Erythemato-Squamous Disease Detection using Best Optimized Estimators of ANN

Rajashekar Deva[1], Dr G .Narsimha[2]
Asst.Prof. Department of C.S.E., Methodist College of Engineering &Technology, Abids, Hyderabad, Telangana, 500001, India[1]
Principal and Professor, JNTUH, Sulthanpur, Sangareddy, Telangana, India[2]

***Abstract*—Medical area focused on automating skin cancer detection after the pandemic era of "Monkey Pox". Previous works proposed ANN mechanisms to classify the type of skin cancer. However, all those models implement layers of ANN with standard estimator components like hidden layers implemented using the ReLu activation function, several neurons are generally a power of two and others, but these values are not always perfect. Few researchers implemented optimization techniques for tuning the estimators of A.I. algorithms, but all those mechanisms require more resources and don't guarantee the best values for each estimator. The proposed method analyzes all the essential estimators of every possible neural network layer. Then it applies a modified version of Bayesian optimization because it avoids the disadvantages of Grid and Random optimization techniques. It picks the best estimator by using the conditional probability of naive Bayesian for every combination.**

***Keywords*—Conditional probability; naive Bayesian; bayesian optimization; grid search; optimization techniques; estimators**

I. Introduction

Most researchers utilize machine learning to identify skin cancers but suffer from overfitting and more resource consumption. The problem can be solved using artificial neural networks, but traditional networks become complicated for smaller datasets. So, the proposed research aims to customize the parameters of the network by performing "Hyper Parameterization".

Hyperparameter tuning is required for behavioural control of the machine-learning model [11]. Hyperparameters can be configured differently for each machine-learning model. Our expected model parameters will yield less-than-ideal outcomes if the hyperparameters are not appropriately tweaked to optimize the loss function [12]. This implies that our model has further problems. When building a model from a particular dataset, hyper-tuning identifies the potential optimal sets of hyperparameters. A single training task executes several trials for hyperparameter tweaking. The task of hyperparameter tweaking involves meta-optimization [27]. The hyperparameter tuner produces the hyperparameter setting, those results in the best-performing model after assessing various hyperparameter settings. By employing the algorithm and the defined limits of hyperparameters, hyperparameter tuning runs multiple training sessions on your dataset to determine which model version is the best.

A. *Hyper Parameter Tuning*

The accuracy of models for machine learning can be significantly increased by using hyperparameter adjustment. Hyperparameter tuning is identifying a collection of appropriate hyperparameter variables for a prediction model and applying this adjusted algorithm to any piece of data. The model's efficiency is maximized by employing that collection of hyperparameters, which minimizes a preset loss function and produces better results with fewer errors. To use an Exhaustive Grid Search in Scikit Learn is a well-known and conventional method for hyperparameter tweaking [13]. Every permutation of each collection of hyperparameters is tested using this procedure. This approach allows us to locate the ideal set of values within the variable search space. Since this approach must test every permutation in the grid size, it typically consumes more computer resources and requires a long time to run. Multiplication of all the variables will determine the size of the parameter grid. Each time a random collection of hyperparameters is tested, the model's performance is recorded. After multiple repetitions, it returns to the mixture with the most significant outcome.

B. *Types of Hyper Tuning*

Irrespective of the type of machine and deep learning models, these tuning algorithms help achieve the minimum error rate with minimum learning rate.

1) Grid search: Grid search optimization takes much time to compute every possible combination of estimators. Suppose we have two parameters for designing the ANN in which parameter-1 can be estimated in X ways, and parameter-2 can be estimated in Y ways. The system needs X into Y ways to apply grid search for this type of neural network. So this can be claimed as an exhaustive search.

2) Random search: Random search only checks some possible estimated values [14]. It randomly selects a few values from each parameter, so this will reduce the number of ways than grid search since random search doesn't cover all possible combinations. So this tuning process only guarantees the best values.

3) Successive halving: The S.H.A. (Successive Halving Algorithm) algorithm can optimize hyperparameters and solve multi-armed bandits challenges. The algorithm's primary goal is to correctly determine the best arm within a strict budget, a constrained time or resource [15]. The algorithm consistently evaluates each arrangement. The weakest performers are removed at the conclusion of each round. The procedure repeats itself until only one configuration is left, with the remaining configurations being examined twice as much as in the previous round. S.H.A. is effective because it uses minimal resources and eliminates data at each level.

Whether or not to look for several configurations in S.H. is still being determined. Some desirable configurations that may initially converge slowly will be eliminated early if n is large. This makes identifying the best allocation approach in the indefinite time frame possible. S.H. requires dynamic updating. Before using the S.H. technique, the hyper-parameters must be manually set. Due to its iterative nature, Successive Halving works well enough on huge datasets. The wide range of cross-validation folds could also be used as the budget for successive Halving. A sufficient budget can prevent good configurations from being terminated too soon. In contrast, an excessive budget can cause subpar configurations to continue for an extended period and waste resources.

4) Bayesian optimization: When deciding which set of hyperparameters to examine next, Bayesian optimization takes into consideration previous assessments. It allows itself to concentrate on those regions of the dimensional space that it considers will provide the most hopeful validation scores by selecting its value, also known intelligently. This method often needs fewer iterations to reach the ideal set of hyperparameter values [16]. Most significantly, it ignores those regions of the dimensional space that it thinks won't contribute anything. As a result, only settings predicted to produce a better validation score are sent through for evaluation, reducing the number of repetitions a model must be trained for validation. It is especially helpful when these assessments are at high costs, when derivatives are absent, or the matter in question is non-convex.

The paper is divided into five sections; the introduction discusses the need and types of hyper optimization techniques. The literature survey section discusses the merits and demerits of the existing approaches. The proposed methodology discusses the customization of the neural network layers with the help of enhanced Bayesian optimization. The results and discussion section elaborates on the metrics obtained by the proposed and compares them with the existing ones. The conclusion section discusses the proof of validity by measuring loss and accuracy.

II. Literature Survey

In [1], MehwishDildar et al. offered a thorough analysis of deep learning methods for early skin cancer detection. The paper concentrated on traditional methods for skin cancer diagnosis, such as ANN, K.N.N., CNN, and GAN. The writers created multiple stages of selection standards. This work aims to evaluate existing models and develop the best N.N. technique for skin cancer detection. The data was filtered using an automated search engine that was developed. Neural networks have been taught to categorize photos and differentiate between photographs of various skin disorders.

Additionally compared were SVM, B.P.N., and three-layer N.N. Due to a shortage of different data, artificial neural networks are trained for skin lesions using tiny data sets. According to the authors, the auto-organization approach, which is still under investigation, may enhance image processing accuracy in the future, particularly in the medical industry.

In [2], Mariam Nawaz et al. developed a deep learning-based segmentation solution that is completely automated. The model uses fuzzy K-means clustering and region-based convolutional neural networks (RCNN). The model applies RCNN using preprocessed data. Using precise localization, faster RCNN could detect skin lesions with accuracy and precision. Various than melanoma, the approach can be applied to other skin conditions. F.K.M. clustering separates the impacted portion of photos from the discovered results. A deep learning framework is faster than RCNN. Faster RCNN relies on the input generic object suggestions, which employ hand-coded models like Edge Box, Selective search, etc. Three datasets are used to apply the model. The proposed model has a chance of being overfitted. This model requires less computing. The F.K.M. model is useful in resolving to overfit.

In [3], Ulzii-OrshikhDorj et al. concentrated on classifying skin cancer using deep CNN and ECOC SVM. The feature extraction method makes use of trained AlexNet. ECOC SVM is employed for classification, along with three fully linked layers. Some photographs in the collection are clear, but others are not because they were pulled from the internet. The work can be expanded by adding the ABCD rule—asymmetry, border, colour, and diameter—for each cancer report. There are four different types of cancer in total. To cut down on noise, the photos are cropped. The pooling layer reduced the size of the input neuron in the CNN. The model's primary goal is quick categorization. Using a deep CNN model, using RGB images allows for detecting actinic keratoses, basal cell carcinoma, melanoma, and squamous cell carcinoma [26].

In [4], Mohammad Ali Kadampur and Sulaiman Al Riyale combined and suggested a model-driven cloud architecture. This is used to build models that help in skin cancer prediction and is based on deep learning algorithms. The metric area beneath the curve for the deep learning algorithm was 99%. Despite the model's lack of programming components, the machine can deal with issues including slowness, accuracy, and a shortage of dermatologists. The suggested model can categorize cell pictures and spot skin cancer. The model is not integrated into the REST API. According to the article, there are now more design options for deep learning classifiers regarding general methods and looping patterns. The paper described the D.L.S. tool's features and showed how to build a deep learning model.

In [5], Andre Esteva et al. used a single CNN-trained end-to-end using images with simple pixels and illness labels as input to demonstrate classification. A computer technique is created that could help doctors and patients keep track of skin blemishes and spot cancer early. Dermatology is built to be automated. No handcrafted components are necessary for the system. By putting the model to the test, the biopsy-proven images are verified. The writers designed the CNN to mimic dermatologists' performance. By putting the biopsy photos to the test, the model is verified. The authors explored the internal properties that CNN learned using T-SNE. Mobiles can be used to deploy this strategy. Several virtual circumstances can be classified using this method, provided there are enough training instances. Saliency maps are created to see the pixels a network focuses on for prediction.

In [6], Ravi Manne et al. summarised numerous studies on utilizing CNNs to categorize skin lesions. CNN has demonstrated incredible image processing power. The research covered a wide range of adversarial approaches in clinical contexts, including colour balance adjustments and input image rotation and translation that can result in incorrect categorization. The authors also noted the variables influencing the findings. Instead of using all the positive data, which would skew the system and produce only good outcomes, the factors are mentioned as vulnerabilities to adversarial assaults. The inks (blue) in dermoscopic images also hurt CNN classification, suggesting that the model may give accurate results if negative input is included. The authors discovered that improperly categorized photos might produce inaccurate results.

In [7], Khalid M. Hosny et al. introduced a system that automatically classifies skin marks. A deep learning network which is pre-trained serves as the model. AlexNet is trained via data segmentation and fine-tuning. In the system to train and evaluate the suggested model, the ph2 dataset is used. The deep convolutional neural network (DCNN) model divides the three forms of skin cancer visible in colour photographs. Any image can be processed and used in the proposed method. Since it is unnecessary, there is no preprocessing. A softmax layer has taken the place of the categorization on layer. The model eliminates the requirement for photographs with labels. To create a deep neural network, labelled images are necessary. The size of the convolution layer is decreased using the pooling layer. The weights have been refined using backpropagation to include additional weights for categorizing skin lesions. Based on dataset photos, the weights are changed using S.G.D. Stochastic Gradient Descent.

In [8], JinenDaghrir et al. introduced a hybrid technique to diagnose melanoma skin cancer. The model examines any suspect lesions. Three distinct approaches are included in the model. Using a set of features including borders, texture, color, etc, a convolutional neural network and two conventional machine learning classifiers are developed. The ABCDE signals are used in the model as melanoma markers. Based on five features, this trait can distinguish benign skin lesion that develops into malignant melanoma. Using colour enhancement on the RGB's blue component, DoG Filtering is used to detect hair. The 124 x 124-pixel pictures that the CNN architecture that is being suggested uses [24]. The CNN contains nine layers, including three convolutional layers with ReLU activation and three with maximum spatial pooling. Strange skin lesions need to be researched. To achieve better results, adopting semi-supervised learning should be the main focus.

In [9], Vidya M & Dr Maya V Karki suggested an approach that employs the ABCD rule for feature extraction. To extract features for the early diagnosis of skin lesions, GLCM and H.O.G. are utilized. The preprocessing enhances the clarity and quality of skin lesions while lowering artefacts such as hair, skin color, etc. Using Geodesic Active Contour, which separates each instructional section, segmentation was carried out. For extracting attributes, including symmetry, border, color, and diameter, the ABCD scoring system is employed. The retrieved characteristics are sent straight to the classifiers. A clear skin lesion site can be obtained using dermoscopy, improving visual impact. The obtained accuracy is 97.32%. G.A.C. detects the largest changes to the entire skin lesion, typically produced around its edges. RGB images are transformed into grayscale and put through a median filter to reduce the noise. On the same, neural networks may be used.

TABLE I. MERITS AND FUTURE SCOPE WORK ANALYSIS OF EXISTING SYSTEMS

Author	Algorithm	Merits	Demerits
MehwishDildar	S.L.R.	Suggesting auto organization approaches.	Need to implement automation in the model, focused only on NN-based models.
Mariam Nawaz	Faster RCNN	FKM clustering is also used	Chance to overfit
Ulzii-OrshikhDorj	CNN and ECOC SVM	The noise was removed, and RGB images improved the properties of the image.	Can add ABCD rule and implement to get better results.
Mohammad Ali Kadampur	Deep learning and cloud.	Used D.L.S. tools, can deal with	Not implemented into an API
Andre Esteva	CNN	Uses simple pixels and saliency maps are used.	No required number of training records
Ravi Manne	ESLER	Focussed on many issues like misclassification and dermoscopic images.	Need to include the operating results of various models.
Khalid M. Hosny	DCNN	Transfer learning, any mage can be preprocessed.	The weights keep changing
JinenDaghrir	MCNN	ReLU activation function, two ML classifiers.	Should implement for semi-supervised data.
Vidya M	Hybrid feature extraction	Uses ABCDE signs, images conversion into grayscale	Neural networks provide better results.
Shunichi Jinnai	FRCNN	Robust, high classification accuracy	It would help if you also used N.N. for generalization. The project should be implemented socially.

In [10], Shunichi Jinnai et al. built a dataset by randomly choosing items and annotating them with bounding boxes. The model that is being suggested is a quicker, region-based CNN called FRCNN. By combining convolutional features from R.P.N. and Fast R CNN into a single network, FRCNN is the result—a classification system based on neural networks that use clinical images rather than dermoscopic ones. , Robustness, high classification accuracy, and speed were all displayed by the model. A momentum stochastic gradient descent optimizer is utilized, with the VGG-16 as a foundation. To employ wearable technology in public, the network must be socially implemented. The skin cancer prognosis should be used to lower treatment expenses. Reduced patient wait times and unneeded visits are desirable [25]. It is important to test the neural network's generalization using images thoroughly. The overall analysis of the existing approaches is presented in Table I.

III. Research Gaps Identified

1) Unsupervised models to solve the non-linear data have increased the dimensionality of the space.

2) Traditional neural networks make the model learn more number of generalized features rather than specific elements.

3) Backpropagation in neural networks makes the model update the weights more randomly to get optimized results.

IV. Proposed Methodology

The proposed methodology contains 34 attributes in the dermatology dataset. Using machine learning approaches like decision trees or ensemble methodologies also takes many resources to build the tree for simple datasets. So the proposed model implemented customized neural networks instead of static neural networks. The components of the ANN, along with their estimators, are presented below in Table II.

A. Input Layer and its Estimators

In general, the input, without performing any transformations, passes the raw information obtained from the input vector to the hidden layers. It multiplies every value of the feature with random weights and then implements the activation function.

TABLE II. Component Description of Neural Network

Components	Remarks	Estimators	Description
Input Layer	The input layer, along with the input shape, contains activation and several neurons which can be customized.	Number of neurons	Any integer value. Generally, it is equal to the number of features in the dataset.
		Activation Functions	Based on the differentiable properties of the neurons, they are classified into 10
Hidden Layer	Depending on the activation function applied and other metrics, it transforms the input into the desired result using the dot product.	Kernel Size	This helps the CNN to process the imaging unit by unit because, without this, the system has to process n*n*3 at a time. Generally, the system works efficiently with an odd number of filters.
		Padding	The system needs to add some bits in terms of padding values to provide accurate results. In general, two types of paddings are available.
		Stride	It defines the step size, representing the number of pixels to ignore. In general, it will be any n value.
		Activation	Based on the differentiable properties of the neurons, they are classified into 10
Pooling Layer	The neural network size is reduced using the strides and pool size. There are three types of pooling mechanisms.	Pool size	It's a two-dimensional filter that maps the features.
		Stride	It defines the step size, representing the number of pixels to ignore. In general, it will be any n value.
Output Layer	Generally, it is a fully connected layer that produces the desired output based on several neurons and activation functions.	Number of neurons	It depends on the type of classification that the application needs.
		Activation Function	Only 3 possibilities exist for the output layer. Linear, sigmoid and softmax
Optimizer	The updation of neuron weights from one layer to another is performed using optimizers.		There are 6 types of optimizers available
Loss Function	It defines the model's standard by computing the difference between the actual and predicted values.		In general, there are 9 types of loss functions available. Since the proposed dataset is a multi-classification problem, it uses only 3 types of loss functions.

TABLE III. Implemented Activation Functions for Input and Hidden Layers

S.No	Activation Function	Description	Merits
1	Elu	It uses the natural gradients to activate the positive values. It suddenly jumps the mean values [18]	It is low computation, and most values are zero centred
2	Selu	This will always scale the values in such a way that the mean should be zero and the variance should be one	In every iteration, it internally performs the normalization
3	Relu	These output values will never enter into saturation point	The computations are faster
4	Leaky Relu	The model uses simple linear components and adjusts in values in terms of smaller decimal places	It solves the problem of dying Relu
5	Sigmoid	It transforms all the input values between 0 to 1 and produces S- a shape curve	It is easy to differentiate, and prediction values are clear
6	Softplus	It is a derivative of the sigmoid and tanh combination	It always generates smooth curves between -1 to +1
7	Softsign	It transforms the linear values into non-linear values	It is efficient in solving the regression problems
8	Tanh	It is very effective when dealing with negative values and values closer to zero.	When the gradient values are small, then tanh improves the performance.

1) Activation function: The activation function is a non-linear change we apply to input before passing it onto the next neuronal layer or finishing it as output [17]. The network can use crucial information and ignore unnecessary data points using activation functions. An activation function takes the values produced by one network level and changes them in a certain way to transfer them to another layer or range of values. The proposed model implements one input layer and three hidden layers. For all these layers, the model passes all the activation functions that suit the dataset as a list of arguments and choose the highest probability as the best parameter. Table III presents the list of activation functions implemented in the proposed model.

B. Hidden Layers and its Estimators

The proposed model implements all the hidden layers as fully connected layers. The basic principle for any neural network is "the network in which all the hidden layers with the same amount of neurons will have more success rate"[20]. So the proposed model implemented all the hidden layers with the same number of neurons. The hidden layers solve the complex problems by constructing the feature map vector that computes the correlation between each input feature and output class label. In the proposed model, after each hidden layer, it implements an integrated layer that normalizes and drops out the threshold values that are less than the alpha cut. Choosing several hidden layers plays a major role in constructing a neural network. Traditional researchers mentioned that the number of layers should be less than the number of neurons, and also it should be tested from a cross-validation test. Hidden layers are famous for automatic dimensionality reduction, i.e., the number of neurons should be less than the original dimensionality. The below section defines the process of customization.

1) Customization of neurons count: The neuron is triggered if the input to an activation function exceeds a threshold; otherwise, it is deactivated. In extremely unusual circumstances, bias will only have one input layer, with the number of input layer neurons equivalent to the number of features in the data [19]. Bias may rarely have just one input layer, with the number of neurons in the input layer equaling the number of features in the data. At the same time, using the model as a classifier or regressor affects the number of neurons in the output. If the algorithm is a regressor, then the output layer will only include one neuron; however, if the system is a classifier, it may contain one or more neurons, depending on the model's classification label. . As a result, the goal variable affects the count of neurons present in the output layer. The amount of training data, the anomalies, the complexity of data that must be learned from, and the kind of activation functions employed all impact the rate of neurons and layers needed for the hidden layer. Equation 1 presents the computation of the number of optimistic neurons in any layer of the neural network

$$Neurons\_Count = \frac{number\ of\ records\ in\ dataset}{bias\_factor * (input\_neurons * output\_neurons)} \quad - (1)$$

Fig. 1 denotes integrating hidden layers with drop-out layers to get the normalized values from each layer.

C. Output Layer and its Estimators

The number of neurons in the output layer equals the number of class labels available in the dataset. Since the proposed model uses a multi-classification dataset, the last layer of a fully connected pattern uses the softmax activation function. It calculates the relative possibilities for all 6 classes and uses an exponential function to normalize the data [21]. This normalization makes the model such that the total 6 class probabilities are summated to 1. But the proposed model uses the softer version of softmax so that all the prediction class label is 1 and other class labels are 0.

D. Optimizers

An optimizer is a technique or procedure to modify the different parameters that can more efficiently reduce the loss [22]. Different types of optimizers are presented in Fig. 2.

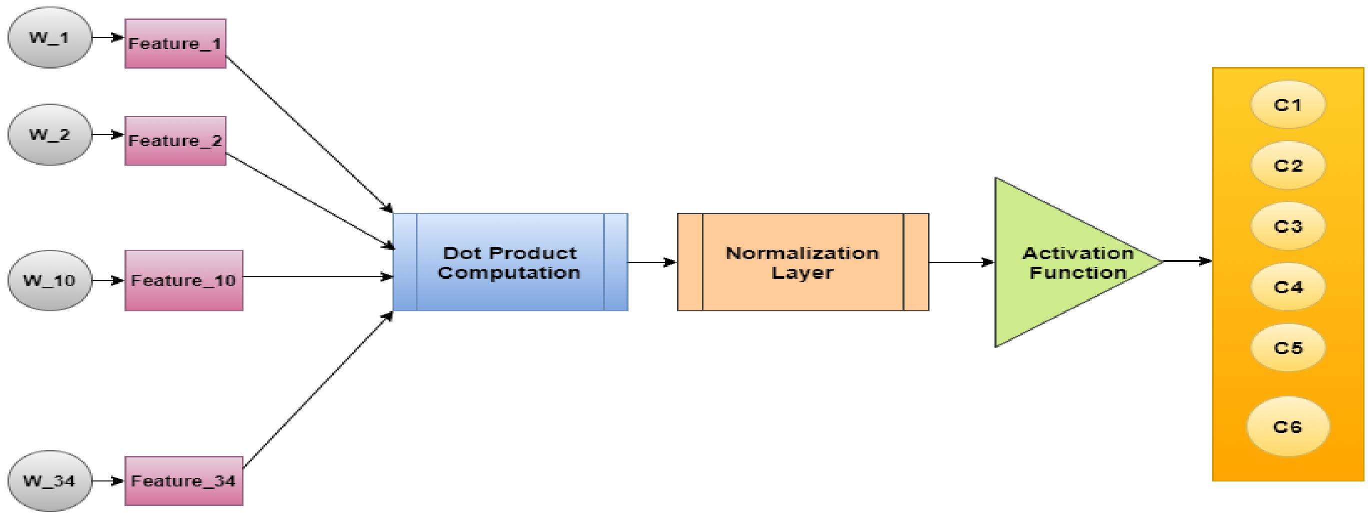


Fig. 1. Integrated architecture of hidden layers.

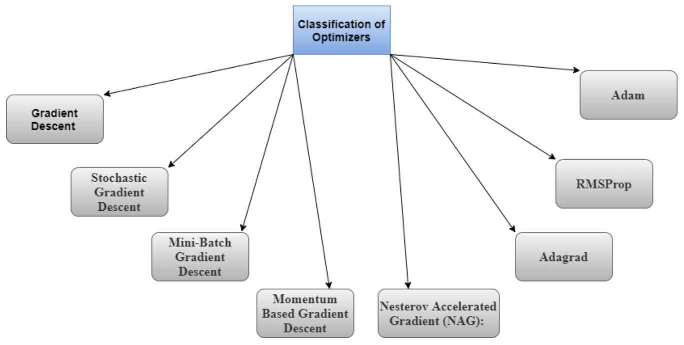

Classification of Optimizers
Gradient Descent
Stochastic Gradient Descent
Mini-Batch Gradient Descent
Momentum Based Gradient Descent
Nesterov Accelerated Gradient (NAG):
Adagrad
RMSProp
Adam

Fig. 2. Classification of optimizers.

1) *Gradient Descent (G.D.):* Calculus is used by the G.D. optimization method to modify the parameters and consistently find the local minimum. This strategy is also used in neural network backpropagation, where the revised parameters are distributed across the various layers based on when the minimal loss is attained.

2) *Stochastic gradient descent:* Every iteration involves updating the model's parameters. It entails testing the loss function and updating the model after each training sample. Thanks to the stochastic gradient, you can pick the data batches at random. It implies you only have to sample a small part of the dataset.

3) *Mini-batch gradient descent:* Only a bit of the dataset is used in the Mini-Batch Gradient Descent to generate the loss function. As a result, all of the datasets need not be analyzed in memory thanks to batching.

4) *Momentum-based gradient descent:* The gradient descent optimization procedure can ride across flat regions of the search space and overcome the oscillations of noisy gradients by adding momentum. This enables the search to acquire inertia within the search space in a specific direction.

5) *Nesterov Accelerated Gradient (N.A.G.):* The strategy used in this case was to first update the parameters with the history component before calculating the derivative, which can move the parameters ahead or backwards. This approach, called the look-ahead technique, is more efficient since it can result in fewer oscillations and more time being saved if the curve moves slowly as it approaches the minima.

6) *Adagrad:* The Adagrad optimizer attempts to provide this adaptability by slowing down the training rate according to the modified history of the gradients. The learning rate does not require manual tweaking.

7) *RMSProp:* RPPROP discretely modifies the step size for each weight using the gradient sign. This approach expedites the optimization method by reducing the total of function estimations to find the local minima.

8) *Adam:* Rather than stochastic gradient descent, an alternative optimization approach called Adam can be employed to develop deep learning models. Adam creates an optimization approach that can manage sparse fluctuations in noisy conditions by combining the best elements of both the RMSGrad and AdaProp algorithms.

E. Loss Functions

The parameter that defines how far the algorithm's current output deviates from the desired output is called the "loss function" [23]. This method is used to judge how well an algorithm matches the data. The parameters that the model learns are established by minimizing a certain loss function, and the loss functions provide a goal against which the model's performance is measured loss functions based on cross-entropy - The distinction between two probabilistic is quantified by cross-entropy. The difference between the probability distribution produced by the activity that produced the data and the distribution that the process model is calculated. The binary cross-entropy is well suited to obtaining one of two outcomes in binary classification scenarios. Multiclass classification uses categorical cross-entropy. In regression circumstances, the model expectation and choices related are real-number values and mean squared error is used. Since the proposed model is a multi-classification problem with six discrete class labels, it implemented "sparse_categorical_crossentropy", whose mathematical representation is presented in equation x. since the loss function computes the difference between true and predicted labels let us consider true class labels as Y_{true} and predicted as $Y_{predict}$

$$Sparse(Y_{true}, Y_{predict}) = -\frac{1}{n} * \sum_{i=0}^{n}[y_{true_i} * \log(y_{predict_i}) + (1 - y_{true_i}) * \log(1 - y_{predict_i}) \quad (2)$$

The proposed model considers three estimators as static and assumes all these estimators with the best values. The remaining component estimators are dynamic and are chosen by the optimizer values of the hyper-tuning process. The architecture of the proposed model is presented in Fig. 3.

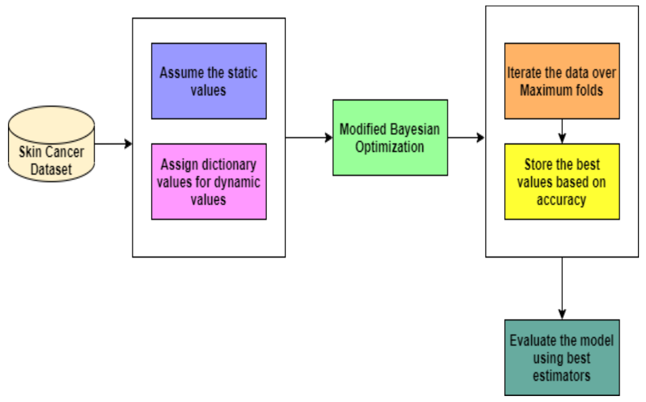

Skin Cancer Dataset
Assume the static values
Assign dictionary values for dynamic values
Modified Bayesian Optimization
Iterate the data over Maximum folds
Store the best values based on accuracy
Evaluate the model using best estimators

Fig. 3. Workflow of the proposed model.

V. Results and Discussion

Table IV denotes the best parameters for every estimator of the component that suits the given dataset for the detection of skin cancer. The proposed model assumes few initial configurations, like it is a designed neural network with three hidden layers, the last dense layer uses softmax function because of the multi-classification, and it implements sparse categorical loss function to evaluate the model.

Fig. 4 denotes the sample screenshot of epochs which have designed the neural network using the best parameters. The training epochs also display the loss and accuracy of training data along with validation loss accuracy. Loss values in almost all iterations are equal to zero for training. So the proposed system is efficient.

Fig. 5 represents the accuracy analysis of existing models with the proposed one to prove state of the art. On X-axis, it denotes the models implemented by the different researchers for identifying skin cancer the Y-axis denotes the percentage accuracies. DCNN has achieved 98% among the existing models. It is the highest accuracy. So when compared to DCNN, the proposed model has achieved +1.65% more.

TABLE IV. BEST ESTIMATORS OF SKIN CANCER DETECTION

S. No.	Estimator Name	Estimator Value
1	Number of neurons in the input layer	47
2	Number of neurons in hidden layer-1	4
3	Number of neurons in hidden layer-2	22
4	Number of neurons in hidden layer-3	24
5	Activation Function for input & hidden layers	Selu
6	Learning Rate	0.46
7	Normalization Rate	0.57
8	Drop Out Rate	0.28
9	Optimizer	Adadelta
10	Batch_size	365
11	Number of epochs	92

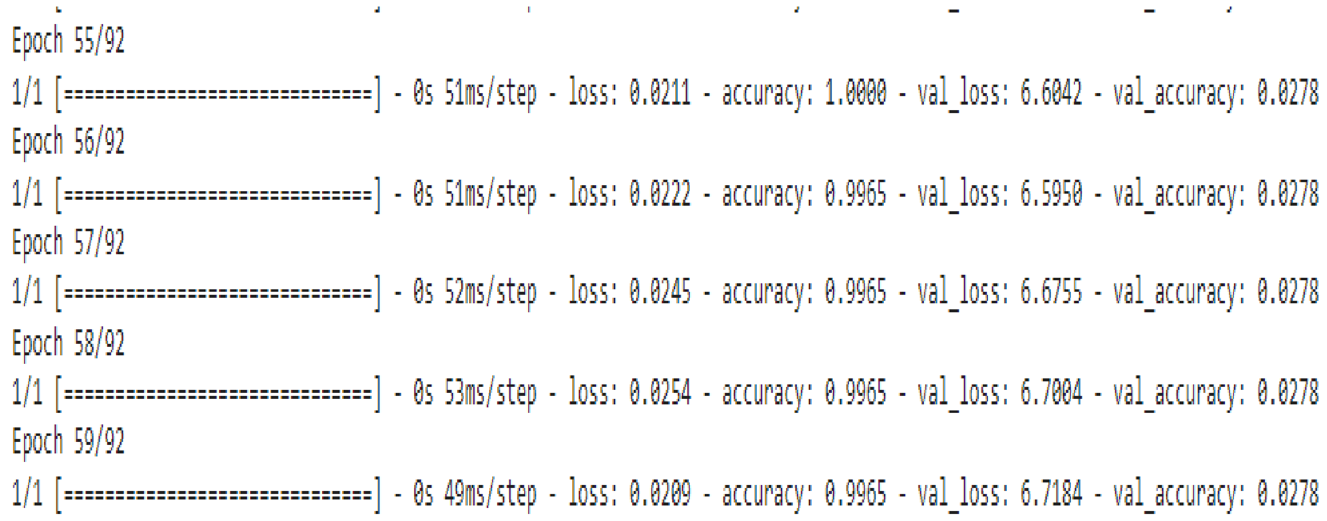

```
Epoch 55/92
1/1 [==============================] - 0s 51ms/step - loss: 0.0211 - accuracy: 1.0000 - val_loss: 6.6042 - val_accuracy: 0.0278
Epoch 56/92
1/1 [==============================] - 0s 51ms/step - loss: 0.0222 - accuracy: 0.9965 - val_loss: 6.5950 - val_accuracy: 0.0278
Epoch 57/92
1/1 [==============================] - 0s 52ms/step - loss: 0.0245 - accuracy: 0.9965 - val_loss: 6.6755 - val_accuracy: 0.0278
Epoch 58/92
1/1 [==============================] - 0s 53ms/step - loss: 0.0254 - accuracy: 0.9965 - val_loss: 6.7004 - val_accuracy: 0.0278
Epoch 59/92
1/1 [==============================] - 0s 49ms/step - loss: 0.0209 - accuracy: 0.9965 - val_loss: 6.7184 - val_accuracy: 0.0278
```

Fig. 4. A sample screen shot epochs with best parameters selected.

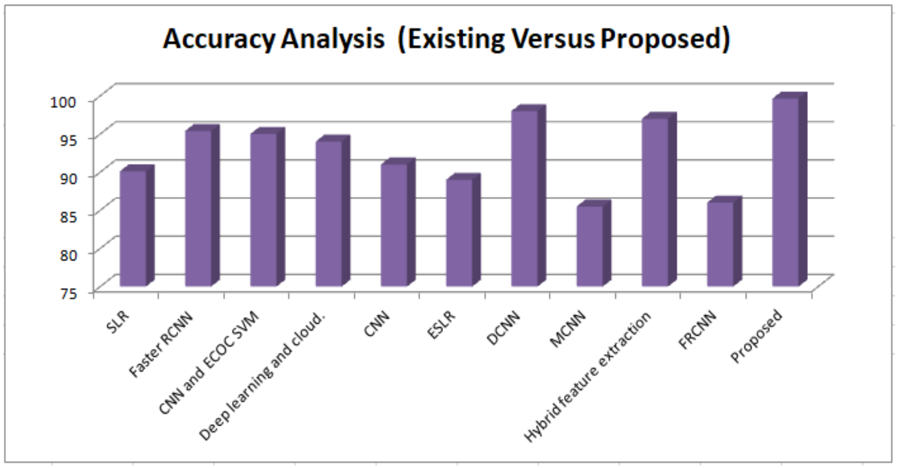
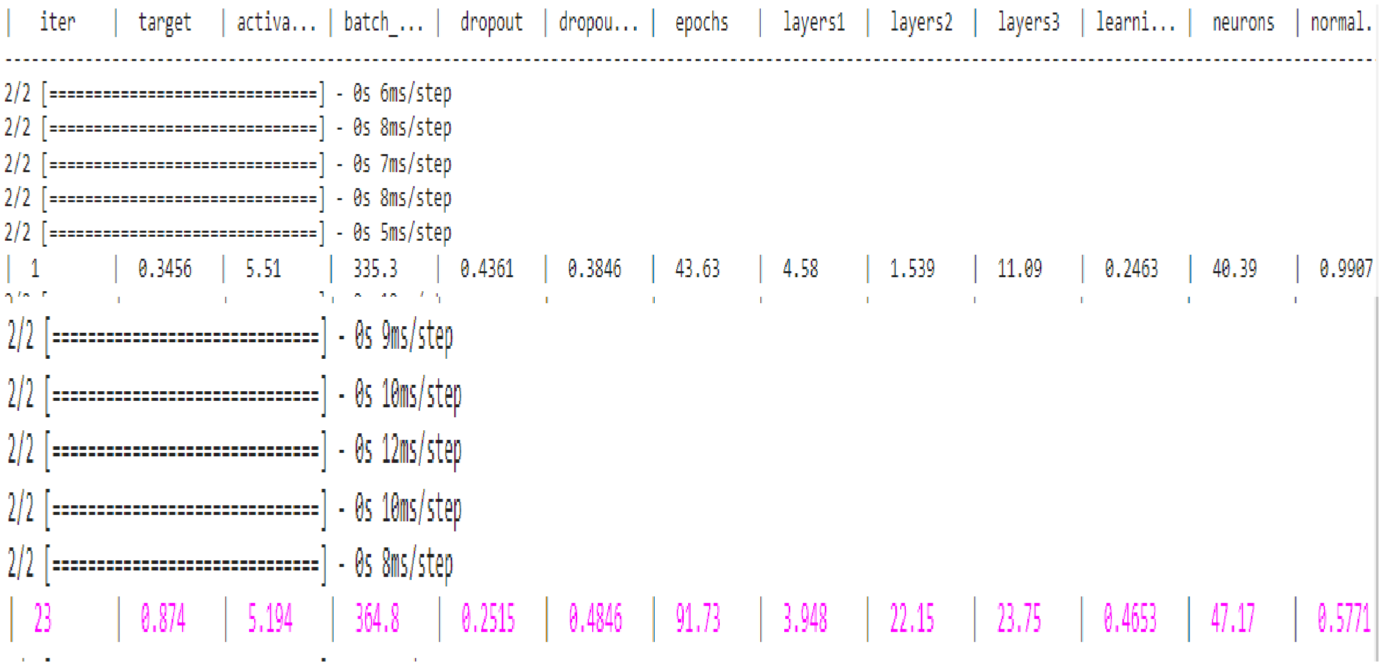


Fig. 5. Evaluation of proposed compared with existing.

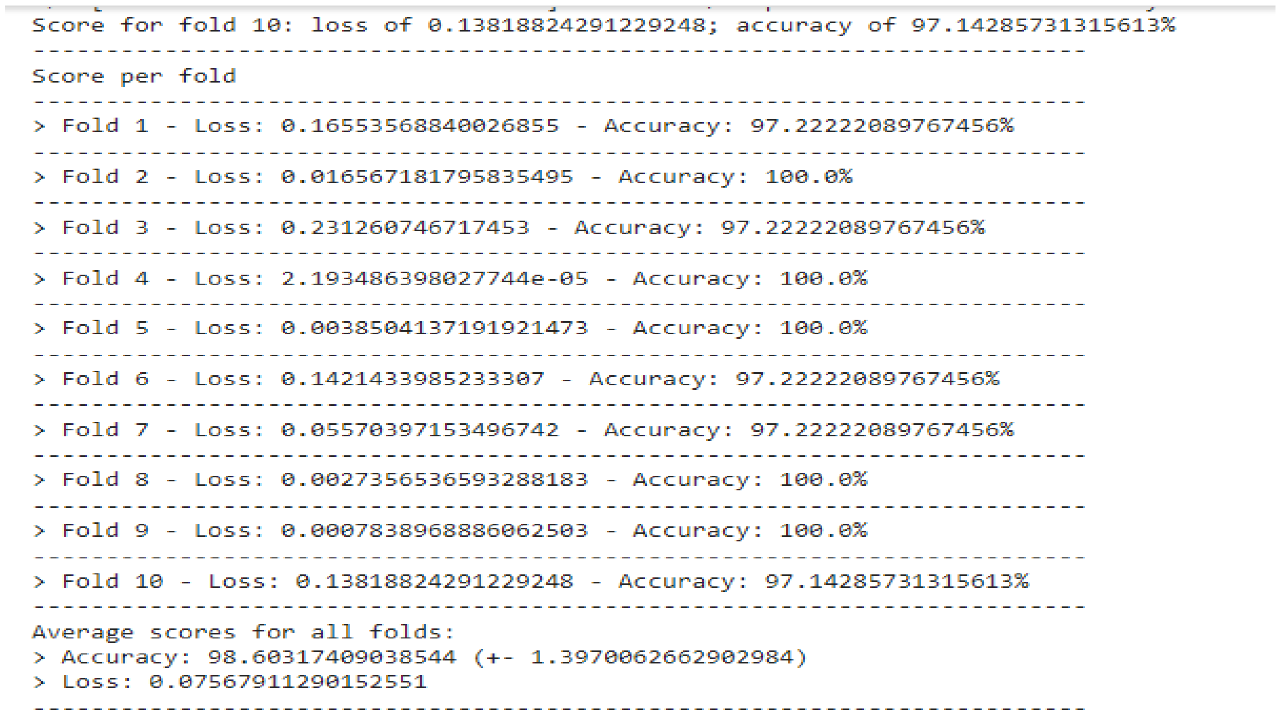


```
|   iter    |  target   | activa... | batch_... |  dropout  | dropou... |  epochs   |  layers1  |  layers2  |  layers3  | learni... |  neurons  | normal.
-------------------------------------------------------------------------------------------------------------------------------------------------------
2/2 [==============================] - 0s 6ms/step
2/2 [==============================] - 0s 8ms/step
2/2 [==============================] - 0s 7ms/step
2/2 [==============================] - 0s 8ms/step
2/2 [==============================] - 0s 5ms/step
| 1         | 0.3456    | 5.51      | 335.3     | 0.4361    | 0.3846    | 43.63     | 4.58      | 1.539     | 11.09     | 0.2463    | 40.39     | 0.9907
2/2 [==============================] - 0s 9ms/step
2/2 [==============================] - 0s 10ms/step
2/2 [==============================] - 0s 12ms/step
2/2 [==============================] - 0s 10ms/step
2/2 [==============================] - 0s 8ms/step
| 23        | 0.874     | 5.194     | 364.8     | 0.2515    | 0.4846    | 91.73     | 3.948     | 22.15     | 23.75     | 0.4653    | 47.17     | 0.5771
```

Fig. 6. 1st Best parameters found at 23rd iteration.

Fig. 6 represents the iterations performed by the modified Bayesian optimization, in which it sets the iterations to 25 and folds to 2. Out of the 50 iterations, it has achieved its first best values at the 23rd iteration and processes the data until it completes all the data processing units. The target is the objective function based on which iteration is best or normal. The higher the target higher the chances of maximization.

Fig. 7 projects the accuracy obtained with the customized ANN by performing 10-fold cross-validation to prove state of the art. The model designs different layers with different activation functions but with a standard number of neurons. Every layer is designed with popular activation functions, but most of the folds got 100% accuracy, representing overfitting. Finally, the average score for the entire iteration with 10 cross-fold validation is "98.6%", less than the proposed model.



```
Score for fold 10: loss of 0.13818824291229248; accuracy of 97.14285731315613%
------------------------------------------------------------------------
Score per fold
------------------------------------------------------------------------
> Fold 1 - Loss: 0.16553568840026855 - Accuracy: 97.22222089767456%
------------------------------------------------------------------------
> Fold 2 - Loss: 0.016567181795835495 - Accuracy: 100.0%
------------------------------------------------------------------------
> Fold 3 - Loss: 0.231260746717453 - Accuracy: 97.22222089767456%
------------------------------------------------------------------------
> Fold 4 - Loss: 2.193486398027744e-05 - Accuracy: 100.0%
------------------------------------------------------------------------
> Fold 5 - Loss: 0.0038504137191921473 - Accuracy: 100.0%
------------------------------------------------------------------------
> Fold 6 - Loss: 0.1421433985233307 - Accuracy: 97.22222089767456%
------------------------------------------------------------------------
> Fold 7 - Loss: 0.05570397153496742 - Accuracy: 97.22222089767456%
------------------------------------------------------------------------
> Fold 8 - Loss: 0.0027356536593288183 - Accuracy: 100.0%
------------------------------------------------------------------------
> Fold 9 - Loss: 0.0007838968886062503 - Accuracy: 100.0%
------------------------------------------------------------------------
> Fold 10 - Loss: 0.13818824291229248 - Accuracy: 97.14285731315613%
------------------------------------------------------------------------
Average scores for all folds:
> Accuracy: 98.60317409038544 (+- 1.3970062662902984)
> Loss: 0.07567911290152551
------------------------------------------------------------------------
```

Fig. 7. Average accuracy score of cross-validation in customized ANN.

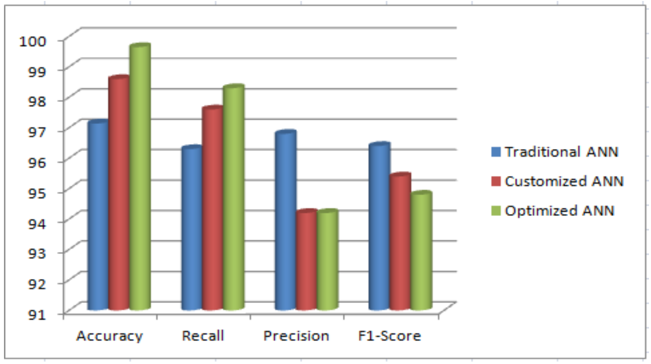


Fig. 8. Metrics evaluation on neural networks algorithms.

Fig. 8 presents the metrics comparison for the three different types of ANN algorithms. Among them optimized ANN has got highest accuracy and recall. X-axis denotes metrics and Y-axis denotes the measurement of metrics.

VI. Conclusion

Identifying Erythemato-Squamous disease using the deep learning approach increases the efficiency of the automation system. The proposed model helps the doctors diagnose the disease with this automation system and predicts the disease at the early stage. Disease identification using the machine learning system has high misclassification and error rates. So few researchers used standard ANNs to train the model with much data and more epochs. But these models are not appropriate because different datasets need different estimators. The proposed model uses optimization techniques to identify the best value for every possible estimator. The model uses a modified version of Bayesian optimization and finds the best values for 11 estimators. The model assumes fixed values like loss functions, number of layers, and number of epochs. Existing optimization techniques need more iterations and memory space. The system to reduce the iterations modifies the Bayesian optimization by comparing the previous iterations and stores only the best values. The model highlights the best values and runs the model till it gets saturation values. Using this model, it has achieved 99.65%. In future work, the model extends the hyper-tuning process by defining a search space, where it can find the parameters depending on the similarity between the attributes and limiting the search space helps the model acquire the learning faster.

References

[1] Dildar, M., Akram, S., Irfan, M., Khan, H. U., Ramzan, M., Mahmood, A. R., ... &Mahnashi, M. H. (2021). Skin cancer detection: a review using deep learning techniques. International journal of environmental research and public health, 18(10), 5479.https://doi.org/ 10.3390/ijerph18105479.

[2] Nawaz, M., Mehmood, Z., Nazir, T., Naqvi, R. A., Rehman, A., Iqbal, M., & Saba, T. (2022). Skin cancer detection from dermoscopic images using deep learning and fuzzy k-means clustering. Microscopy Research and Technique, 85(1), 339-351.https://doi.org/10.1002/jemt.23908.

[3] Dorj, UO., Lee, KK., Choi, JY. et al. The skin cancer classification using deep convolutional neural network. Multimed Tools Appl 77, 9909–9924 (2018). https://doi.org/10.1007/s11042-018-5714-1.

[4] Mohammad Ali Kadampur, Sulaiman Al Riyaee,Skin cancer detection: Applying a deep learning based model driven architecture in the cloud for classifying dermal cell images,Informatics in Medicine Unlocked,Volume 18,2020,100282,ISSN 2352-9148,https://doi.org/10.1016/j.imu.2019.100282.

[5] Esteva, A., Kuprel, B., Novoa, R. et al. Dermatologist-level classification of skin cancer with deep neural networks. Nature 542, 115–118 (2017). https://doi.org/10.1038/nature21056.

[6] Manne, Ravi, SnigdhaKantheti, and Sneha Kantheti. "Classification of Skin cancer using deep learning, convolutional neural Networks-Opportunities and vulnerabilities-A systematic Review." International Journal for Modern Trends in Science and Technology, ISSN (2020): 2455-3778. https://doi.org/10.1016/j.imu.2019.100282.

[7] Hosny, K. M., Kassem, M. A., &Foaud, M. M. (2018). Skin Cancer Classification using Deep Learning and Transfer Learning. 2018 9th Cairo International Biomedical Engineering Conference (CIBEC). doi:10.1109/cibec.2018.8641762.

[8] Daghrir, J., Tlig, L., Bouchouicha, M., &Sayadi, M. (2020). Melanoma skin cancer detection using deep learning and classical machine learning techniques: A hybrid approach. 2020 5th International Conference on Advanced Technologies for Signal and Image Processing (ATSIP). doi:10.1109/atsip49331.2020.9231544.

[9] Vidya, M., & Karki, M. V. (2020). Skin Cancer Detection using Machine Learning Techniques. 2020 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT). doi:10.1109/conecct50063.2020.9198489.

[10] Jinnai, S.; Yamazaki, N.; Hirano, Y.; Sugawara, Y.; Ohe, Y.; Hamamoto, R. The Development of a Skin Cancer Classification System for Pigmented Skin Lesions Using Deep Learning. Biomolecules 2020, 10, 1123. https://doi.org/10.3390/biom10081123.

[11] Mohammad Alnabhan, Ahmad Khader Habboush, Qasem Abu Al-Haija, Arup Kumar Mohanty, SaumendraPattnaik, Binod Kumar Pattanayak, "Hyper-Tuned CNN Using E.V.O. Technique for Efficient Biomedical Image Classification", Mobile Information Systems, vol. 2022, Article ID 2123662, 12 pages, 2022. https://doi.org/10.1155/2022/2123662.

[12] Karegowda, A. G., & G., D. (2022). Meta-Heuristic Parameter Optimization for ANN and Real-Time Applications of ANN. In I. Management Association (Ed.), Research Anthology on Artificial Neural Network Applications (pp. 166-201). I.G.I. Global. https://doi.org/10.4018/978-1-6684-2408-7.ch008.

[13] H. Alibrahim and S. A. Ludwig, "Hyperparameter Optimization: Comparing Genetic Algorithm against Grid Search and Bayesian Optimization," 2021 IEEE Congress on Evolutionary Computation (C.E.C.), 2021, pp. 1551-1559, doi: 10.1109/CEC45853.2021.9504761.

[14] Singh, P., Chaudhury, S., &Panigrahi, B. K. (2021). Hybrid MPSO-CNN: Multi-level Particle Swarm optimized hyperparameters of Convolutional Neural Network. In Swarm and Evolutionary Computation (Vol. 63, p. 100863). Elsevier B.V. https://doi.org/10.1016/j.swevo.2021.100863.

[15] Pietruszka, M., Borchmann, Ł., &Graliński, F. (2021). Successive Halving Top-k Operator. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 35, Issue 18, pp. 15869–15870). Association for the Advancement of Artificial Intelligence (AAAI). https://doi.org/10.1609/aaai.v35i18.17931.

[16] Sajedi, S., & Liang, X. (2021). Deep generative Bayesian optimization for sensor placement in structural health monitoring. In Computer-Aided Civil and Infrastructure Engineering (Vol. 37, Issue 9, pp. 1109–1127). Wiley. https://doi.org/10.1111/mice.12799.

[17] Jagtap, A. D., Shin, Y., Kawaguchi, K., &Karniadakis, G. E. (2022). Deep Kronecker neural networks: A general framework for neural networks with adaptive activation functions. In Neurocomputing (Vol. 468, pp. 165–180). Elsevier B.V. https://doi.org/10.1016/j.neucom.2021.10.036.

[18] Apicella, A., Donnarumma, F., Isgrò, F., &Prevete, R. (2021). A survey on modern trainable activation functions. In Neural Networks (Vol. 138, pp. 14–32). Elsevier B.V. https://doi.org/10.1016/j.neunet.2021.01.026.

[19] Dey, N., Zhang, Y.-D., Rajinikanth, V., Pugalenthi, R., & Raja, N. S. M. (2021). Customized VGG19 Architecture for Pneumonia Detection in Chest X-Rays. In Pattern Recognition Letters (Vol. 143, pp. 67–74). Elsevier B.V. https://doi.org/10.1016/j.patrec.2020.12.010.

[20] Muhammad Mazhar Bukhari, Bader Fahad Alkhamees, Saddam Hussain, Abdu Gumaei, Adel Assiri, Syed Sajid Ullah, "An Improved

Artificial Neural Network Model for Effective Diabetes Prediction", Complexity, vol. 2021, Article ID 5525271, 10 pages, 2021. https://doi.org/10.1155/2021/5525271.

[21] ThangaSelvi, R., Muthulakshmi, I. RETRACTED ARTICLE: An optimal artificial neural network based big data application for heart disease diagnosis and classification model. J Ambient Intell Human Comput 12, 6129–6139 (2021). https://doi.org/10.1007/s12652-020-02181-x.

[22] Abdolrasol M.G.M., Hussain SMS, Ustun TS, Sarker MR, Hannan MA, Mohamed R, Ali JA, Mekhilef S, Milad A. Artificial Neural Networks Based Optimization Techniques: A Review. Electronics. 2021; 10(21):2689. https://doi.org/10.3390/electronics10212689.

[23] Goceri, E. (2021). Diagnosis of skin diseases in the era of deep learning and mobile technology. In Computers in Biology and Medicine (Vol. 134, p. 104458). Elsevier B.V. https://doi.org/10.1016/j. compbiomed.2021.104458.

[24] Attique Khan, M., Akram, T., Sharif, M., Kadry, S., & Nam, Y. (2021). Computer Decision Support System for Skin Cancer Localization and Classification. In Computers, Materials &amp; Continua (Vol. 68, Issue 1, pp. 1041–1064). Computers, Materials and Continua (Tech Science Press). https://doi.org/10.32604/cmc.2021.016307.

[25] W. O'Keefe, B. Ide, M. Al-Khassaweneh, O. Abuomar and P. Szczurek, "A CNN Approach for Skin Cancer Classification," 2021 International Conference on Information Technology (ICIT), 2021, pp. 472-475, doi: 10.1109/ICIT52682.2021.9491760.

[26] Balaji, M.S.P., Saravanan, S., Chandrasekar, M. et al. RETRACTED ARTICLE: Analysis of basic neural network types for automated skin cancer classification using Firefly optimization method. J Ambient Intell Human Comput 12, 7181–7194 (2021). https://doi.org/10.1007/s12652-020-02394-0.

[27] P. S. Silpa et al., "Designing of Augmented Breast Cancer Data using Enhanced Firefly Algorithm," 2022 3rd International Conference on Smart Electronics and Communication (ICOSEC), Trichy, India, 2022, pp. 759-767, doi: 10.1109/ICOSEC54921.2022.9951883.