IIETA International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Traitement du Signal
Vol. 41, No. 5, October, 2024, pp. 2623-2633
Journal homepage: http://iieta.org/journals/ts

A Deep Convolutional Neural Network (DCNN) with Fine Tuned Hyper Parameters for Skin Cancer Classification

Abdul Rahaman Shaik[1,2*], Rajesh Kumar Pullagura[1]

[1] Department of ECE, AU College of Engineering, Visakhapatnam 530003, India
[2] Department of ECE, Vishnu Institute of Technology, Bhimavaram 534202, India

Corresponding Author Email: abdulrahman.s@vishnu.edu.in

https://doi.org/10.18280/ts.410535

Received: 13 September 2023
Revised: 14 March 2024
Accepted: 12 July 2024
Available online: 31 October 2024

Keywords:
neural network, skin cancer, classification, normalization, dropout, data augmentation

ABSTRACT

Skin cancer is increasingly becoming a prominent community health hazard across the globe, and detecting its onset very early gives an edge in advancing patient care quality. This work introduces a novel approach, an advanced convolutional neural network model rooted in deep learning to effectively classify skin lesions, with the primary focus on improving accuracy by fine tuning the hyper parameters. The model is fitted and assessed using the HAM10000 dataset. The dataset has 10,015 dermoscopic pictures, encompassing a range of skin manifestations. To enhance the accuracy of our model, we employed several techniques, including batch normalization, dropout, data augmentation, and data balancing. The proposed model outperformed several existing models and achieved an impressive accuracy of 96% in classifying skin lesions, demonstrating its role as a key asset in assisting dermatologists and clinicians in diagnosing skin cancer.

1. INTRODUCTION

Skin cancer is distinguished by its high occurrence rate, experiencing a global surge in its incidence. Prompt and precise skin lesion diagnosis is vital for better treatment and patient results. Dermoscopy, a non-invasive imaging technique, has emerged as a beneficial aid for assisting dermatologists in diagnosing dermal malignancy in the initial stages. Manually interpreting dermoscopic images is time-intensive and susceptible to observer differences.

Recent progress in deep learning holds significant potential for automating skin lesion classification. Convolutional deep learning networks which are commonly abbreviated as CNNs, have exhibited impressive prowess in accomplishing image recognition tasks. In this study, we aim to harness the power of deep CNNs to develop a well calibrated and swift model for the identification and labelling of skin malignancies.

We utilized the HAM10000 dataset, containing a comprehensive range of 10,015 dermoscopic pictures. The dataset encompasses seven categories of skin deformities. The proposed model's effectiveness is enhanced by incorporating several techniques. Batch normalization was applied to ensure stable learning and faster convergence during the training process. Dropout regularization was utilized to reduce the risk of the model fitting to random noise in the training data and to enhance its robustness on data it hasn't encountered before.

In addition, strategies like data augmentation were employed to the training dataset. This process involved applying geometric transformations that include rotation, scaling, and flipping, as well as introducing random noise to the images. This augmented dataset allowed the model to learn more robust representation of skin lesions, thereby improving its classification performance.

Furthermore, data balancing techniques were utilized to tackle the issue of skewed class distribution in the data. As skin cancer is a relatively rare condition compared to benign lesions, imbalanced class distributions can adversely affect the model's performance. By oversampling the minority classes and down sampling the prevalent class, We strived to correct the imbalance among classes and uphold fair representation across the board during training.

The outcomes of our investigations showcase the efficacy of our proposed model. Our DCNN produced an impressive accuracy of 94% when the count of images per each label (sample size) is 2000 per class and the accuracy increased to 96% when the sample size is increased to 2800. With its impressive accuracy our DCNN model showcases its capability as a reliable instrument in assisting dermatologists to identify skin lesions properly. The development of automated classification systems greatly helps in making accurate and timely diagnoses, potentially reducing the burden on dermatologists and improving patient care.

2. LITERATURE REVIEW

Gururaj et al. [1] used transfer learning methodology and trained their model using DenseNet169 and Resnet 50 and achieved an accuracy of 91.2% with under sampling and 83% with over sampling using a data split ratio of 80:20. Gong et al. [2] used an ensemble of multiple Convolutional Neural Networks (CNNs) in combination with Generative Adversial Networks (GANs) and achieved an accuracy of 92.6%. Zhou et al. [3] applied neural spiking architectures incorporating

spike-timing adjustment techniques resulting in an 88% accuracy rate. Son et al. [4] proposed a method in which segmentation is done first using UNET and then segmented images are classified using Efficient-Net and achieved an accuracy of 87.1%. Attention Residual Learning CNN (ARL-CNN) is used for dermatological malignancy categorization and achieved an accuracy of 85% [5]. In a study by Thurnhofer-Hemsi et al. [6], a blend of neural networks based on convolution filters are synergistically employed with a systematically applied shifting technique, resulted in a noteworthy 83.5% accuracy for classifying skin lesions. Several CNNs are combined to create an ensemble by combining the class insights offered by the separate classifiers.

Ahmed et al. [7] presented a strategy in which certain layers of ResNet-152 and ResNet-V2- Inception are finetuned with a triplet loss function. They achieved an accuracy of 84.91% with ResNet152 and with InceptionResNet-V2 the accuracy achieved is 87.42%. The HAM10000 dataset was employed with a novel segmentation approach using Shuffle-Net [8]. They calculated the cumulative moments of order one with entropy-based weighting (EW-FCM) of the skin image. Then using these moments, the required part of the lesion is isolated from the backdrop and this segmented part is fed as input to a new deep learning structure wide-Shuffle-Net and estimate the skin lesion category. This tactic achieved 86.33% accuracy. Tschandl et al. [9] developed the HAM10000 data set of skin lesions which can be used to train and test various classification algorithms. A fully convolutional residual network (FCRN) is integrated with a residual deep learning model to form a two stage frame work and achieved an accuracy of 85.5% [10].

Harangi [11] obtained 86.6% accuracy by employing a fusion of GoogleNet, AlexNet, ResNet, and VGGNet within an ensemble framework. Demyanov et al. [12] achieved 88% accuracy. A partially supervised model is presented [13]. This model has a self-advised learning mechanism for automated recognition of melanocytic carcinoma. They employed an auto-correcting SVM model to improve categorization performance by addressing the impact of incorrectly classified data, resulting in 89% accuracy. Mahbod et al. [14] used a hybrid deep neural networks that resulted in an accuracy of 84.7%. The approach revolves around harnessing the power of a pre-trained deep neural network (DNN) to autonomously extract a comprehensive array of distinctive features [15]. These features subsequently prove instrumental in the accurate diagnosis of skin lesions for malignancy. By leveraging the proficiency of the DNN, they streamlined the process of identifying key indicators within skin samples, facilitating a more refined and reliable diagnostic process for malignancy to record 93.64% of accuracy. Albahar [16] used a pre diction model with a novel regularizer that resulted in an accuracy of 97.49%. Li and Shen [17] developed a deep learning framework consisting of two fully convolutional residual networks (FCRN) to simultaneously produce the segmentation result and the coarse classification result achieving an accuracy of 91.2% on ISIC 2017 data set. The model outlined in study [18] employs image segmentation, feature extraction and synthetic minority oversampling technique (SMOTE). The extracted features are applied to various ML classifiers and Random Forest gave the best accuracy when evaluated on the ISBI dataset. Ichim and Popescu [19] proposed a novel intelligent system for melanoma classification that leverages a two-tiered structure of interconnected neural networks. Characteristics such as texture, shape, color, size are captured in the first tier and in the second tier an objective perceptron-based classifier aggregates the weighted outputs from the first tier, ultimately determining whether a lesion is melanoma or non-melanoma with an accuracy of 97.5%.

Ali et al. [20] came up with a DCNN that is evaluated on the HAM10000 data set and resulted in a validation accuracy of 91.93%. Various EfficientNet architectures were validated on the same dataset and recorded a Top-1 accuracy of 87.91% [21]. Datta et al. [22] studied the effectiveness of soft-weighted attention mechanism on the neural networks and observed that it boosted the performance by 4.7% and achieved an accuracy close to 94% on HAM10000 dataset. A high-performance data augmentation technique is used with a base line EfficientNet model and achieved 85.3% accuracy [23]. Calderón et al. [24] used a bilinear approach that includes an augmentation step and earned an accuracy of 93.21%. Qian et al. [25] introduced the concept of multi-scale attention blocks organized into a grouping (GMAB). These GMABs serve the purpose of capturing intricate details in the data, ultimately enhancing the neural network's overall performance with an accuracy of 91.6%. Murthy and Prasad [26] developed a generative model with adversarial features that is guided by transformer learning technique to proficiently categorize distinct lung cancer types. Brain tumour images are classified using ensemble classifiers that produced better results than traditional classifiers [27]. A technique that removes impulse noise from images using a residual CNN is discussed [28]. A CNN model based on segmentation and fusion techniques is developed to detect glaucoma [29].

3. METHODOLOGY

The aim of this paper is to develop a Deep Convolutional Neural Network (DCNN) whose hyperparameters are fine tuned to achieve the highest possible accuracy with the simplest possible architecture. In most of the works the hyperparameters are selected randomly, but if care is taken and extra effort is put to tune them properly, we can achieve great improvement in the performance of a DCNN network. To fine tune the hyper parameters we adopted an iterative method. Three optimizers namely Adam, SGD, Adagrad are considered for evaluation with learning rates 0.1,0.01 and dropout rates 0.2,0.4. The code snippet in python for the iterative method for tuning these parameters is provided in section 3.7. The model is designed and validated on HAM10000 image set. The DCNN consists of three Convolutional Layers. BN (batch normalization), DP (dropout) layers follow the CNN layers There are two dense layers and a flatten layer.

The DCNN is developed using Python and training and testing is done using Kaggle.

3.1 Data set

Comprising dermatoscopic images of skin lesions, the HAM10000 dataset serves as a comprehensive collection in this field. It contains 10,015 images, which were gathered from different sources and hospitals. The dataset was specifically generated to drive innovations in skin cancer screening and the identification of melanoma, with the primary goal of contributing to research progress in this field.

The HAM10000 dataset was compiled by a group of researchers from the Medical University of Vienna and the Austrian Society for Dermatology. The dataset includes

images of seven different types of skin lesions, with metadata providing information about the spread of the disease across sex, gender, and age groups.

Each pic in the HAM10000 image set is accompanied by various clinical metadata and expert-annotated ground truth labels. This dataset has been broadly used in the research community for developing and evaluating machine learning networks and computer vision models for skin cancer screening, melanoma detection, and related tasks. Sample images of different classes are shown in Figure 1.

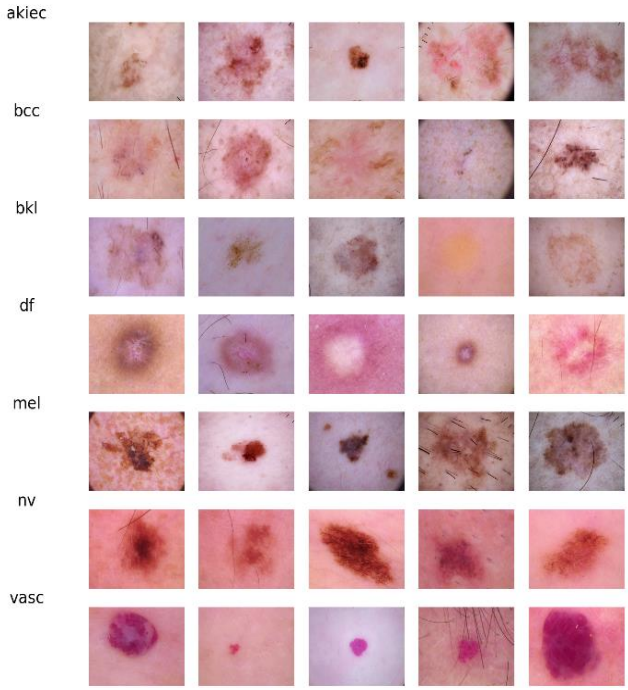

Figure 1. Sample images from the data set for each class

3.2 Data visualisation

The image set has 7 classes of skin cancer images which are distributed in an unbalanced manner. Figure 2 shows the data distribution among the various classes according to cell type. In Table 1, a concise summary of class abbreviations along with the corresponding image counts for each category within the dataset is given.

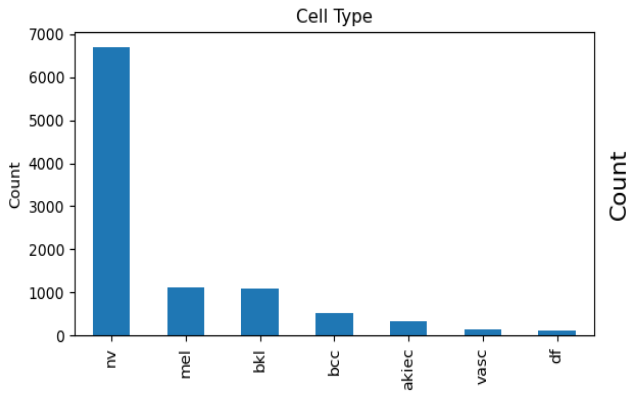

Figure 2. Data distribution before balancing

3.3 Data resizing

HAM10000 data set consists of images of size 600×450. If we try to train the DCNN with these images directly, the system memory will be overloaded and hence we shall reduce the size of the images, but reducing image size may result in loss of information and we might lose some of the crucial attributes and this may lead to reduced accuracy. When the image size is reduced to 32×32 an accuracy of 79.31% is achieved. After various trial and error attempts to achieve an optimal trade-off between accuracy and computational complexity, an optimum image size of 75×75 is chosen which resulted in an accuracy of 96%. The original and resized sample images are shown in Figure 3.

Table 1. Various skin cancer classes and their count

S.No	Class Name/ Class Abreviation	Number of Images
1	NV - (Melanocytic- Nevus)	6705
2	MEL - (Melanocytic- Nevus)	1113
3	BKL - (Benign or Seborrheic- Keratoses Lesions)	1099
4	BCC - (Basal- Cell- Carcinoma)	514
5	AKIEC - (Acnetic- Keratoses and Intraepithelial- Carcinoma)	327
6	VASC - (Vascular- Lesions)	142
7	DF - (Dermatofibroma)	115

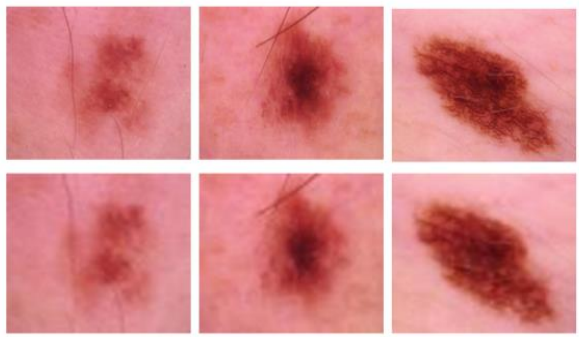

Figure 3. Resized and filtered sample images

3.4 2D Bilateral filtering

The images are subjected to 2D Bilateral filtering before they are applied to the CNN. 2D Bilateral filtering resulted in minimum bilateral filtering is a non-linear image filtering technique used for various image processing tasks, including noise reduction, edge preservation, and image enhancement. It is designed to smooth images while preserving important edges and details. The "bilateral" part of the name stems from the reality that the filtering process considers both spatial proximity and pixel intensity differences. Employing 2D bilateral filtering prior to inputting images into a CNN model confers the advantage of enhancing data quality and robustness. This technique achieves noise reduction and preserves crucial image intricacies, ultimately leading to improved performance. Figure 3 shows the resized and filtered images.

We can represent the input image I as a function of (x, y) that are the spatial coordinates of a pixel. The filtered output is indicated by O as a function of (x, y).

The bilateral filter uses a Gaussian kernel to compute the weights for spatial proximity. The spatial weight is determined by the Euclidean distance between pixels. The Gaussian spatial kernel is defined as:

$$G_{spacial}(d) = \exp\left(-\frac{d^2}{2\sigma_{spacial}^2}\right) \quad (1)$$

Here, d indicates the pixel to pixel Euclidean displacement and $\sigma_{spatial}$ is a parameter that controls the spatial decay of the weights.

Similarly, the bilateral filter uses a Gaussian range kernel to compute the weights based on pixel intensity differences. The range weight is determined by the difference in pixel intensities. The Gaussian range kernel is defined as:

$$G_{range}(r) = \exp\left(-\frac{r^2}{2\sigma_{range}^2}\right) \quad (2)$$

where, r represents the difference in pixel intensities, and σ_{range} is a parameter controlling the range decay of the weights.

The filtered output pixel O(x,y) is figured as the average of the weighted pixel measurements I(u, v) within a defined neighbourhood centered around x,y:

$$O(x,y) = \frac{1}{W}\sum I(u,v).G_{spacial}(||(x,y) - (u,v)||).G_{range}(|I(x,y) - I(u,v)|) \quad (3)$$

Here, W is the normalization factor computed as the sum of the weights over the neighbourhood:

$$W = \sum G_{spacial}(||(x,y) - (u,v)||).G_{range}(|I(x,y) - I(u,v)|) \quad (4)$$

The bilateral filter effectively smooths the image by considering both spatial proximity and pixel intensity differences. The spatial kernel controls the smoothing within the local neighbourhood, while the range kernel preserves edges and details based on intensity differences.

By adjusting the parameters $\sigma_{spatial}$ and σ_{range}, you can control the amount of smoothing and the sensitivity to pixel intensity variations, respectively.

The 2D bilateral filtering resulted in Mean Square Error (MSE) of 3.225 and PSNR of 43.335 dB.

3.5 Data balancing

Attaining balanced data holds paramount importance in CNN classification tasks, particularly when confronted with imbalanced datasets characterized by significant disparities in the sample counts across different classes. Here are some key reasons why data balancing is important in CNN classification problems:

3.5.1 Mitigating bias towards majority classes

Imbalanced datasets can result in skewed model training, causing the model to favour prevalent class due to its higher prevalence.

3.5.2 Improved model performance

Imbalanced datasets can diminish the effectiveness of the CNN model, particularly in accurately predicting the minority classes.

3.5.3 Preventing overfitting

Imbalanced datasets can promote overfitting, resulting in excessive specialization in predicting the majority class and fails to generalize well on unseen data.

3.5.4 Preserving class boundaries

When classes are imbalanced, the decision boundaries learned by the model may not accurately represent the underlying patterns and characteristics of the minority classes.

3.5.5 Fair evaluation and interpretation

Balanced datasets ensure fair evaluation and interpretation of model performance across all classes. To examine the model's proficiency, we need indicators like accuracy, that yields valuable insights, particularly when dealing with a balanced dataset. It allows for a fair comparison of model's proficiency across different categories and aids in identifying areas where the model may need improvement.

Balancing the data in CNN classification problems is crucial for overcoming bias, improving model performance, preventing overfitting, preserving class boundaries, and ensuring fair evaluation. By addressing the class imbalance, the CNN model becomes more reliable, accurate, and capable of effectively classifying samples from all classes.

Table 2. Data after balancing

S.No	Class Name/Class Abreviation	Sample Size-1 (Accuracy 94%)	Sample Size-2 (Accuracy 96%)
1	NV - (Melanocytic-Nevus)	2000	2800
2	MEL - (Melanocytic-Nevus)	2000	2800
3	BKL - (Benign or Seborrheic- Keratoses Lesions)	2000	2800
4	BCC - (Basal- Cell-Carcinoma)	2000	2800
5	AKIEC - (Acnetic-Keratoses and Intraepithelial–Carcinoma)	2000	2800
6	VASC - (Vascular-Lesions)	2000	2800
7	DF - (Dermatofibroma)	2000	2800

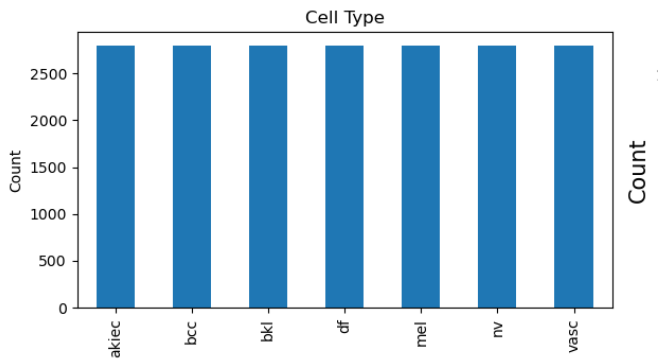

Figure 4. Data distribution after balancing

As we can see from Table 1, there's a lot of imbalances in the dataset. The class NV has highest number of images i.e. 6705 while the class VASC has the lowest number of images i.e. 142. In order to balance the data, the classes whose number

of images are less are oversampled and the classes whose number of images are more are under sampled.

The data balancing can be done in a number of ways. The image classes 1 to 7 where the number of images per class are less than 2800, are oversampled to 2800 while the images of class 0 which are 6705 in number are down sampled to 2800.

The data after balancing is shown in Table 2 and is visualised as shown in the Figure 4.

When the number of images sampled are taken as 500 for each class it resulted in an accuracy of 71.43%. When the sampled images for each class is increased to 1000 images per class the accuracy increased to 79.31%. When the sampled images for each class is increased to 2000 images per class the accuracy increased to 94% and with 2800 images for each class the accuracy further increased to 96%. Given the hardware capability of our system this is the highest sample size that we could take, Thus, the DCNN is trained with 2800 images of each class split into 80:20 ratio that give 2240 images for training and 560 images for testing.

With the application of augmentation on-the-fly we created augmented images while training and this will increase diversity, robustness and avoids overfitting and addresses class imbalance. We used 'ImageDataGenerator' in Keras Tensorflow for augmentation. 'ImageDataGenerator' in TensorFlow does not increase the number of training images in your dataset but dynamically generates augmented versions of the existing images on-the-fly during the training process. This indicates that the count of unique pictures in your dataset remains the same while a new set of augmented images appear for training for each batch.

In Figure 5 the original pics along with the augmented pics produced by 'ImageDataGenerator' are shown.

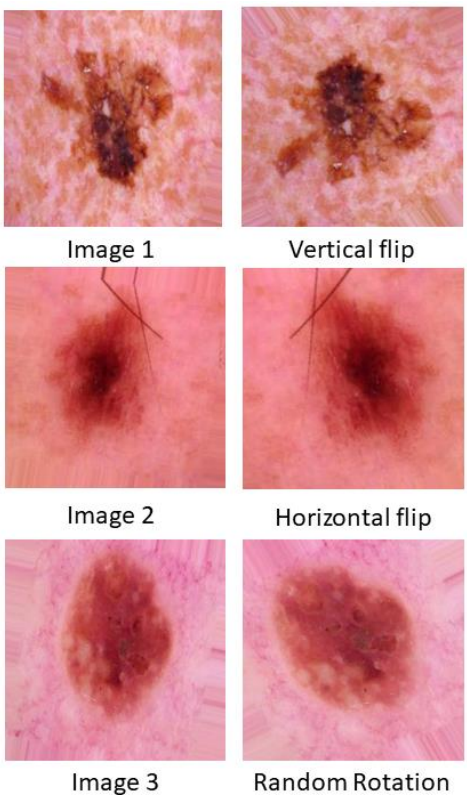

Figure 5. Sample images and their augmented counterparts

3.6 DCNN architecture

The proposed DCNN architecture has four stages, stage 1 has four layers one Convolution layer (Conv2D), one Maxpooling layer (MaxPool), one Batch normalization layer (BN) and one Dropout layer. Stage 2 and 3 also have similar layers. The data is flattened at the end of stage 3 and passed on to stage 4, this stage has a dense layer followed by Softmax layer. Figure 6. shows the architecture of DCNN and Table 3. shows the model summary.

3.6.1 Conv2D layer

This layer provides the Convolution filters. In the first stage a configuration of 256 filters, each possessing a 3x3 kernel and activated by the ReLU (Rectified Linear Unit) function. The first layer processes the raw image data by applying 256 convolutional filters. Each filter learns to detect specific features or patterns in the image, such as edges or textures. By incorporating ReLU non-linearity is introduced into the output, enabling the network to effectively grasp intricate and complex relationships. In second stage we have 128 filters, stage 3 has 64 and stage 4 has 32 filters.

3.6.2 MaxPool layer

Stage 1 to 3 each have a MaxPooling layer with kernel size (2,2). MaxPooling diminishes the resolution of activation maps by extracting the highest value from a local window, typically a 2×2 grid, as it moves across the input data. This helps to down sample the data, lowering the parameter count and introducing some degree of translation invariance.

3.6.3 BN layer (Batch-Normalization layer)

BN Layer is commonly used to scale the weights of each layer, so that the learning phase will be more stable and efficient. It addresses internal covariate shift the problem, which refers to the change in the distribution of layer inputs during learning phase.

Mathematically, batch normalization can be defined as follows:

Given a mini-batch of activations, denoted as x of size (m, n). m stands for the batch size and n for the count of features. The steps for batch normalization are as follows:

Estimate the centre value (mean) and variance for each feature dimension across the mini-batch:

$$mean\ \mu = \frac{1}{m}\sum(x) \tag{5}$$

$$variance\ \sigma^2 = \frac{1}{m}\sum(x-\mu)^2 \tag{6}$$

Normalize the activations using the mean and variance:

$$x' = (x-\mu)/\sqrt{(\sigma^2+\varepsilon)} \tag{7}$$

Here, ε is a small constant (e.g., 10e-8) added for numerical stability to avoid division by zero.

Scale and shift the normalized activations using learned parameters, typically a scaling factor (γ) and a bias term (β):

$$y = \gamma * x' + \beta \tag{8}$$

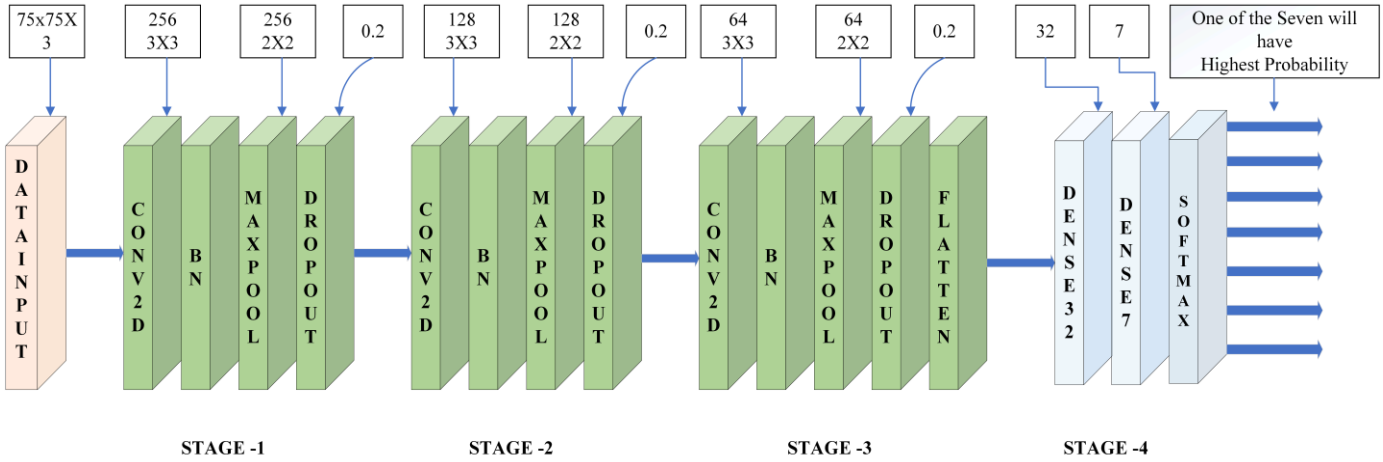

Figure 6. DCNN architecture

Table 3. DCNN model summary

Type of the Layer	Output Dim	Params
conv2d	(None, 73, 73, 256)	7,168
batch_normalization	(None, 73, 73, 256)	1,024
2D-Max-Pool	(None, 36, 36, 256)	0
dropout	(None, 36, 36, 256)	0
conv2d_1	(None, 34, 34, 128)	295,040
batch_normalization_1	(None, 34, 34, 128)	512
2D-Max-Pool-1	(None, 17, 17, 128)	0
dropout_1	(None, 17, 17, 128)	0
conv2d_2	(None, 15, 15, 64)	73,792
batch_normalization_2	(None, 15, 15, 64)	256
2D-Max-Pool-2	(None, 7, 7, 64)	0
dropout_2	(None, 7, 7, 64)	0
Flatt layer	(None, 3136)	0
Dense	(None, 32)	100,384
Dense_1	(None, 7)	231

The parameters γ and β are learnable parameters and are applied to each feature dimension individually.

During training, batch normalization also introduces additional parameters to the network, γ and β, which are learned during the optimization process. These parameters allow the network to undo the normalization if it is beneficial for the task at hand.

Batch normalization helps in addressing the internal covariate shift problem and provides several benefits, including faster convergence, reduced sensitivity to the network's initialization, and regularization effects, which can improve generalization performance.

It's worth noting that during inference or prediction, the mean and variance are typically estimated using the entire training set rather than the mini-batch statistics. This is because batch normalization is applied to individual samples during training, but during inference, we usually want the model to make predictions on individual samples without the need for mini-batches.

3.6.4 Dropout layer

Dropout stands as a regularization technique that introduces an element of randomness by selectively nullifying a portion of input units during each training iteration. It supports averting excessive fitting by alleviating the dependency on any individual unit, forcing the network to learn more fortified features. Dropout rates of 0.4, 0.3 and 0.2 are tested in this model and the dropout rate of 0.2 is selected as this produced the best results.

3.6.5 Flatt layers

The 2-D traits are mapped into a 1-D vector, by the flat layers, prepping the data for the subsequent fully connected layers.

3.6.6 Dense and SoftMax layers

The first dense layer produces 32 outputs which are reduced to 7 by the second dense layer and SoftMax activation function produces the seven probabilities for each class.

3.7 Hyper parameter tuning

While designing a Convolutional Neural Network we come across several hyper parameters. The most important of them are Optimizer, Learning rate and Dropout Rate. Improper tuning of these parameters may result in poor accuracy, excessive run time and even the model may not converge.

3.7.1 Optimizer

Various popular optimizers available are Adam, SGD, Adagrad.

Stochastic Gradient Descent (SGD): SGD is a distinct foundational optimizer used in computational tasks and deep learning to train models by minimizing a given loss function. The algorithm is a version of the gradient descent and is notably suitable for bigger datasets.

SGD begins by initializing the model's parameters randomly or using some predefined values.

Unlike traditional gradient descent, where the entire dataset is utilized to estimate the gradient of the loss function in each iteration, SGD operates on smaller sub groups of the data called mini-batches. In every training loop, a mini-batch of data points is randomly sampled from the entire dataset. This randomness helps introduce stochasticity into the optimization process, which can be beneficial for escaping local minima and speeding up convergence.

For each mini-batch the loss function's gradient concerning the model's parameters is estimated. This gradient represents the direction in which the parameters ought to be tuned to mitigate the loss. The parameters are then updated using the negative gradient direction and a learning rate.

SGD's main advantage lies in its efficiency with large datasets since it only requires a subset of the data in each iteration. This can lead to faster training times compared to using the entire dataset. However, SGD can exhibit noisy updates due to the random mini-batch sampling, which can

cause the optimization process to be less stable and take longer to converge.

Adaptive Gradient Algorithm (Adagrad): Adagrad is another optimizer used often in computational tasks. It is designed to automatically modulate the rate of learning for every network parameter depending on the information about the history of the gradient. Adagrad is particularly best-suited for handling sparse data and scenarios where the features have significantly different scales.

Adagrad initializes a vector 'G' of the same dimension as the model's parameters. This vector keeps track of the sum of squared gradients for each parameter. In each training iteration, Adagrad calculates the derivative of the cost function concerning the model's parameters.

The key feature of Adagrad is the adaptive learning rate. As the optimization progresses, the vector 'G' accumulates the squared gradients for each parameter. Parameters associated with frequent or large gradients will have smaller effective learning rates, as their corresponding elements in 'G' grow. Conversely, parameters associated with infrequent or small gradients will have larger effective learning rates.

While Adagrad's adaptive learning rates can be advantageous, they can also lead to diminishing learning rates over time, making it difficult for the optimization process to escape from valleys with flat gradients. This issue is especially pronounced in deep networks and can slow down convergence.

Adaptive Moment Estimation (Adam): Adam optimizer is a popular choice for optimization used in machine learning and deep learning to train models by minimizing a given loss function. It amalgamates the positive aspects of two techniques: Dynamic learning rates from RMSProp and momentum-influenced updates. Adam is designed to provide efficient and effective optimization across a wide range of tasks.

Adam maintains two sets of exponentially decaying moving averages:

- m: The first moment estimate, which is a running average of gradients.
- v: The second moment estimate, which is a running average of squared gradients.

Both m and v are initialized to zero vectors of the same dimension as the model's hyperparameters.

With each loop of training, Adam calculates the slope of the loss function concerning the model's parameters. It then updates the moving averages. The moving averages m and v are initialized with zeros and can be biased towards zero, especially during the initial iterations. To correct this bias, Adam performs bias correction. Finally, Adam uses the bias-corrected moment estimates to update the model's parameters.

Adam's combination of adaptive learning rates and momentum from the moving averages allows it to perform well on a wide range of optimization problems.

3.7.2 Learning rate

The learning rate stands as a hyper-parameter governing the rate at which the model's parameters receive updates in the optimization process. It determines the pace at which the model adjusts to the training data.

When training a CNN, the goal is to reduce the loss function that quantifies how well the model's predictions match the actual target values. The learning rate plays a crucial role in this process.

Large Learning Rate: Using a large learning rate can result in rapid updates to the model's parameters. While this might help the model converge quickly, it could also lead to overshooting the optimal solution, causing the optimization process to oscillate or even diverge.

Small Learning Rate: A small learning rate will result in slower parameter updates. This might be helpful for fine-tuning when the optimization is close to the smallest value of the loss function. However, using an excessively small learning rate can make the optimization process very slow and could get stuck in local minima.

Picking an apt learning rate helps in striking a balance amid fast convergence and stable optimization.

3.7.3 Dropout rate

Dropout functions as a regularization strategy commonly used in CNNs and other types of neural networks to prevent overfitting. Overfitting materializes when a model exhibits strong performance during the learning phase while faltering to generalize effectively to novel data. Dropout helps address this issue by reducing the interdependencies between neurons and stimulating the network towards enhanced learning of robust and generalizable features.

Dropout involves "dropping out" a random subset of neurons (both hidden and input) in each training iteration. This means that these neurons are temporarily removed from the network architecture during that iteration. The probability of a neuron being dropped out is dictated by a parameter referred to as the dropout rate.

During inference (when the trained model is used to make predictions), all neurons are active, but their outputs are scaled down by the dropout rate. This scaling ensures that the expected output of each neuron during inference remains close to its average behaviour during training.

- Regularization: By randomly dropping neurons, dropout keeps the network from being overly reliant on any single neuron. This encourages the network to distribute its learning across multiple neurons and, subsequently, learn more robust features.
- Reduced Overfitting: Dropout reduces the risk of overfitting by introducing noise during training. This compels the network to acquire a broader, more universal understanding of the data.
- Ensemble Effect: During training, dropout creates many different network architectures by randomly dropping neurons.
- When making predictions, these different architectures are combined, acting as an ensemble.
- This ensemble effect can lead to improved generalization.

Dropout is a powerful technique, but it requires careful tuning of the dropout rate, especially for different layers in the network. It's common to start with a moderate dropout rate (e.g., 0.2 to 0.5) and adjust it through experimentation based on the model's efficacy on a validation set. It's also crucial to note that dropout is generally not applied during inference, as the goal during inference is to make accurate predictions without introducing randomness.

We adopted an iterative method to tune the hyper parameters, the code snippet used to fine tune the hyper parameters is given below.

Hyperparameter search space

```
dropout_values =[0.2, 0.3, 0.4]
optimizer_values =['adam', 'adagrad', 'sgd']
learning_rate_values =[0.1, 0.01]

for dropout_rate in dropout_values:
  for optimizer in optimizer_values:
    for learning_rate in learning_rate_values:
      model = Sequential ()
          #add Conv2D with 256 filters and size (3, 3),
         input_shape=(SIZE, SIZE, 3)))
      # add BN Layer
      # add MaxPool 2D layer of size (2, 2)
      # add dropout layer

      # add Conv2D with 128 filters and size (3, 3)
      # add BN Layer
      # add MaxPool 2D layer of size (2, 2)
      # add dropout layer

      # add Conv2D with 64 filters and size (3, 3)
      # add BN Layer
      # add MaxPool 2D layer of size (2, 2)
      # add dropout layer
      # add Flat layer

      #add Dense layer with 32 filters, activation='relu'
          #add Dense layer with filters=num_classes and
           activation='softmax'))
      # Choose the optimizer based on the hyperparameter
      #Write code for choosing one of the three optimizers
           using a if-else loop
       raise ValueError("Invalid optimizer")
      #Compile the model
          # Fit the model
```

Table 3 displays the metrics with different combinations of optimizer, learning rate and dropout rate.

The results of hyper parameter tuning shown in Table 4. tells that the best combination is to use Adam as an optimizer with Learning Rate of 0.1 and Dropout Rate of 0.2.In our proposed method while tuning the hyper parameters we observed that Adam resulted in the best performance for the given data set.

Table 4. Impact of hyper parameter tuning on various metrics

Optimizer	Learning_Rate	Dropout_Rate	Accuracy (%)	Precision	Recall	F1-Score	Specificity	Sensitivity
Adagrad	0.1	0.4	78	0.78	0.78	0.79	0.96	0.78
	0.1	0.2	88	0.88	0.88	0.88	0.98	0.88
	0.01	0.4	77	0.77	0.76	0.76	0.95	0.76
	0.01	0.2	88	0.89	0.88	0.88	0.98	0.88
SGD	0.1	0.4	80	0.84	0.80	0.80	0.96	0.80
	0.1	0.2	93	0.93	0.93	0.93	0.99	0.93
	0.01	0.4	89	0.89	0.89	0.89	0.98	0.89
	0.01	0.2	95	0.95	0.95	0.95	0.99	0.95
Adam	0.1	0.4	91	0.92	0.91	0.91	0.98	0.91
	0.1	**0.2**	**96**	**0.96**	**0.96**	**0.96**	**0.99**	**0.96**
	0.01	0.4	94	0.94	0.94	0.94	0.99	0.94
	0.01	0.2	79	0.85	0.80	0.80	0.96	0.80

4. RESULTS AND DISCUSSION

With the above architecture the DCNN achieved an Accuracy of 96%. A confusion matrix is a tabular representation that numerically represents the performance of a classification model by displaying the counts of T_{Pos}, T_{Neg},. F_{Poss}, F_{Neg} predictions. It's a useful tool for assessing the quality of predictions made by a model and understanding where it might be making errors.

The Confusion Matrix gives the information about the following

- T_{Pos}: These are instances where your model accurately identified the positive class (the desired condition), and the actual classification was indeed positive.
- T_{Neg}: These are instances where your model correctly recognized the negative class, and the actual classification was indeed negative.
- F_{Poss}: These are instances where your model predicted the positive class, but the actual classification was negative. This is also referred to as a Type I error.
- F_{Neg}: These are instances where your model predicted the negative class, but the actual classification was positive. This is also referred to as a Type II error.

Various performance metrics that can be used to estimate the predictive capabilities of DCNN are given below:

$$Accuracy = \frac{T_{Pos} + T_{Neg}}{T_{Pos} + T_{Neg} + F_{Pos} + F_{Neg}} \quad (9)$$

$$Precision = \frac{T_{Pos}}{T_{Pos} + F_{Pos}} \quad (10)$$

$$Recall = \frac{T_{Pos}}{T_{Pos} + F_{Neg}} \quad (11)$$

$$F1 - score = \frac{2 * T_{Pos}}{2 * T_{Pos} + T_{Neg} + F_{Pos} + F_{Neg}} \quad (12)$$

Figure 7 shows the confusion matrix while Figure 8 shows the classification reports. Figure 9 shows the ROC curve. In Table 4, the performance metrics for various combinations of learning rate, dropout and optimizer are given. By observing Table 4, we understand the impact of hyperparameter tuning and it turns out that Adam leads to the best results with a learning rate of 0.1 and a dropout rate of 0.2. The second-best result is given when learning rate is 0.01, dropout 0.2 and SGD is the optimizer. Table 5 gives the impact of train-test split on

the performance of the model. We observe that a split of 80:20 gave the best results.

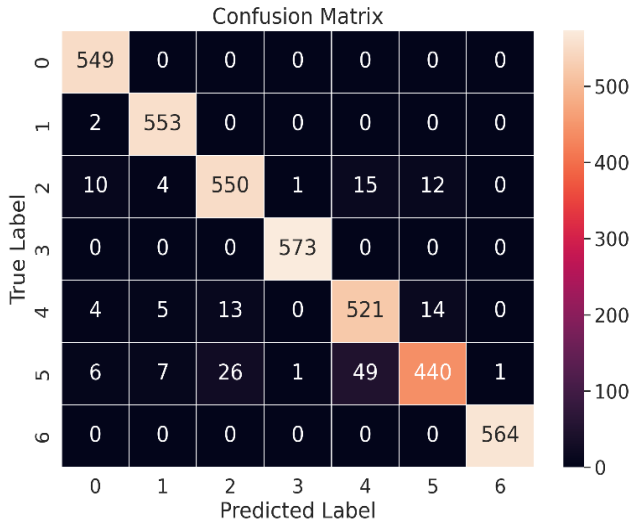

Figure 7. Confusion matrix of DCNN

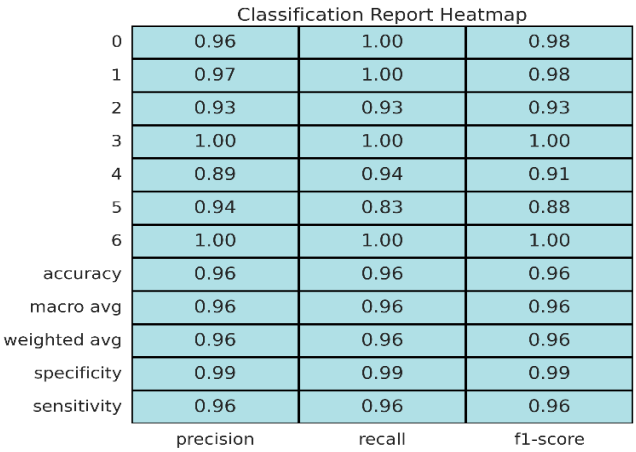

Figure 8. Classification report of DCNN

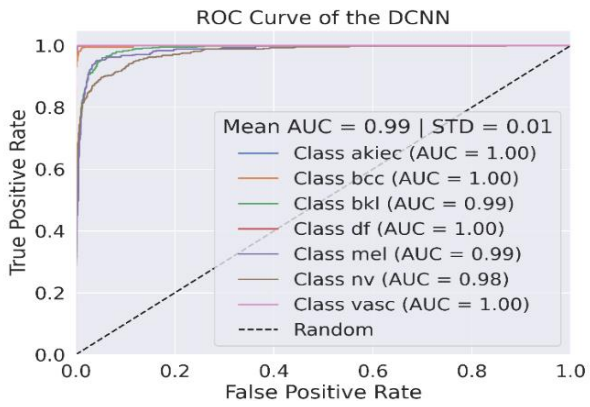

Figure 9. ROC curve of DCNN

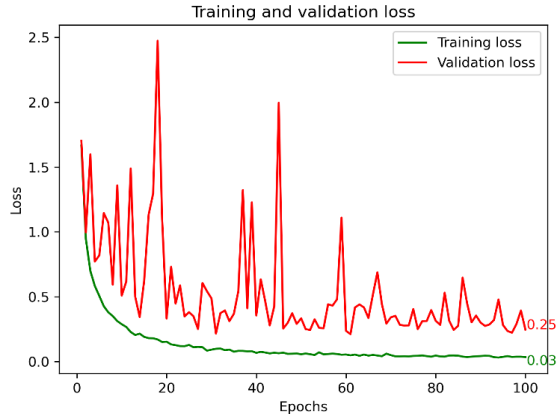

Figure 10. Training vs testing loss

Figure 11. Training vs testing accuracy

A classification report is a concise summary of various performance metrics for a classification model. It provides valuable insights into how well the model performs across different classes and gives a more detailed evaluation than just looking at accuracy. Figure 8 shows the classification report of the DCNN for the case with learning rate 0.1, dropout 0.2 with Adam optimizer. The classification report generally provides benchmarks like precision, recall, F1-score, and the number of instances for each class.

The ROC curve (Receiver Operating Characteristic curve) is shown in Figure 9. The mean Area Under the Curve (AUC) is 0.99 with a Standard Deviation (STD) of 0.01. the legend provided in the figure shows the AUC for individual classes. The loss and accuracy curves for training phase and testing phase can be seen in Figure 10 and Figure 11 respectively.

Table 6 shows the comparative results of our proposed DCNN with some of the existing state of the art works. We can see that our DCNN did extremely well and out performed many of the published works.

Table 7 shows the models performance with respect to the other prominent data sets like PH^2 [30], ISIC 2016 [31] and ISIC 2017 [32]. The findings in Table 7 suggests that the model is robust and consistent.

Table 5. Impact of train-test split ratio on various metrics

S.No.	Train-Test Split Ratio	Accuracy (%)	Precision	Recall	F1-Score	Specificity	Sensitivity
1	90:10	94	0.94	0.94	0.94	0.99	0.94
2	**80:20**	**96**	**0.96**	**0.96**	**0.96**	**0.99**	**0.96**
3	70:30	95	0.95	0.95	0.95	0.99	0.95
4	60:40	93	0.93	0.93	0.93	0.99	0.93
5	50:50	92	0.92	0.92	0.92	0.99	0.92

Table 6. Accuracy, specificity, sensitivity comparison with the published models that used HAM10000 data set

S.No	Existing Model	Accuracy	Specificity	Sensitivity
1	Gong et al. [2]	92.6%	0.977	0.483
2	Zhou et al. [3]	87.7%	0.847	0.903
3	Son et al. [4]	87.1%	0.919	0.873
4	Thurnhofer-Hemsi et al. [6]	83.5%	0.954	0.656
5	Hoang et al. [8]	86.33%	0.977	0.863
6	Shen et al. [23]	85.3%	0.973	0.789
7	**Our work**	**96%**	**0.96**	**0.99**

Table 7. Performance of DCNN on various data sets

S.No.	Dataset	Accuracy (%)	Specificity	Sensitivity
1	PH^2 200 images	97.2	0.99	0.968
2	ISIC 2016 900 images	96.7	0.99	0.96
3	ISIC 2017	96.5	0.99	0.965
4	HAM10000 10015 images	96	0.99	0.96

5. CONCLUSION AND FUTURE WORK

The findings demonstrate that the suggested DCNN model has delivered encouraging outcomes, boasting an accuracy rate of 96%, a mean precision of 0.96, a mean recall of 0.96, and a mean F1-score of 0.96, specificity of 0.96 and sensitivity of 0.99. Moreover, the model has surpassed the performance of numerous established models.

The model further needs to be tested on big datasets like ISIC 2019 (25,331 images) and ISIC 2020 (33,126 images) which are very huge. There is a scope to increase the accuracy and other performance metrics by using ensemble method where we can integrate our DCNN with the well-established classification models like DenseNet, INCEPTION-v3.

REFERENCES

[1] Gururaj, H.L., Manju, N., Nagarjun, A., Aradhya, V.M., Flammini, F. (2023). DeepSkin: A deep learning approach for skin cancer classification. IEEE Access, 11: 50205-50214. https://doi.org/10.1109/ACCESS.2023.3274848

[2] Gong, A., Yao, X., Lin, W. (2020). Classification for dermoscopy images using convolutional neural networks based on the ensemble of individual advantage and group decision. IEEE Access, 8: 155337-155351. https://doi.org/10.1109/ACCESS.2020.3019210

[3] Zhou, Q., Shi, Y., Xu, Z., Qu, R., Xu, G. (2020). Classifying melanoma skin lesions using convolutional spiking neural networks with unsupervised STDP learning rule. IEEE Access, 8: 101309-101319. https://doi.org/10.1109/ACCESS.2020.2998098

[4] Son, H.M., Jeon, W., Kim, J., Heo, C.Y., Yoon, H.J., Park, J.U., Chung, T.M. (2021). AI-based localization and classification of skin disease with erythema. Scientific Reports, 11(1): 5350. https://doi.org/10.1038/s41598-021-84593-z

[5] Zhang, J., Xie, Y., Xia, Y., Shen, C. (2019). Attention residual learning for skin lesion classification. IEEE Transactions on Medical Imaging, 38(9): 2092-2103. https://doi.org/10.1109/TMI.2019.2893944

[6] Thurnhofer-Hemsi, K., López-Rubio, E., Dominguez, E., Elizondo, D.A. (2021). Skin lesion classification by ensembles of deep convolutional networks and regularly spaced shifting. IEEE Access, 9: 112193-112205. https://doi.org/10.1109/ACCESS.2021.3103410

[7] Ahmad, B., Usama, M., Huang, C.M., Hwang, K., Hossain, M.S., Muhammad, G. (2020). Discriminative feature learning for skin disease classification using deep convolutional neural network. IEEE Access, 8: 39025-39033. https://doi.org/10.1109/ACCESS.2020.2975198

[8] Hoang, L., Lee, S.H., Lee, E.J., Kwon, K.R. (2022). Multiclass skin lesion classification using a novel lightweight deep learning framework for smart healthcare. Applied Sciences, 12(5): 2677. https://doi.org/10.3390/app12052677

[9] 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(1): 1-9. https://doi.org/10.1038/sdata.2018.161

[10] Yu, L., Chen, H., Dou, Q., Qin, J., Heng, P.A. (2016). Automated melanoma recognition in dermoscopy images via very deep residual networks. IEEE Transactions on Medical Imaging, 36(4): 994-1004. https://doi.org/10.1109/TMI.2016.2642839

[11] Harangi, B. (2018). Skin lesion classification with ensembles of deep convolutional neural networks. Journal of Biomedical Informatics, 86: 25-32. https://doi.org/10.1016/j.jbi.2018.08.006

[12] Demyanov, S., Chakravorty, R., Abedini, M., Halpern, A., Garnavi, R. (2016). Classification of dermoscopy patterns using deep convolutional neural networks. In 2016 IEEE 13th International Symposium on Biomedical Imaging (ISBI), Prague, Czech Republic, pp. 364-368. https://doi.org/10.1109/ISBI.2016.7493284

[13] Masood, A., Al-Jumaily, A., Anam, K. (2015). Self-supervised learning model for skin cancer diagnosis. In 2015 7th International IEEE/EMBS Conference on Neural Engineering (NER), Montpellier, France, pp. 1012-1015. https://doi.org/10.1109/NER.2015.7146798

[14] Mahbod, A., Schaefer, G., Wang, C., Ecker, R., Ellinge, I. (2019). Skin lesion classification using hybrid deep neural networks. In ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Brighton, UK, pp. 1229-1233. https://doi.org/10.1109/ICASSP.2019.8683352

[15] Pomponiu, V., Nejati, H., Cheung, N.M. (2016). Deepmole: Deep neural networks for skin mole lesion

classification. In 2016 IEEE International Conference on Image Processing (ICIP), Phoenix, AZ, USA, pp. 2623-2627. https://doi.org/10.1109/ICIP.2016.7532834

[16] Albahar, M. A. (2019). Skin lesion classification using convolutional neural network with novel regularizer. IEEE Access, 7: 38306-38313. https://doi.org/10.1109/ACCESS.2019.2906241

[17] Li, Y., Shen, L. (2018). Skin lesion analysis towards melanoma detection using deep learning network. Sensors, 18(2): 556. https://doi.org/10.3390/s18020556

[18] Javaid, A., Sadiq, M., Akram, F. (2021). Skin cancer classification using image processing and machine learning. In 2021 International Bhurban Conference on Applied Sciences and Technologies (IBCAST), Islamabad, Pakistan, pp. 439-444. https://doi.org/10.1109/IBCAST51254.2021.9393198

[19] Ichim, L., Popescu, D. (2020). Melanoma detection using an objective system based on multiple connected neural networks. IEEE Access, 8: 179189-179202. https://doi.org/10.1109/ACCESS.2020.3028248

[20] Ali, M.S., Miah, M.S., Haque, J., Rahman, M.M., Islam, M.K. (2021). An enhanced technique of skin cancer classification using deep convolutional neural network with transfer learning models. Machine Learning with Applications, 5: 100036. https://doi.org/10.1016/j.mlwa.2021.100036

[21] Ali, K., Shaikh, Z.A., Khan, A.A., Laghari, A.A. (2022). Multiclass skin cancer classification using EfficientNets-a first step towards preventing skin cancer. Neuroscience Informatics, 2(4): 100034. https://doi.org/10.1016/j.neuri.2021.100034

[22] Datta, S.K., Shaikh, M.A., Srihari, S.N., Gao, M. (2021). Soft attention improves skin cancer classification performance. In Interpretability of Machine Intelligence in Medical Image Computing, and Topological Data Analysis and Its Applications for Medical Data: 4th International Workshop, iMIMIC 2021, and 1st International Workshop, TDA4MedicalData 2021, Held in Conjunction with MICCAI 2021, Strasbourg, France, Proceedings. Springer International Publishing. Springer, Cham, 4: 13-23. https://doi.org/10.1007/978-3-030-87444-5_2

[23] Shen, S., Xu, M., Zhang, F., Shao, P., Liu, H., Xu, L., Zhang, C., Liu, P., Yao, P., Xu, R.X. (2022). A low-cost high-performance data augmentation for deep learning-based skin lesion classification. BME Frontiers, 2022. https://doi.org/10.34133/2022/9765307

[24] Calderón, C., Sanchez, K., Castillo, S., Arguello, H. (2021). BILSK: A bilinear convolutional neural network approach for skin lesion classification. Computer Methods and Programs in Biomedicine Update, 1: 100036. https://doi.org/10.1016/j.cmpbup.2021.100036

[25] Qian, S., Ren, K., Zhang, W., Ning, H. (2022). Skin lesion classification using CNNs with grouping of multi-scale attention and class-specific loss weighting. Computer Methods and Programs in Biomedicine, 226: 107166. https://doi.org/10.1016/j.cmpb.2022.107166

[26] Murthy, S.V.S.N., Prasad, P.M.K. (2023). Adversarial transformer network for classification of lung cancer disease from CT scan images. Biomedical Signal Processing and Control, 86: 105327. https://doi.org/10.1016/j.bspc.2023.105327

[27] Subbarao, M.V., Ram, G.C., Kumar, D.G., Terlapu, S.K. (2022). Brain tumor classification using ensemble classifiers. In 2022 International Conference on Electronics and Renewable Systems (ICEARS), Tuticorin, India, pp. 875-878. https://doi.org/10.1109/ICEARS53579.2022.9752177

[28] Varma, A.K.C., Dileep, M., Prudhvi Raj, B., Prasanna Kumar, G. (2023). Impulse noise removal using residual convolutional neural networks. In AIP Conference Proceedings. AIP Publishing, 290(1). https://doi.org/10.1063/5.0178635

[29] Shaik, A.R., Chandra, K.R., Raju, B.E., Budumuru, P.R. (2021). Glaucoma identification based on segmentation and fusion techniques. In 2021 International Conference on Advances in Computing, Communication, and Control (ICAC3), Mumbai, India, pp. 1-4. https://doi.org/10.1109/ICAC353642.2021.9697174

[30] Mendonça, T., Ferreira, P.M., Marques, J.S., Marcal, A.R., Rozeira, J. (2013). PH 2-A dermoscopic image database for research and benchmarking. In 2013 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC). IEEE. Osaka, Japan, pp. 5437-5440. https://doi.org/10.1109/EMBC.2013.6610779

[31] Gutman, D., Codella, N.C., Celebi, E., Helba, B., Marchetti, M., Mishra, N., Halpern, A. (2016). Skin lesion analysis toward melanoma detection: A challenge at the international symposium on biomedical imaging (ISBI) 2016, hosted by the international skin imaging collaboration (ISIC). arXiv Preprint arXiv: 1605.01397. https://doi.org/10.48550/arXiv.1605.01397

[32] Codella, N.C., Gutman, D., Celebi, M.E., Helba, B., Marchetti, M.A., Dusza, S.W., Kalloo, A., Liopyris, K., Mishra, N., Kittler, H., Halpern, A. (2018). Skin lesion analysis toward melanoma detection: A challenge at the 2017 international symposium on biomedical imaging (ISBI), hosted by the international skin imaging collaboration (ISIC). In 2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018), Washington, DC, USA, pp. 168-172. https://doi.org/10.1109/ISBI.2018.8363547