Skin Cancer Detection Using CNN

Dr. Ch Raja[1], Dr. Shaik Irfan Babu[2]

[1]**Associate Professor, Dept. of ECE, Mahatma Gandhi Institute of Technology, Hyderabad, Telangana, India.**

[2]**Assistant Professor, Dept. of CSEET, Mahatma Gandhi Institute of Technology, Hyderabad, Telangana, India.**

ABSTRACT

This paper aims to detect and classify skin cancer using state-of-the-art machine learning techniques. The HAM10000 (Human Against Machine) dataset, which contains 10,015 labelled images of skin lesions, was released for academic research and has since been widely used to showcase various machine learning approaches. Neural networks, inspired by the human brain, have become increasingly popular across numerous domains. A specialized variant known as Convolutional Neural Networks (CNNs) has proven particularly effective for image processing tasks. Standard architectures such as VGG16, ResNet50, GoogLeNet, and LeNet are recognized for their high performance and have consistently ranked among the top models in the annual ImageNet competition. Although one of these champion models could have been applied to this classification problem, the dataset used is highly imbalanced and skewed. Without appropriate data representation, even the most advanced models struggle to learn the distinguishing features between classes effectively. To address this challenge, several experiments were conducted to enhance the accuracy of the classification results.

Keywords: Skin Cancer, HAM10000,Convolutional Neural Network (CNN), Imbalanced Dataset, VGG16, ResNet50, GoogLeNet, LeNet

INTRODUCTION

Skin cancers are malignant growths that originate from the skin's layers. These cancers result from the uncontrolled proliferation of abnormal skin cells, which can invade surrounding tissues and metastasize to other parts of the body. Over 90% of skin cancer cases are attributed to prolonged exposure to ultraviolet (UV) radiation from the Sun. The depletion of the ozone layer and the growing popularity of artificial tanning methods, such as tanning beds, have significantly increased UV exposure, thereby elevating the risk of developing skin cancer. Accurate and early diagnosis of skin cancer is crucial for effective treatment and improved survival rates. Traditionally, diagnosis relies heavily on dermatologists' expertise, which may not always be readily accessible, particularly in under-resourced regions. Therefore, an automated system that can assist in the early detection and classification of skin cancer types would be highly valuable. This technique is centred on developing a machine learning-based classifier capable of identifying and distinguishing between various types of skin cancer from dermatoscopic images. By leveraging deep learning techniques, particularly Convolutional Neural Networks (CNNs), this project proposes a robust classification framework. Two

different deep learning models have been constructed and evaluated, each designed to learn and extract distinctive features from the images to accurately predict cancer types. The comparative analysis of these models highlights their individual strengths, limitations, and suitability for real-world applications. With access to a larger and more diverse dataset, the models can be further trained and optimized to improve their performance, potentially achieving state-of-the-art accuracy. Such advancements would have a transformative impact on the field of dermatology and oncology, offering scalable solutions for early detection and diagnosis, ultimately saving lives and benefiting global healthcare. It includes a comparative discussion of popular deep learning models and their effectiveness in addressing similar challenges. Following this, the core chapters delve into the technical aspects of the project—covering data pre-processing, model architecture, training procedures, performance evaluation, and observed outcomes.

LITERATURE SURVEY

Alex Krizhevsky ,et.al., "ImageNet Classification with Deep Convolutional Neural Networks. Neural Information Processing Systems", has explored the dataset in a beautiful manner, entirely wholesome for the purpose of classification. Exploration of the dataset is a prerequisite for machine learning. We must ensure that the dataset is balanced and processed in order to effectively produce a working model. Due to limitations of processing power, the size of the image has to be reduced. This is exactly what Alex did; he resized every image into 100x75 pixels, allowing the kaggle processors to work better and product faster results. The Machine learning model created in this case is a classic deep convolution neural network. This usually involves a series of convolution layers with tapering kernel size and increasing channel size, intercepted by a dropout layer every now and then. The author has rightly relied upon Categorical Cross Entropy, which is required for multi class classification models, also called Softmax loss. It is Softmax activation plus a Cross-Entropy loss. If we use this loss, we will train a CNN to output a probability over the classes for each image. It is used for multi-class classification. A slightly preceding model to the adam optimizer is the Adadelta, which is Adaptive Detla optimizer. This is of great importance for learning rate annealing, which requires special heuristics to ensure rapid and efficient convergence of the convolution network model.

Detection and analysis of Skin cancer:

Figure1 comprises of all the steps taken in order to create a trained machine learning model which classifies skin cancer into 7 different types. The subsequent sections of this chapter explain every step in detail.

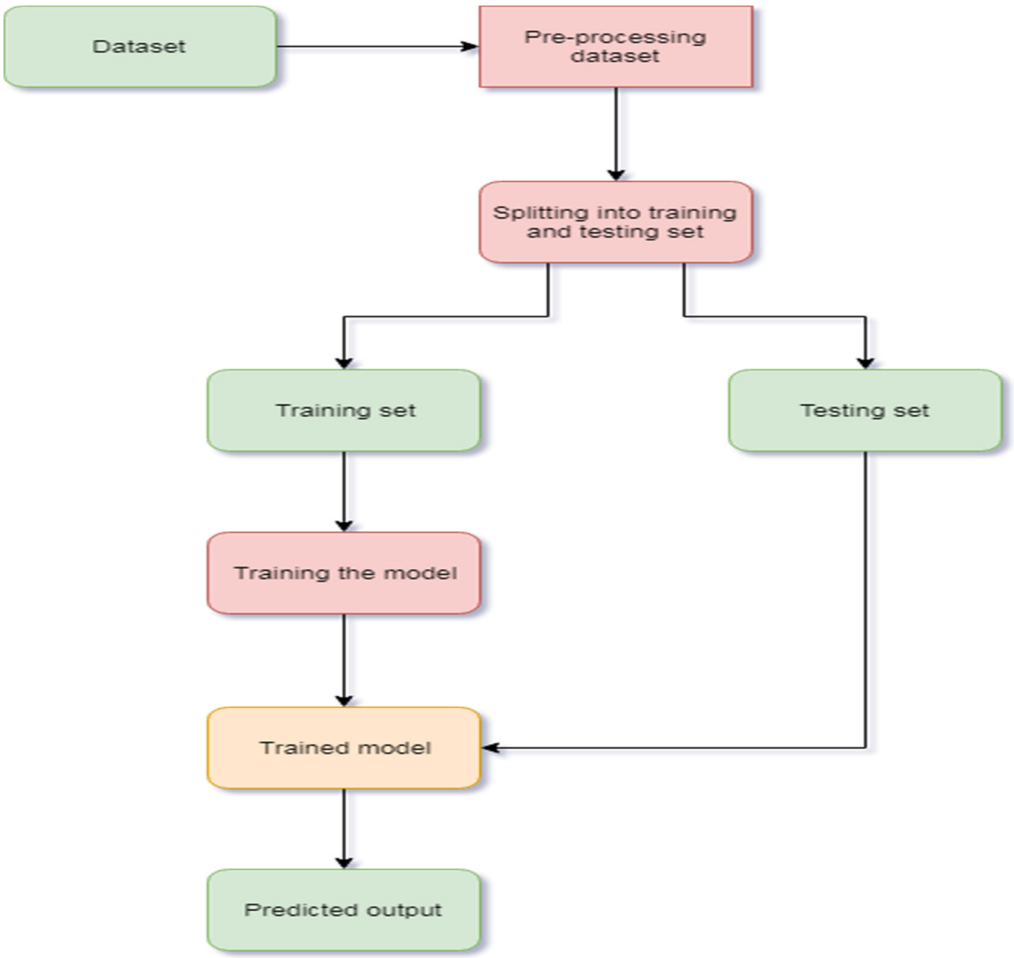

Figure1: Flow chart to create the model

Pre-processing and preparation of data

Preprocessing data enables the neural network to identify the significant features which is a rather difficult task for a neural network on its own. Since, the following model classifies the skin cancer solely based on the image of the kin lesion, the most popular form of preprocessing images is normalizing the pixel values. It has been proven that normalizing the input increases the accuracy and training speed of the model.

Normalization

For the dataset provided, two types of normalizations were used. Those are Min-Max normalization and Z-score normalization. Out of these, the Z-score normalization has produced a significant accuracy, unlike that of min-max normalization.

Z-Score normalization:

The primary advantage of using the Z-score normalization is the forced binding of the dataset to a single distribution. Machine learning models find it hard to learn when the dataset consists of different distributions, thus slowing down the learning process and the classification will lead to higher error rates due to the non-familiarity of unseen data. Thus, constraining the data to a particular distribution helps with improved accuracy. It is highly advised to create a preprocessing pipe line which performs normalization. Figure 2. displays the difference between the normalized image and the image prior to normalization. In this case, every channel (RGB) of the image has been normalized individually.

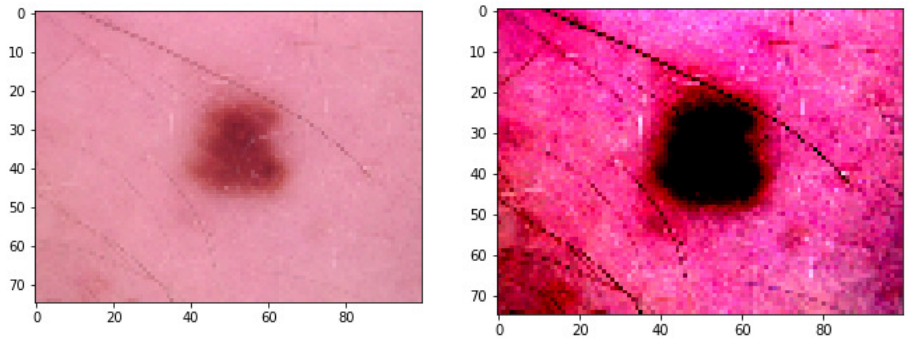

Figure2: Before and after Normalization

Splitting dataset into training and testing set

It's a standard practice to split the available data into two sets, the training set and the validation set. The primary purpose of machine learning is to identify the general patterns in the data in order to predict or classify unseen data. Thus, we would like to test the performance of our model by running on a validation set that validates the classification and produces an accuracy score. This is a very important step as it contributes to detecting overfitting, a real programmer's nightmare. Overfitting is characterized by high accuracy of the model over the training set and lower accuracy over the validation set. Overfitting occurs when the model is identifies patterns and features which are local to the training set, rather than the general population as a whole. Remember, an overfitted model is utterly useless and a massive waste of resources.

Residual Neural Network Model

The model generated for this task is a specific convolution neural network known as residual neural networks. It is designed with the primary idea of keeping vanishing gradient at bay, which is an issue that stalls the learning and makes all the resources spent fruitless.

This model has 4 different types of convolution layers, each varying in the number of filters and their dimension.

The convolution layers are:

1. 64 channel, 3x3 kernel
2. 128 channel, 3x3 kernel
3. 256 channel, 3x3 kernel
4. 512 channel, 3x3 kernel

Structure of the Residual Neural network

Layer (type)	Output Shape	Parameter #
max_pooling2d_7 (MaxPooling2	(None, 18, 24, 64)	0
conv2d_169 (Conv2D)	(None, 37, 49, 64)	1792
conv2d_170 (Conv2D)	(None, 18, 24, 64)	36928
conv2d_171 (Conv2D)	(None, 18, 24, 64)	36928
conv2d_172 (Conv2D)	(None, 18, 24, 64)	36928
conv2d_173 (Conv2D)	(None, 18, 24, 64)	36928
conv2d_174 (Conv2D)	(None, 18, 24, 64)	36928
conv2d_175 (Conv2D)	(None, 18, 24, 64)	36928
conv2d_176 (Conv2D)	(None, 8, 11, 128)	73856

conv2d_177 (Conv2D)	(None, 8, 11, 128)	147584
conv2d_178 (Conv2D)	(None, 8, 11, 128)	147584
conv2d_179 (Conv2D)	(None, 8, 11, 128)	147584
conv2d_180 (Conv2D)	(None, 8, 11, 128)	147584
conv2d_181 (Conv2D)	(None, 8, 11, 128)	147584
conv2d_182 (Conv2D)	(None, 8, 11, 128)	147584
conv2d_183 (Conv2D)	(None, 3, 5, 256)	295168
conv2d_184 (Conv2D)	(None, 3, 5, 256)	590080
conv2d_185 (Conv2D)	(None, 3, 5, 256)	590080
conv2d_186 (Conv2D)	(None, 3, 5, 256)	590080
conv2d_187 (Conv2D)	(None, 3, 5, 256)	590080
conv2d_188 (Conv2D)	(None, 3, 5, 256)	590080
conv2d_189 (Conv2D)	(None, 3, 5, 256)	590080
conv2d_190 (Conv2D)	(None, 1, 2, 512)	1180160
conv2d_191 (Conv2D)	(None, 1, 2, 512)	2359808

conv2d_192 (Conv2D)	(None, 1, 2, 512)	2359808
conv2d_193 (Conv2D)	(None, 1, 2, 512)	2359808
conv2d_194 (Conv2D)	(None, 1, 2, 512)	2359808
conv2d_195 (Conv2D)	(None, 1, 2, 512)	2359808
conv2d_196 (Conv2D)	(None, 1, 2, 512)	2359808
flatten_7 (Flatten)	(None, 1024)	0
dense_25 (Dense)	(None, 1000)	1025000
dense_26 (Dense)	(None, 500)	500500
dense_27 (Dense)	(None, 250)	125250
dense_28 (Dense)	(None, 4)	1004

The convolution layers are interceded with batch normalization layers in an attempt to reduce covariate shift and improve training speed. Another alternative would have been to use dropout layers instead of normalization layers, but they also contribute in avoiding the dreaded case of overfitting. After every 4 sets of convolution layers, a convolution layer without the activation function is provided so that the result produced can be added to the earlier result produced by the layer, preceding the current layer by four layers. Finally, after the series of convolution layers, we add a flatten layer which converts the 2D representation of the outputs from the convolution layer into a single dimension output. The flatten layer allows, us to further connect several layers of dense feed-forward layers, which eventually conclude with an output layer.

In this example, utilized a 3 layer deep feed-forwards which ends with an output layer with a Softmax activation function. Our task being a multi classification task, cannot be expressed with sigmoid function, instead we use the Softmax functions which is the generalized version of the sigmoid function specifically meant for multi class classification. Sigmoid and Softmax can be alternatively used for binary classification.

Optimizer

Adaptive momentum (Adam optimizer) is the latest and most effective of the currently available adaptive rate adjusting algorithms. It detects the change in momentum during back propagation, allowing it to smartly adjust the decaying learning rate, to improve the convergence rate to optimal solution. Since this task is a multi-classification problem, the loss function employed is categorical cross entropy which is an extension to the binary cross entropy.

The training was performed with a batch size of 19 for 90 epochs. Several other batch sizes were tried and tested in an attempt to find the proper value to avoid overfitting. The standard convention identified is that, the batch size must be decreased to increase fitting and reduced to avoid overfitting. Thus, by carefully adjusting the number of epochs and batch size, we can avoid the usage of a dropout layer.

Accuracy and observations

Validation set for testing purpose wasn't used to determine test metrics during training as it drastically reduced the accuracy for the final result. Since image processing demands a massive dataset and HAM10000 has certain classes barely exceeding 100 occurrence which inhibits the usage of validation set approach. Fig 6.1 displays the graphical representation of Training Metrics for every epoch.

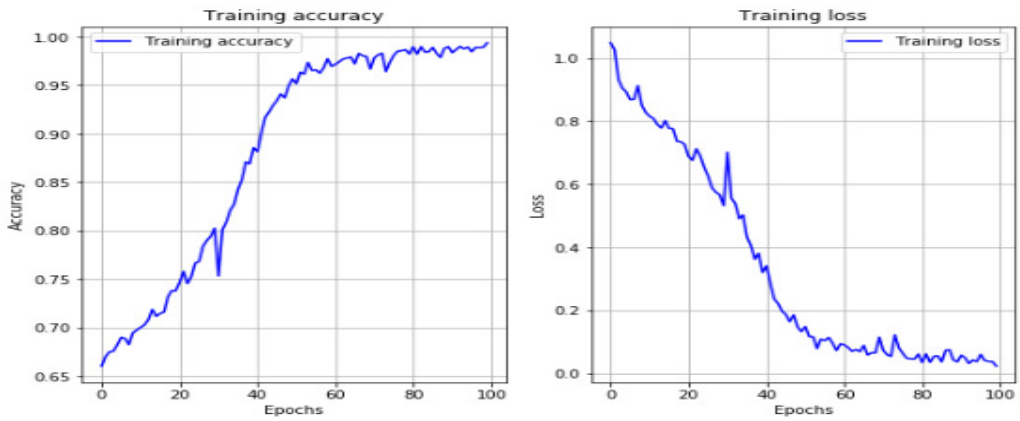

Figure 3: Graphs displaying training accuracy and training loss for every epoch

As a validation set isn't used for training, it is very easy for over fitting to occur. As can be seen in this case, the training accuracy is a massive 99% as shown in Figure 3, however when the final model was tested on the test set the accuracy was still a remarkable 76%. Perhaps, with further tuning we can achieve a greater accuracy.

```
Epoch 1/100
8012/8012 [==============================] - 29s 4ms/step - l
oss: 1.0491 - acc: 0.6605
```

Figure 4: First epoch of training

```
Epoch 28/100
6345/8012 [======================>.......] - ETA: 4s - loss:
0.5713 - acc: 0.7887
```

Figure 5: Mid-training, notice the increment in accuracy

From Figure 4 and 5 notice how the accuracy increases over time with every epoch as the model begins to learn.

Adaptive boosting model

An Adaboosted model is an ensemble technique which is widely relied upon when imbalanced data is required to be classified. This type of model forces the individual models to specially focus and identify the features of underrepresented classes. It is done by resampling a training set by favoring misclassified examples. There are several other ensemble methods; however a variation of Adaboosting is what stuck with me as I realized that other ensemble techniques are specifically used for tree style model.

The traditional Adaboosting model was created for binary classification and libraries with code already existed. Since, decision trees aren't the best model for image classification, the code to incorporate Convolution Neural Networks into Adaboosting, and thus we could call this a neural ensemble technique.

In this paper, created a basic neural network whose instance will be repeatedly spawned and trained to create the ensemble. Adaboosted technique makes use of 30 models trained over selected samples of training data to learn specific features of the images.

Architecture of Base Model

This model has 3 different convolution layers, following the standard Alex net design. Didn't require much thought as this is a tried and tested model.

1) 32 channel with 3x3 filter
2) 64 channel with 3x3 filter
3) 128 channel with 3x3 filter

Structure of the Base neural network model

_____

Layer (type) Output Shape Parameter #

==========

conv2d_205 (Conv2D) (None, 73, 98, 64) 1792

_____

batch_normalization_198 (Bat (None, 73, 98, 64) 256

_____

conv2d_206 (Conv2D) (None, 36, 48, 128) 73856

_____

batch_normalization_199 (Bat (None, 36, 48, 128) 512

_____

conv2d_207 (Conv2D) (None, 17, 23, 256) 295168

_____

batch_normalization_200 (Bat (None, 17, 23, 256) 1024

_____

conv2d_208 (Conv2D) (None, 8, 11, 512) 1180160

_____

batch_normalization_201 (Bat (None, 8, 11, 512) 2048

_____

flatten_10 (Flatten) (None, 45056) 0

_____

dense_35 (Dense) (None, 128) 5767296

_____

dropout_10 (Dropout) (None, 128) 0

_____

dense_36 (Dense) (None, 64) 8256

_____

dense_37 (Dense) (None, 4) 260

==========

Total parameters: 7,330,628

Trainable parameters: 7,328,708

Non-trainable parameters: 1,920

_____

Every convolution layer is intercepted with normalization layers as these improve convergence speed and reduce overfitting as an added bonus. The final three layers are fully connected dense layers, with rectilinear activation function. The output layer makes use of Softmax function for multiclass classification.

Optimizer

Adaptive momentum (Adam optimizer) being one of the best and the most effective of the currently available adaptive rate adjusting algorithms is utilized for the base models. Since this base model must again classify the images into multiple classes, the loss function employed is categorical cross entropy.

Aggregation of models

Thirty models would suffice for an improvement in classification when compared to the non-ensemble method. All these models are weak classifiers which when combined can improve the accuracy by a considerable amount.

For each model, we need a training iteration. For each iteration, a sample of the training data is selected based on weights. These weights are adjusted in every iteration, based on the previous model's classification result in order to focus on misclassified samples. After the models are trained, they are tested with the unused examples of the training set and the accuracy obtained is used to determine the effectiveness of the neural networks which is used to determine the contribution of the

network in the overall prediction. The models with higher accuracy contribute more in weighted votes for determining the final results.

```
Training Model:  1
Training Accuracy:  0.9939999985694885
7012/7012 [==============================] - 1s 150us/step
Testing Accuracy:  [1.6868633169094307, 0.6896748431260696]
```

Figure 6: The Accuracy of the first model generated

Figure 6. is a snapshot of the accuracy of the first model generated. Similarly, 30 different models were generated with varying accuracies ranging from 50 percent to 70 percent, each model targeting a different feature.

Accuracy and observation

The prediction process is performed by determining the maximum weighted sum of the predictions from individual networks. Figure 7. displays the weighted sum of predictions for every class, the estimated class is obtained by using the argmax() function of Numpy.

```
array([[2.05248681e-03, 2.63391411e-03, 1.14785097e-02, ...,
        7.10601120e-03, 8.10162942e+01, 2.51405906e-01],
       [5.48931566e-05, 1.26762776e-04, 7.60617895e-02, ...,
        3.03465016e-03, 8.12120976e+01, 7.63436915e-05],
       [2.21022261e-04, 3.23869478e-03, 1.62769588e-01, ...,
        9.55574872e+00, 7.15619312e+01, 6.89642535e-03],
       ...,
       [6.63940109e-05, 4.33452983e-06, 1.67947966e-02, ...,
        9.60474433e-02, 8.11781790e+01, 3.92433288e-04],
       [1.35190950e+00, 2.60744002e-01, 1.36430598e+01, ...,
        1.03901924e+01, 5.52068331e+01, 3.48745507e-01],
       [9.53152386e-01, 1.42448077e-02, 7.07045337e+00, ...,
        3.26087835e+01, 4.06215257e+01, 8.86899083e-03]])
```

Figure 7: Prediction matrix

```
Overall Accuracy is:  73.33999001497753 %
```

Figure 8: Accuracy of the ensemble

It is interesting to find that even though none of the individual models have an accuracy that exceeds 70%, the overall accuracy of all 30 models combined is 73.3% as displayed by Figure 8.

Conclusion

It is fascinating and highly unusual; to find that large neural networks often perform worse than shallow ones due to the diminishing gradient problem. Diminishing gradient occurs when repeated differentiation causes imploding values, this eventually produces zero gradient value, thus nullifying weight change. Hence, the neural network will remain stuck in the dangerous doldrums of n-dimensional space. Residual Neural Network solves this by creating bypasses to previous layers so that gradients are never zero, thus always allowing the network to change its weights and navigate to lower regions of the hyper-space. Hence, Residual Neural Networks are designed to be very deep for improved performance. The only drawback would be the significantly larger training time. Unfortunately, Kaggle has a limit on the training time permitted and this inhibits me from experimenting with bigger and better models. Adaboosting algorithm is a popular algorithm which boosts models and improves performance as a whole. It is exciting to note that a collection of weak classifiers can produce improved accuracies when used together. Champion neural networks models usually consist of an ensemble of massive networks capable boosting the accuracy to astounding extents. Thus, a combination of the models described in the project could greatly boost the performance.

References:

[1]Christos Stergiou, Imperial College London and DimitriosSiganos,Imperial College London, "Neural Networks" in "*Study"*, [Online article], 2019

[2]Apoorva Agarwal, "Loss Functions and Optimization Algorithms. Demystified." in"*Data Science Group IIT Roorkie, Medium*", [Online article], 2017.

[3]KeviMader, CTO at 4Quant and Lecturer at ETH Zürich, Switzerland,"Dermatology MNIST: Loading and Processing" in "Kaggle", [Online Kernel], 2018.

[4] Sergey Ioffe, Google and Christian Szegedy, Google, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift" in "Arxiv.org",[Published Paper] 2015. Available reference online, https://arxiv.org/abs/1502.03167v3 [Accessed: March 2, 2019]

[5] Alex Krizhevsky ,University of Toronto, Ilya Sutskever, University of Toronto and Geoffrey E. Hinton, University of Toronto. "ImageNet Classification with Deep Convolutional Neural Networks. Neural Information Processing Systems", 2012. 25. 10.1145/3065386.

[6]Tschandl, Philipp &Rosendahl, Cliff & Kittler, Harald, 2018. "The HAM10000 Dataset: A Large Collection of Multi-Source Dermatoscopic Images of Common Pigmented Skin Lesions. Scientific Data". 5. 10.1038/sdata.2018.161.

[7]Schwenk, Holger. (2001)."AdaBoosting Neural Networks: Application to on-line Character Recognition" 10.1007/BFb0020278.

[8] Bergstra, James & Pinto, Nicolas & Cox, David, 2015. SkData: Data sets and algorithm evaluation protocols in Python. Computational Science & Discovery. 8. 014007. 10.1088/1749-4699/8/1/014007.

[9] ZohebAbai, NishadRajmalwar, [Preprint] 2019. “DenseNet Models for Tiny ImageNet Classification”.

[10] Open Source, “Sci-Kit learn User Guide” in “Sci-kit learn”, [Online Documentation] 2007. Available online reference, https://scikit-learn.org/stable/user_guide.html [Accessed: August, 2017]

[11] Tharwat, Alaa. Frankfurt University of Applied Sciences, 2018. “AdaBoost classifier: an overview” 10.13140/RG.2.2.19929.01122.

[12] Korovin, Iakov, [Preprint] 2019. “An Effective Hybrid Approach for Optimizing the Learning Process of Multi-Layer Neural Networks”

[13] Open Source, “Keras: The Python Deep Learning library” in “Keras.io”, [Online Documentation] 2018. Available online reference, https://keras.io/getting-started/sequential-model-guide/ [Accessed: December, 2018]

[14] E. Schapire, Robert, Microsoft Research, 2013. “Explaining AdaBoost” 10.1007/978-3-642-41136-6_5.

[15] Baig, Mubasher&Awais, Mian& El-Alfy, El-Sayed, 2017, “AdaBoost-Based Artificial Neural Network Learning. Neurocomputing” 248. 10.1016/j.neucom.2017.02.077