SAJSET
ISSN: 2992-3581

Savannah Journal of Science and Engineering Technology
Volume: 02; Issue: 06; Pages: 341-348; 2024.
www.sajsetjournal.com.ng

DOI: xx.xxxx.x/sajset-01-2024-0049

SAJSET
ISSN: 2992-3581

Integration of modified pre-trained VGGNet with inception modules and vanishing gradient problem solution-based activation functions for rice disease classification

[1]Abubakar, B., [2]Adekale, A. D., [3]Adamu, N. M., [4]Sunusi, M. A., [2]Adebiyi, R. F. and [2]Bello-Salau, H.
[1]*Department of Computer Engineering, Federal Polytechnic Bali, Taraba State, Nigeria*
[2]*Department of Computer Engineering, Ahmadu Bello University, Zaria, Kaduna State, Nigeria*
[3]*Department of Electrical and Electronic Engineering, Federal Polytechnic Kabo, Kano State, Nigeria*
[4]*Department of Electrical and Electronic Engineering, Federal Polytechnic Bali, Taraba State, Nigeria*

Paper History
Received: 30th Nov., 2024
Accepted: 12th, Dec., 2024
Published: December, 2024

Corresponding author
Abubakar, B.
senator029@gmail.com

Abstract:
Rice is one of the crops grown and consumed in almost every nation. However, the crop is challenged by variety of diseases which results in degraded and low quality of the end produce. This calls for automation of disease identification and classification. Recently, Deep Learning (DL) models got much attention due to its remarkable performance in classification of plants disease. However, three factors influence the performance of DL models. These factors include the nature of dataset used in training, the architecture of the model, and the type of activation function used in the model. This work digs into these three factors; first, a new high quality dataset for rice disease was collected over the course of this research work. Secondly, VGG-16 pre-trained model was modified to optimise the performance of the model on the new dataset for training. Inception modules were then added and Fully Connected (FC) layer was replaced with an average pooling layer. The Rectified Linear Unit (ReLU), which is the traditional activation function of DL models, faces the dying neurons problem. Different activation functions that solve the problem of dying neurons were applied to the modified architecture to determine which function performs best on the rice dataset. The model utilizing the Exponential Linear Unit (ELU) activation function achieved remarkable accuracies, with a training accuracy of about 92.0%. Such performance holds significant promise for early disease detection in agricultural settings, potentially leading to increased farm productivity.

Keywords: *Activation, Architecture, Batch-Normalisation, Convolution, Epoch, ImageNet, Swish*

1. Introduction

Rice, been one of the highly grown crops globally, feeds billions of people and serve as source of revenue for millions of farmers [1, 2]. This crop is faced with many forms of diseases that affect the quality and quantity of the farm produce thereby causing several financial losses to the farmers [3]. This in turn, increase the cost input in treating the diseases, thus, raises food prices and aggravates food insecurity [2]. Most of the time, these diseases have severe impact on the crops because farmers find it difficult to identify these diseases at early stage for timely intervention. Recently, Deep Learning (DL) models have become apparent means for the identification and classification of these diseases simply for the reason of their high accuracy [3, 4]. The practical implementation of the DL models on handheld devices is easy, and can be efficiently deployed on farms. This makes early detection of the diseases at early stage easy.

However, the performance of these DL models depends on three factors. Namely: The model architecture, choice of activation functions and the nature of dataset used during the training [5]. Some literatures assessed how these three factors influence the model's performance. For example, Hiroki *et al.,* [6] fine-tuned the VGGNET Convolutional neural network sigmoid function as its classifier. The model had eight convolutional neural network layers and two fully connected layers. The convolutional layers have lesser feature maps when compared to the original VGGNET, and Batch normalization was performed on all fully connected layers. The model achieved an accuracy of 95.5% which is nine times faster compared to the original VGGNET architecture [6]. Ma *et al.,* [7] proposed a model for recognition of four cucumber diseases based on deep convolutional neural network. The images used in the study were resized to 20x20x3 due to the fact that the regions for the symptoms were small. To show the effect of size of data on the model, balanced and unbalanced dataset was used. The accuracy for the balanced and unbalanced dataset classification was 93.4% and 92.2% [7]. Chen *et al.* [8]

proposed an approach that involved the substitution of VGGNET last convolutional layers with extended convolution Layer of 3 × 3 × 512, Batch Normalization (BN) was added and ReLU function was also substituted with swish function. The authors also added two Inception module layers and a maxpooling layer substituted the fully connected layers. The Model was based on rice disease, maize datasets. The rice dataset had higher result accuracy, with average of 92.0% [8]. Lu *et al.,* [9] addressed the persistent issue of vanishing and exploding gradients in the training of deep neural networks. For instance, the authors note that batch normalization can even exacerbate the issue and has limitations based on mini-batch sizes. Similarly, neural networks with Scaled Exponential Linear Units (SELU) can fail to fully prevent vanishing/exploding gradients, and dynamical isometry may constrain neural network functionality too closely to linear models. The paper proposed three solutions. First it proposed a new type of neural network architecture that incorporates orthogonal weight matrices and a novel class of activation functions known as Gaussian–Poincaré normalized functions, which can be adapted from many common activation functions. Secondly, they demonstrated that with a sufficiently large layer width, the vanishing/exploding gradients issue can be resolved with high probability. Finally, the paper presented experimental validation using both synthetic and real-world data to confirm the effectiveness of their proposed solution in maintaining nonlinearity in deep neural networks while addressing the gradient problem [9].

Largely, the literatures modified the architectures of the DL models, proposed their activation functions or adopted one or the other functions. These models developed in the literatures were then compared with one activation function or two other functions that solved the Vanishing Gradient (VG) problem. Most of the datasets used were proprietary and cannot be publicly accessed. This research conforms to these literatures pattern, where Visual Geometry Group (VGG-16) pre-trained model is modified, batch normalisation and four different activations were successively added. Finally, two inception modules were added, a global pooling layer replaced the fully connected layers and Softmax function was used as the classifier. The paper also curates a Rice dataset for the duration of the research. The VGG-16 pre-trained model was chosen over VGG-19 due to its superior performance observed in the course of this study.

1.1 Fundamental Concepts

1.1.1 Artificial Neural Network (ANN)

These are a computational algorithm/model motivated by the biological artificial neural networks designed to simulate the learning ability and decision-making processes of the human brain [10, 11]. The ANN systems adjust the weights of their neurons based on the labelled training data without any specified task rule for classification [11]. The systems then try to classify new instances based on the learned weights. The ANN normally has multiple layers ranging from the input layer which neurons equal to feature vector length and then one or more hidden layers, then the output layer which will have number of neurons equal to the number of classes [12-14]. The number of hidden layers and number of neurons in every hidden layer cannot be computed in advance as these depend on the specific problem set [15].

1.1.2 Convolutional neural network (CNN)

The CNN is defined as a neural network that performs a convolution operation in one or more of its layers [16]. Convolutional neural networks are the basis of deep learning. It takes raw image as input and extracts high features. The three layers found in CNN are the convolution layers, the pooling layers and the fully connected layers. The layer after the input layer is the convolution layers where kernels of different sizes are convolved on the spatial pixels of the image to learn the image's intrinsic features. The convolution operation is "the dot-product operation between a grid-structure set of weights and similar grid-structured inputs drawn from different spatial localities (such as image) in the input volume [17].

The ReLU is an activation function found in the CNN just as it is also found in the traditional neural networks [18]. After the convolution, an activation function is applied, and then the pooling layer. The pooling layers reduce the spatial sizes of the images by one of maxpooling, minpooling or random pooling. The pooling operation runs on lesser size on spatial locations and generates another layer of same depth size as in contrast to the convolution operation filters. Pooling operation is done on every feature map generated by the activation map, and produces the same number of feature maps. Therefore, the pooling operation does not change the number of feature maps.

After several numbers of the convolution and pooling is the Fully Connected (FC) layer that extracts deep features for classification. The fully connected layer architecture works just like a conventional feed forward network as output of the previous layer is fed forward the next layer. The number of states in the fully connected layer corresponds to the number of feature maps generated in a spatial layer just before the first fully connected layer. A CNN can have more than one FC layer when there is requirement for more computational power in the end, and then the output of the last FC layer is fed to Softmax classification layer [19].

It can be generalized that, the higher the number of model's parameters, the higher its capacity, and the larger the amount of data they will need in order to have high power on generalization of the unseen test data [20]. The key success attributed to neural networks is that, simple features are put stacked together which builds unto a complex feature. These features are used repetitively as building blocks of a complex feature. The pre-trained models turn out to be suitable when utilized in a careful manner. [20].

1.1.3 Pre-Trained DL Models

To sustain the requirements of deep neural networks on training large datasets, deep learning basically uses the pre-trained networks [8]. Two transfer learning approaches

mostly used are the fine-tuning of specified pre-trained model and use of specific pre-trained network directly [21]. The advantage of the second approach is that there will be no need of training the deep network where extracted features of the pre-trained model is operated into the current image analysis pipelines [22].

Often, deep learning models are too big or under-used when there is only a small volume of data. In the scenario where the data is small, it will be better to enhance the shallow machine learning models where the result will even be easier to translate. The neural networks have more flexibility to construct a complex function, where it adds neurons to the computational graph where the amount of data increases [20].

GoogleNet introduced novel inception architecture, characterized by a "network within a network" design. Since its inception, the architecture has garnered significant attention from researchers. The key to its performance lies in its ability to capture critical information at various layers, enhancing its scalability. The GoogleNet consists of 22 inception layers with varying filter sizes, and it replaces the fully connected layers with average pooling in its structure. Although it achieved a 6.7% error rate and won the 2014 ILSVRC, the GoogleNet architecture is more complex than the runner-up, VGGNet.

The Visual Geometry Group (VGG) secured second place in the 2014 ImageNet Large Scale Visual Recognition Challenge (ILSVRC), achieving an error rate of 7.3%. The VGG architecture has been tested with configurations such as VGG19 and VGG16, consisting of 19 and 16 layers, respectively, and trained on the ImageNet dataset. While the most effective configurations include layers 16 to 19, the number of layers can range from 11 to 19. A notable feature of the VGG architecture is its reduction in filter sizes while simultaneously increasing the depth of the network [20]. VGG employs a cascaded network structure, comprising two to three convolutional layers followed by a pooling layer. The convolutional layers utilize small 3x3 convolutional filters [8].

AlexNet emerged as the winner of the 2012 ImageNet Large Scale Visual Recognition Challenge (ILSVRC). Its internal architecture features two parallel preprocessing pipelines, enabling the use of two GPUs working collaboratively to accelerate model training through faster processing and shared memory. AlexNet accepts an input of size 224x224x3 and employs 96 filters of size 11x11x3 in its first layer. This layer utilizes four strides, resulting in an output size of 55x55x96, followed by max-pooling. Each convolutional layer applies a ReLU activation function, followed by response normalization and then max-pooling. Although the convolutional layers utilize different kernel sizes, all pooling layers use a 3x3 kernel with 2 strides. The fully connected layers consist of 4096 neurons, representing a 4096-dimensional feature space for the input image. The final layer of AlexNet is a softmax layer, which performs 1000-class classification [20].

1.1.4 Vanishing Gradient (VG)

In the training of deep learning networks especially those networks that have large number of layers, a common problem encountered is the gradients becoming too small during the training, this is called the vanishing gradient. This affects the learning rate as the weights of the layers are updated slowly during the back propagation and this leads to the network not learning significant features from the data [23]. In addressing this problem, many activation functions were proposed and the Rectified Linear Unit (ReLU) performed remarkably well in solving the vanishing gradient problem. Nevertheless, even with its performance there are functions proposed which perform better than the ReLU in mitigating the dying neurons problem which is common in the ReLU, these include the Leaky Rectified Linear Unit (LeakyReLU), Swish, Exponential Linear Unit (ELU), Scaled Exponential Linear Unit (SELU), and the Parametric Rectified Linear Unit (PReLU).

Rectified Linear Unit (ReLU) is a non-linear activation function that effectively handles complex data, including features that may not be correlated. It outperforms linear functions, which tend to struggle with complex data. ReLU is computationally efficient because it does not process negative input values but instead maps them to zero, thereby reducing computational complexity. It only accepts input values of zero or greater, which helps prevent the gradient from vanishing as quickly as in other activation functions. This property has significantly contributed to the development and improved performance of deep neural networks. However, despite its advantages, ReLU suffers from the "dying neuron" problem, where negative neurons become inactive by being mapped to zero throughout training, which can impair model performance. The mathematical representation of ReLU is given in equation 1 [24].

$$f(x) = max(0, x) \tag{1}$$

Leaky Rectified Linear Unit (LeakyReLU) is a variant of the ReLU activation function designed to address the issue of dying neurons during training. In ReLU, negative neurons are rendered inactive once they are mapped to zero, preventing them from contributing to the training process. In contrast, LeakyReLU allows for a small positive value for negative inputs, which results in a small gradient for these negative inputs and keeps the neuron active during training. The formula for LeakyReLU is provided in equation 2 [25]. However, the introduction of this small positive value makes LeakyReLU computationally more expensive than ReLU.

$$f(x) = \begin{cases} x & if\ x \geq 0 \\ \alpha . x & if\ x < 0 \end{cases} \tag{2}$$

The Swish function is similar to LeakyReLU in that it introduces small positive values for negative neurons, but it differs by using a sigmoid function, as described in equation 3. The Swish function has an activation range between +3 and -3, allowing for a broader range of inputs to the network. While the Swish function has shown superior performance in deep neural networks compared to the traditional ReLU, its effectiveness may vary depending on the architecture and the characteristics of the dataset used during training. Notably, Swish is computationally more expensive than ReLU, yet it does not introduce any

additional learning parameters, contributing to its simplicity. The formula for the Swish activation is provided in equation 3 [24].

$$swish(x) = x.sigmoid(x) \quad (3)$$

The Exponential Linear Unit (ELU) addresses the dying neurons issue observed in ReLU by employing a different approach. Instead of simply zeroing out negative values, ELU saturates them to a negative value determined by a hyperparameter, which ensures non-zero gradients for negative inputs. Additionally, ELU is zero-centered, which aids in the faster convergence of the network during training. The formula for ELU is provided in equation 4 [26].

$$ELU(x) = \begin{cases} x & if\ x \geq 0 \\ \alpha.(\exp(x) - 1) & if\ x < 0 \end{cases} \quad (4)$$

1.1.5 Batch Normalization (BN)

Sometimes activation gradients in proceeding layers either reduces or increases exponentially in magnitude, a problem called the "exploding gradient". To tackle this issue, recently the batch normalization is developed. The batch normalization adds normalization layers between the hidden layers to withstand the exploding gradient problem, by forming to some degree, a similar variance [20]. To make sure the inputs of layers always fall within the same range, even if previous layers are updated, Batch normalization is used; this decreases the number of training iterations and virtually helps to regularize the model [27].

1.1.6 ImageNet Database

This is large repository of approximately 14 million images of 1000 different sets of objects, this makes it almost possible for it to cover up most of things we come across in our daily life activities. It's a point of reference for most neural networks, for the reason that it is usually available and its results are known because the yearly "ImageNet Large Scale Visual Recognition Challenge (ILSVRC)" is based on the database. The competition is highly regarded in the research community of computer vision. ImageNet database is also important for the fact that it is large and diverse enough for the representation of the visual conceptions within the area of classical machine learning which employs optimization and gradient-descent methods, parameterized learning models. Examples include logistic regression, linear regression, support vector machines, matrix factorization, and dimensionality reduction [16].

The ImageNet is employed in pre-training of Convolutional Neural Networks, and give a prepared-to-use features. These pre-trained models can be downloaded, and an image dataset can be made to pass through the pre-trained models to create a multidimensional representation [20].

1.1.7 Augmentation

Many forms of augmentations do not need much mathematical computations, thus these augmentations can be generated while training the network. Different augmentations on same dataset helps network identify same image in different forms, e.g. same image with different intensities, orientations, flips, etc. [20].

2. Materials and methods

This section presents the materials and methods that were used in carrying out of this piece research work.

2.1 Materials

Google Colab Pro (R) was used for the training the Transfer learning-based model using modified VGGNet and inception modules for Diminishing gradient-based activation functions. Two datasets of Rice were used. The first being the curated Rice dataset collected within the Federal Polytechnic Bali Farms and the second from the plant village dataset which can be found on www.kaggle.com as reported in Koklu, *et al.* [28].

2.1.2 Curation of Rice diseases Dataset

At some point in this study, plant disease experts were consulted. Of important to note is the experts from the Institute of Agricultural Research (IAR) Zaria. Some of the publicly available dataset of the plant village were proven to be either to be mixed and some wrongly classified. With guidance of the experts, a new plant disease dataset was curated for the Rice Crop. The Canon 100 digital camera was used for the data capturing of the Rice Crop diseases images. Three classes were captured, namely the leave spot, leave blight and the healthy rice classes. It was ensured that every class has at least 100 images each, after that the images were then pre-processed, including background removal, sharpening and image resizing (224x224). The images were then augmented using various augmentation techniques, such as rotation, vertical and horizontal flip, shear, zoom, among others. After augmenting each of the class images, the resultant image of each class were 500 images. Figure 1 shows the sample images for the three classes.

(a) Healthy (b) Blight (c) Spot

Figure 1: Sample images for the Collected Rice Dataset

2.1.3 Modification of VGG-16 architecture

The VGGNet fifth convolutional layer and other layers beyond were removed and replaced with convolutional layer of dimension 3x3x512 with one of the following activation functions ELU, SELU, PReLU, Swish, or LeakyReLU functions. After that two inception layers were added with each having a feature map of 256, the global pooling layer and Softmax activation functions were added as the top layers. The new generated network after compilation had approximately 10.4 million parameters of which only about 2.7 million are trainable. The trainable parameters are the weights of the network which are updated to learn the features and adjusted to suit the task at hand. To lower the errors between the true labels and the predicted labels, these weights are adjusted during the training process using back propagation. The non-trainable parameters are the weights that are freezed and are not updated during the training but rather adopted from the earlier pre-trained model. This is due to the fact that mostly the computational resources and dataset at hand usually are very limited compared to the originally pre-trained computational resources and dataset. The adoption of the non-learnable parameters also helps in lowering the chances of overfitting of the trained model. The learnable parameters are from the layers added while non-learnable parameters are from the earlier pre-trained VGGNet architecture layers adopted.

2.1.4 Application of the generated model

The new model generated applied on the rice dataset, using each of the LeakyReLU, PReLU, ELU and Swish functions to determine which function has the best performance. The results for ten and thirty epochs were chosen for documentation for easy comparison as adopted by Chen, *et al.* [8]. Images in each class were augmented to 500 images, from the original 100 images curated. In total for maize, 1500 images were used, of which 70 and 30 percent of the images were used for training and testing respectively. The general flowchart of the model is given in Figure 2.

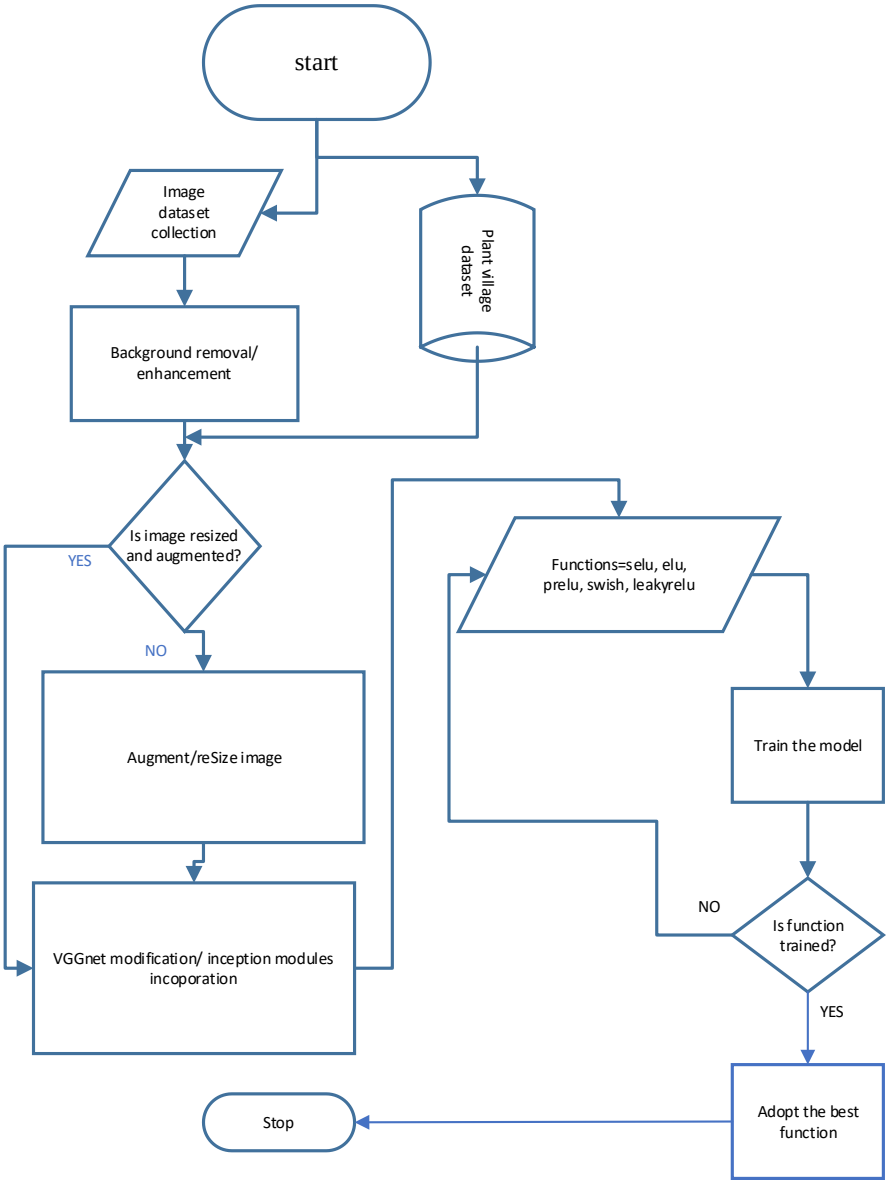


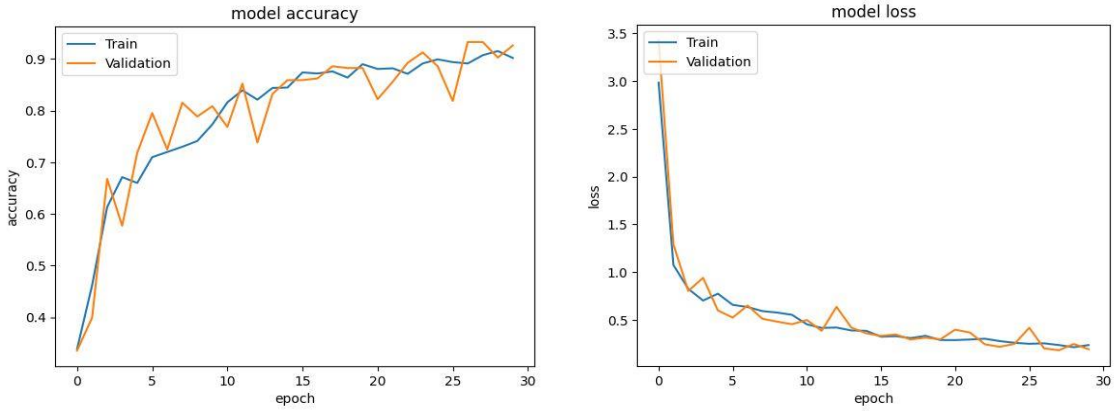
Figure 2: Flowchart Diagram of the Developed Model

3. Results and Discussion

In this section, the outcomes of the model using two datasets were presented and analysed: one comprising of the curated rice plants and other sourced from a publicly available database on www.kaggle.com as reported in Koklu, *et al.* [28] called the plant village datasets. The results encompass evaluations of multiple activation functions, including ELU, SELU, PReLU, LeakyReLU, and swish functions, all designed to mitigate the issue of vanishing gradients that can hamper model performance during training. At first, the findings from the classification curated Rice diseases dataset were discussed, followed by an examination of the results from the classification of Plant Village Rice diseases dataset.

3.1 Performance of Model on Curated Rice

The curated rice dataset was analysed across three classes: healthy, rice spot, and rice blight disease. The acquisition of the rice images posed challenges due to the predominantly folded nature of rice leaves. Consequently, some of the image quality in the curated dataset was notably inferior. The ELU activation function emerged as the top performer after 30 epochs, achieving a training and validation accuracy of 91.95% and 92.95%, with corresponding losses of 0.2019 and 0.1687 respectively. However, examining the gradient graph as in Figure 3a revealed that the model's convergence was not consistent even at epoch 20, possibly due to the image quality and the intricate nature of rice diseases. Accuracy and loss graphs are shown in Figures 3a and 3b respectively. The second-best performing function for the curated rice dataset was found to be PReLU, yielding a training accuracy of 91.87% as shown in Table 1. This underscores the influence of model architecture, activation functions, and dataset characteristics on performance, as demonstrated by the superior performance of the swish function for maize dataset in a yet to be published paper of this research. A comprehensive summary of the performance of the five functions investigated for the curated rice dataset is provided in Table 1.



(a) Accuracy (b) Loss

Figure 3: Accuracy and Loss Graphs of ELU for Curated Rice Dataset

Table 1: Summary of Activation Function Performances for Curated Rice Dataset

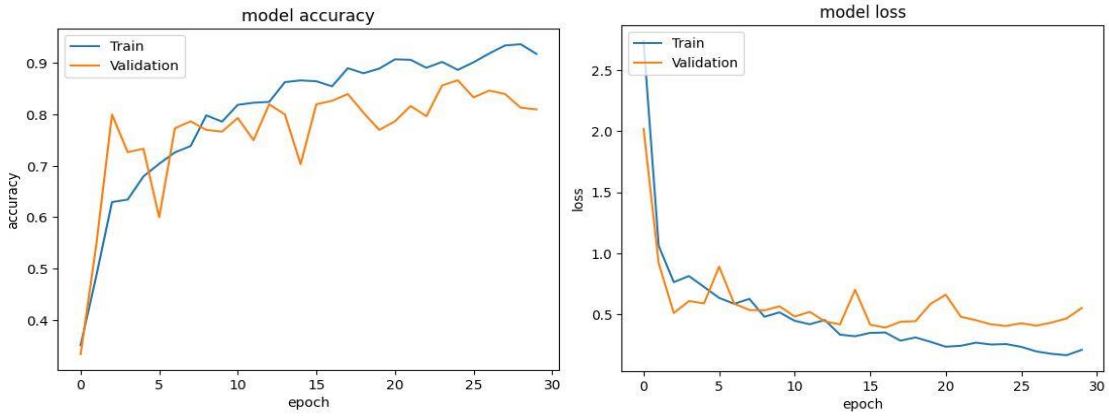
S/N	Function	Epoch 10				Epoch 30				Testing
		Training accuracy	Validation accuracy	Training Loss	Validation Loss	Training accuracy	Validation accuracy	Training Loss	Validation Loss	
1.	Elu	83.07	79.53	0.4165	0.4636	91.95	92.95	0.2019	0.1687	80.43
2.	Leaky-ReLU	77.53	84.56	0.5350	0.4135	91.13	89.26	0.2446	0.2394	81.05
3.	PreLU	82.13	77.53	0.4296	0.5398	91.87	91.33	0.2303	0.1882	79.99
4.	Swish	79.73	80.54	0.4606	0.4640	91.73	91.61	0.2099	0.1944	80.56
5	Selu	76.60	75.17	0.5678	0.5969	90.07	90.94	0.2393	0.1912	79.33

3.2 Results for plant village rice dataset

The Plant Village disease rice dataset comprises three classes: brown leaf spot, leaf smut, and leaf blight. Following consultation from the IAR, it was discovered that some diseases in the Plant Village dataset were misclassified. Nevertheless, for analytical purposes, the model was trained using the available dataset. The ELU activation function demonstrated exceptional performance, surpassing other functions with a training accuracy of 92.13%. The model exhibited better convergence compared to the curated rice dataset, attributed to the superior quality of images in the Plant Village dataset shown in Figure 4a, highlighting the significant impact of image quality on model performance. Despite the poor image quality in the curated rice dataset, the model performed commendably on this dataset, notwithstanding slightly less effective than on the Plant Village dataset, with a difference of 0.26% for the ELU activation function, which yielded the best performance across both datasets analysed in this study. Following ELU, the Swish function demonstrated the next best performance, achieving an accuracy of 91.80% as can be seen in Table 2.

Based on the outcomes observed across the two datasets investigated, the ELU activation function has demonstrated resilience across both rice disease datasets analysed in this study. This robust performance could be

credited to the inherent advantage of the ELU activation function in effectively handling networks with numerous layers, as well as its capability to delineate meaningful features, particularly in scenarios where negative activations are present. The summary of the performances of the plant village dataset for rice is given in Table 2 at epochs 10 and 30.



(a) Accuracy (b) Loss

Figure 4: Accuracy and Loss Graphs of ELU for Plant Village Rice Dataset

Table 2: Summary of Function Performances for Plant Village Rice Dataset

S/N	Function	Epoch 10				Epoch 30				Testing
		Training accuracy	Validation accuracy	Training Loss	Validation Loss	Training accuracy	Validation accuracy	Training Loss	Validation Loss	
1.	Elu	80.33	73.67	0.4876	0.5644	92.13	87.33	0.2138	0.3424	79.87
2.	Leaky-ReLU	74.42	71.00	0.6079	0.6959	90.67	89.60	0.2345	0.1950	75.30
3.	PreLU	75.41	74.00	0.5814	0.6656	87.13	83.00	0.3201	0.5222	71.93
4.	Swish	78.61	76.67	0.5165	0.5660	91.80	81.00	0.2089	0.5521	77.65
5	Selu	74.64	74.17	0.6087	0.6237	90.00	69.00	0.2545	0.8028	74.41

4. Conclusion

In this study, activation functions were incorporated into a hybrid model comprising VGGNet and Inception Net, aimed at addressing the vanishing gradient problem encountered during training in deep learning models. Rice plant was chosen as focal point due to its cultivation prevalence in Northern Nigeria. The curated dataset was collected from the Federal Polytechnic Bali Farms land, supplemented by the Plant Village dataset for evaluating the model's performance. Results indicate that the ELU activation function outperformed the Swish activation function, as well as three other activation functions specifically designed to mitigate the vanishing gradient problem, as described by Chen, *et al.* [8]. The model utilizing the ELU activation function achieved remarkable accuracies, with a training accuracy of 91.95% and a validation accuracy of 92.95%. Such high performance holds significant promise for early disease detection in agricultural settings, potentially leading to increased farm productivity.

Recommendation

In the course of this research, only the performance of different activation functions was considered. However some of these functions add more complexities to the models, increasing the model's hyper parameters and computational resources needed to train the model. It is recommended in further works, to examine how these activation functions affects the general model complexity and training time taken for each function.

Acknowledgement

This work is supported by the Tertiary Education Trust Fund (TETFUND) under the Institution Based Research (IBR) with ID. (TETF/DRandD/CE/POLY/BALI/TARABA/IBR/2022/VOL.1) . The Author would also like to express his gratitude to the Research and Development committee of Federal Polytechnic Bali for their insightful comments and ideas that greatly contributed to the development of this study.

References

[1]. Singh, G. and Singh, R. (2023). *Deep Learning-based Rice Leaf Disease Diagnosis using Convolutional Neural Networks*. 2023 International Conference on Sustainable Computing and Smart Systems (ICSCSS), Coimbatore, India, pp. 107-111.

[2]. Gogoi, M., Kumar V., Begum, S. A., Sharma, N. and Kant, S., (2023). Classification and detection of rice diseases using a 3-stage CNN architecture with transfer learning approach*, Agriculture*, 13(8), 1505-1515.

[3]. Venkatesh, N., Sarkar, S., Sunil, A., Priya, T. G., Aarthi, R. and Devi, K. M., (2023). *Deep Learning-based Identification and Classification of Rice Plant Diseases for Precision Agriculture*, 2023 2nd

International Conference on Automation, Computing and Renewable Systems (ICACRS), Pudukkottai, India, pp. 1027-1032.

[4]. Li, Y., Chen, X., Yin, L. and Hu, Y., (2024). Deep Learning-Based Methods for Multi-Class Rice Disease Detection Using Plant Images*, Agronomy*, 14(9), 1879-1786.

[5]. Seelwal, P., Dhiman, P., Gulzar, Y., Kaur, A., Wadhwa, S. and Onn, C. W., (2024). A systematic review of deep learning applications for rice disease diagnosis: current trends and future directions*, Frontiers in Computer Science*, 6, 1452961

[6]. Hiroki, T. A. N. I., Kotani, R., Kagiwada, S., Hiroyuki, U. G. A., and Iyatomi, H. (2018, October). Diagnosis of multiple cucumber infections with convolutional neural networks. In *2018 IEEE Applied Imagery Pattern Recognition Workshop (AIPR), Washington DC, USA,* pp. 1-4.

[7]. Ma, J., Du, K., Zheng, F., Zhang, L., Gong, Z. and Sun, Z., (2018). A recognition method for cucumber diseases using leaf symptom images based on deep convolutional neural network*, Computers and electronics in agriculture*, 154, 18-24.

[8]. Chen, J., Chen, J., Zhang, D., Sun, Y. and Nanehkaran, Y. A., (2020). *Using deep transfer learning for image-based plant disease identification.* Computers and Electronics in Agriculture, 173, 105393-105401.

[9]. Lu, Y., Gould, S. and Ajanthan, T., (2023). Bidirectionally self-normalizing neural networks*, Neural Networks*, 167, 283-291.

[10]. Anandhi, D. F. R. and Sathiamoorthy, S, (2023). *Deep learning based automated rice plant disease recognition and classification model*. First International Conference on Advances in Electrical, Electronics and Computational Intelligence (ICAEECI), Tiruchengode, India, pp. 1-6.

[11]. Hasan, M. M., Rahman, T., Uddin, A. S., Galib, S. M., Akhond, M. R., Uddin, M. J. and Hossain, M. A., (2023). *Enhancing rice crop management: Disease classification using convolutional neural networks and mobile application integration. Agriculture*, 13(8), 1549-1555.

[12]. Kaur, P., Harnal, S., Gautam, V., Singh, M. P. and Singh, S. P., (2023). A novel transfer deep learning method for detection and classification of plant leaf disease*, Journal of Ambient Intelligence and Humanized Computing*, 14(9), 12407-12424.

[13]. Shoaib, M., Shah, B., Ei-Sappagh, S., Ali, A., Ullah, A., Alenezi, F., Ali, F. (2023). An advanced deep learning models-based plant disease detection: A review of recent research*, Frontiers in Plant Science*, 14, 1158933.

[14]. Balafas, V., Karantoumanis, E., Louta, M. and Ploskas, N., (2023). *Machine learning and deep learning for plant disease classification and detection.* IEEE Access, 11, 114352 - 114377.

[15]. Jibrin, B., Bello-Salau, H., Umoh, I. J., Onumanyi, A. J., Salawudeen, A. T. and Yahaya, B., (2021). Development of hybrid automatic segmentation technique of a single leaf from overlapping leaves image*, Journal of ICT Research and Applications*, 14(3), 257-273.

[16]. O'Shea Keiron, N. R. (2015). An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*.

[17]. Saleem, M. H., Potgieter, J. and Arif, K. M., (2020). *Plant disease classification: A comparative evaluation of convolutional neural networks and deep learning optimizers.* Plants, 9(10), 1319-1331.

[18]. Padshetty, S. A. (2023). Leaky ReLU-ResNet for Plant Leaf Disease Detection: A Deep Learning Approach*, Engineering Proceedings*, 59(1), 39-48.

[19]. Yadav, S., Sengar, N., Singh, A., Singh, A. and Dutta, M. K., (2021). *Identification of disease using deep learning and evaluation of bacteriosis in peach leaf,* Ecological Informatics, 61, 101247-101255.

[20]. Aggarwal, C. C. (2018). *Neural networks and deep learning,* Cham, springer, Vol. 10, No. 978, p. 3**.**

[21]. Cap, Q., Suwa, K., Fujita, E., Uga, H., Kagiwada, S. and Iyatomi, H., (2018). An end-to-end practical plant disease diagnosis system for wide-angle cucumber images, *International Journal of Engineering and Technology*, 7(4.11), 106-111.

[22]. Gao, Z., Luo, Z., Zhang, W., Lv, Z. and Xu, Y., (2020). Deep learning application in plant stress imaging: a review*, AgriEngineering*, 2(3), 430-446.

[23]. Pandey, G. K. and Srivastava, S., (2023). *ResNet-18 comparative analysis of various activation functions for image classification*. *2023 International Conference on Inventive Computation Technologies (ICICT), Lalitpur, Nepal.* pp. 595-601.

[24]. Harshanand, B. and Sangaiah, A. K., (2020). Comprehensive analysis of deep learning methodology in classification of leukocytes and enhancement using swish activation units*, Mobile networks and applications*, 25, 2302-2320.

[25]. Nair V. and Hinton, G. E., (2010). *Rectified linear units improve restricted boltzmann machines*, Proceedings of the 27th international conference on machine learning (ICML-10), Canada, pp. 807-814.

[26]. Alkhouly, A. A., Mohammed, A. and Hefny, H. A., (2021). Improving the performance of deep neural networks using two proposed activation functions*, IEEE Access*, 9, 82249-82271.

[27]. Chen, L. and Yuan, Y., (2018). *Agricultural disease image dataset for disease identification based on machine learning*. International Conference on Big Scientific Data Management, Beijing, China. pp. 263-274.

[28]. Koklu, M., Cinar, I. and Taspinar, Y. S. (2021). Classification of rice varieties with deep learning methods, *Computers and Electronics in Agriculture*, 187, 106285. https://doi.org/10.1016/j.compag.2021.106285