 276 for validation and 134 for test. We didn't apply data augmentation and normalization; the original image size was 640 x 640 so we resized to 224 x 224 to allow easy running of the process since we don't have enough gpu to run high resolution images.

VI. METHODOLOGY

This method uses specific deep learning algorithm like myCNN and ResNet-18 to perform classification on our skin types dataset. The input data being images from the skin type dataset classified into Normal, oily and dry. we extracted information from these images and use it. The CNN model works by simply transforming the input which, is an image and a forward pass performed into a feature map and then processed through three convolutional blocks and pooling layers to return a predicted output/result.

My CNN: - For myCNN model the aim was to train a convolutional neural network model for classification of image to be either oily, normal or dry category. This model will extract features from the images and use these features to classify them into one of the three classes.
Firstly, we downloaded the dataset from Kaggle and proceeded to resizing it, the original size was 640 x 640 but we resized to 224 x 224 for both train and test to enable it run smoothly without consuming excess gpu, after that we split to test and train sets, after which we applied data augmentation (horizontal flip) to improve our model's robustness, we used base transform to normalize pixel and convert images to Pytorch tensors, we set our batch size to 32 for a faster process and less memory usage then our data loaders as 2. We proceeded to building our model named the class MyCNN model, created an instance of the class and constructor then container for my fully connected layer. my first convolutional block defines our 2D convolutional layer with input channels as 3 representing 3 color channels (RGB) our output channels as 64 which are the number of filters and padding as 1, we applied rectified linear unit (ReLU) activation function after the convolution.
Our maxpool was applied with kernel size of 2x2 and stride of 2 and padding set to zero. This process is repeated with increasing output channel (64 -> 128 -> 256) i.e., for second block 2D convolutional layer 128 and for third block 2D convolutional layer 256 and ReLU 4x4. [3]
For the fully connected layers we added the flatten function to flatten our outputs from convolutional layers into a single vector. This converts it into a 1D vector i.e., transforms the data from a 2D to a collection of features and makes it more suitable for fully-connected layers, we defined our first fully connected layers with 256*14*14 and output size 256 and also our ReLU activation, we defined a second 256 neuron fully connected layer with ReLU activation and our final fully connected layers with 7 outputs. Finally, we defined our forward pass method which takes and input image pass it through our convolutional layers and gives an output.
We created an instance for cross entropy loss function from PyTorch, this is the loss function that will be used to for measurement of our model's predictions against the true values during training, mathematically cross entropy loss function can be defined as
L = -sum (t_i x log (p (t_i | s_i)
where **t** = true label for the i-th sample and
p (t_i | s_i) he model's predicted probability for the correct class given the input **s_i** and also Adam optimizer.



ResNet-18: -
The ResNet-18 model is a pretrained convoluted neural network model that is 18 layers deep. [4] this pretrained model has been trained on more than a million images from ImageNet database and classifies images into over 1000 image categories.
For the ResNet-18 we took advantage of the knowledge that the pretrained ResNet-18 on large dataset and applied to help improve our classification task which is just 3 categories which the ResNet-18 is pretrained on 1000 categories. We will fine tune the final layers of the model so it is able to adapt and learn new features in our classification tasks.
For our task we loaded the pretrained model from the **torchvision.models** module and set it to true, we modified the final layer of the fully-connected layer of the model with a with our trained dataset after which we move the model to our chosen device i.e., cuda to ensure they run on appropriate hardware for efficiency.
We created an instance for cross entropy loss function from PyTorch, this is the loss function that will be used to for measurement of our model's predictions against the true values during training, mathematically cross entropy loss function can be defined as **L = -sum(t_i x log(p(t_i | s_i)**

where **t** = true label for the i-th sample and **p(t_i | s_i)** the model's predicted probability for the correct class given the input **s_i** and also Adam optimizer.

VII. EXPERIMENTAL RESULTS AND DISCUSSION

For our training strategy we proceeded to implement our training loop for our model, we defined the number of training epochs i.e., cycles our model will go through, due to our limited size of gpu we could only use few Epochs to enable the model run faster, we created our empty lists for recording information, our train loss epoch for storing average training loss per epoch, train accuracy epoch for accuracy, valid loss epoch for validation loss and valid accuracy epoch for accuracy epoch.

We then set our model to training mode, record information for training loss and training accuracy, we prepare the data to be fed into the device using forward pass, add our cross-entropy loss function to calculate the loss between predicted and true, loss backward function to compute gradients for each weight and any bias found. We used our Adam optimizer to update weight and biases of the model then clear previous gradient to prevent accumulation.

After which we calculated, recorded, and validated our epoch information.

We did not perform hyper parameter tunning and cross validation search because of our limited allocated gpu.

Result Evaluations: -

Our primary metrics are Accuracy, Precision, F1 score and Recall. [5]

1. Precision: - it is the calculated ratio between predicted positive observations to the total predicted positives, it measures model accuracy in classification and can be denoted as: -

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

2. Accuracy: - this generally describes the model's performance across all classes of classifications

$$Accuracy = \frac{True_{positive} + True_{negative}}{True_{positive} + True_{negative} + False_{positive} + False_{negative}}$$

3. F1 score: - this score metric measures a model's accuracy as well and it's simply a combination of averages of precision and recall.

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

4. Recall: - this metric is calculated as the ratio of correct predictions to the actual predictions.

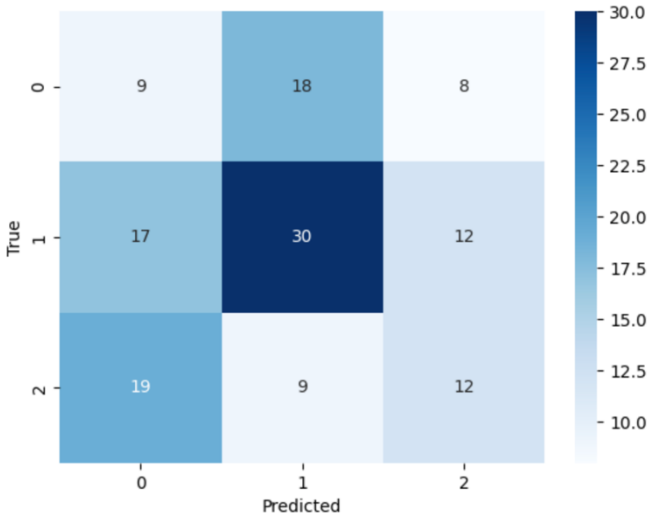
$$Recall = \frac{True_{positive}}{True_{positive} + False_{negative}}$$

Precision (ResNet-18)	Recall (ResNet-18)	F1-Score (ResNet-18)	Support (ResNet-18)
0.23	0.23	0.23	35
0.47	0.46	0.47	59
0.21	0.23	0.22	40
0.33	0.33	0.33	134
0.31	0.30	0.30	134
0.33	0.33	0.33	134

Class	Precision (MyCNN)	Recall (MyCNN)	F1-Score (MyCNN)	Support (MyCNN)
0	0.20	0.26	0.23	35
1	0.53	0.51	0.52	59
2	0.38	0.30	0.33	40
Accuracy	0.38	0.38	0.38	134
Macro Avg	0.37	0.36	0.36	134
Weighted Avg	0.40	0.38	0.39	134

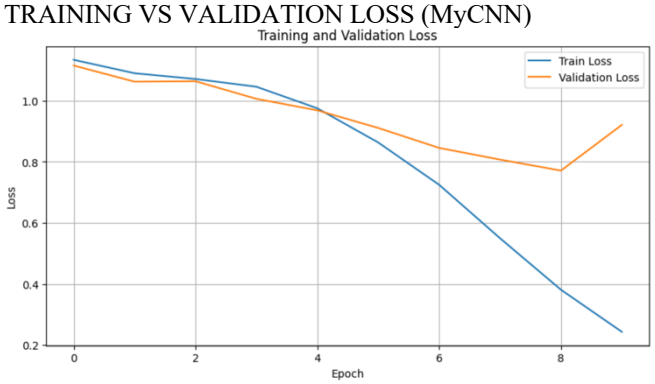
MyCNN model outperforms ResNet-18 model in all metrics Accuracy, F1-Score, Precision and Recall in both Class 1 and Class 2. My CNN has a higher accuracy of 38 – 40% compared to ResNet-18 score of 33% and shows better macro and weighted averages.

CONFUSION METRICS FOR MyCNN



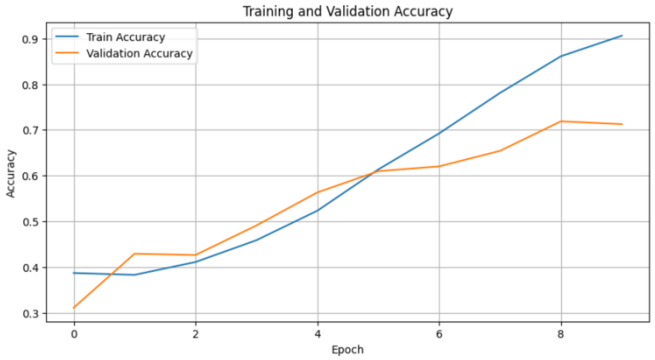
From the chart MyCNN model got an accuracy of 38 – 40%. It predicted 30 instances in class 1 and 12 instances in class 2 which is higher than ResNet-18.

TRAINING VS VALIDATION LOSS (MyCNN)



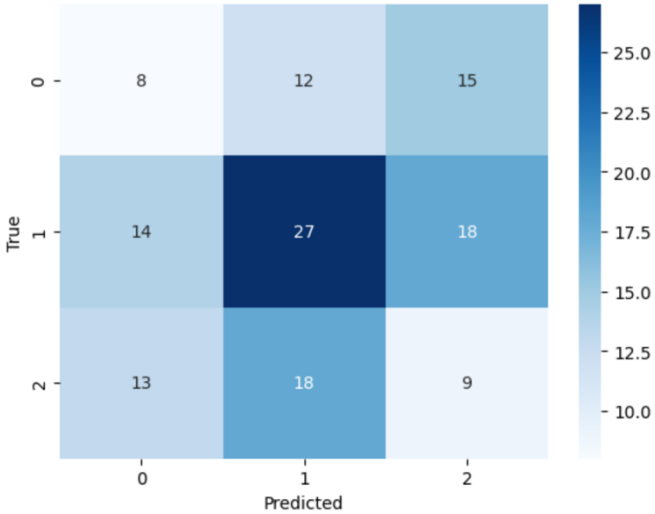
From the graph we can see that the training loss decreases steadily with the increase of number of Epochs, this shows the model learning from training and improving on the test while the validation loss shows decrease but starts increasing around 8 Epochs this is a sign of overfitting.

TRAINING VS VALIDATION ACCURACY (MyCNN)



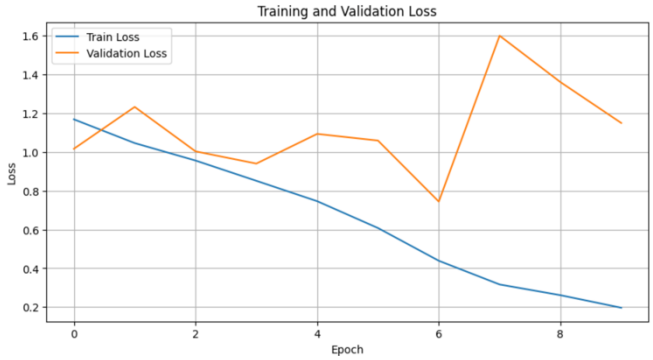
From the graph we can see that the model learns from the beginning during the initial epochs, but starts overfitting around Epoch number 8, this is because validation accuracy no longer increases and is likely to decrease, while training accuracy continues to rise.

CONFUSION METRICS RESNET-18



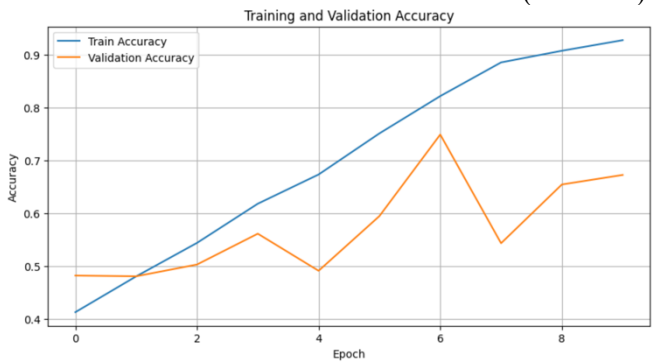
From the chart, the ResNet-18 model has shown higher accuracy for only Class 1 with 27 numbers of correct predictions, although there is some misclassification in class 1 and 2.

TRAINING VS VALIDATION LOSS (RESNET-18)



For the graph both training and validation losses decrease and shows a good learning rate. However, after the 6th Epoch there is an increase in validation loss while training loss continues to decrease showing signs of overfitting.

TRAINING VS VALIDATION ACCURACY(RESNET)



From the graph both training and validation are increasing indicating a better leaning and generalizations, this changes on the third Epoch which reduces validation accuracy i.e., it begins an irregular upwards and downwards movement while train accuracy continues to increase, this is another sign of model overfitting

VIII. CONCLUSION

We built our deep learning models to be able to detect and predict skin types with the dataset. We firstly loaded our dataset, augmented and resized it for smooth process, split into test, validation and train, used a large batch size of 32 for easy running created our custom CNN model and pretrained ResNet-18 model, add our cross-entropy loss and optimizer, created our training loops and epoch size to 10, this is because of our very limited resources used test loader to compare our results.

For the both models, MyCNN seems to perform much better than the ResNet-18 and this is due to the fact that it is already a pretrained model and was not reconfigured by us, unlike the CNN model we built from scratch.

Limitations we had was extremely limited computational resources, we were not able to do a proper cross validation to improve score or reduce batch size and workers or use a greater number of Epochs also there was not enough time for the project. If we had enough computational resources, more team members and enough time we would have been able to explore the dataset further to perform deep learning steps like cross validation, increase number of Epochs, reduce number of batches to get better score

REFRENCES

[1] Human skin type classification using image processing and deep learning approaches, Sirawit Saiwaeo et al. (2023)

[2] Cosmetic Skin Type Classification Using CNN With Product Recommendation, Arya Kothari et al. (2021)

[3] Transfer learning and fine-tuning,
https://www.tensorflow.org/tutorials/images/transfer_learning

[6] Oily, Dry and Normal Skin Types Dataset, Shakya Dissanayake, (2024).

[7] Modelling of Human Skin by the Use of Deep Learning, Xin Xiong et al. (2021)

[4] www.mathworks.com. (n.d.). *ResNet-18 convolutional neural network - MATLAB resnet18*. [online] Available at: https://www.mathworks.com/help/deeplearning/ref/resnet18.html#References [Accessed 24 May 2024].

[5] Ahmed Fawzy Gad (2020), Evaluating Deep Learning Models: The Confusion Matrix, Accuracy, Precision, and Recall