ISSN 2278-3091

Volume 10, No.1, January - February 2021

International Journal of Advanced Trends in Computer Science and Engineering

Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse191012021.pdf

https://doi.org/10.30534/ijatcse/2021/191012021

An overview of activation functions used in Neuronal Networks: Application for plant disease diagnosis

E. RAOUHI, M. LACHGAR, A. KARTIT

LTI Laboratory,ENSA, University Chouaib Doukkali, El Jadida, Morocco

raouhi@gmail.com, lachgar.m@gmail.com, alikartit@gmail.com

ABSTRACT:

Deep learning techniques, particularly Convolutional Neural Networks (CNNs), have led to significant progress in image processing. Many applications in automatic identification of plant diseases have been developed. This work adopts a new approach that focuses on studying a relevant parameter that make a significant impact on the performance of CNNs, namely, the variants of activation function, particularly the most famous used functions and their influence on the model's performance and accuracy. We will also present the different types of activation functions, which are also called transfer functions. Then, and through a case study application to the plant disease detection, we will have the opportunity to compare the results of these different functions with a graphical presentation using evaluation metrics, such as accuracy functions and loss functions as Binary Cross-Entropy. The training of the models was carried out using a free accessible database of 20,639 photographs, taken both in the laboratory and in real conditions from the crop fields. The data includes three plant species in fifteen distinct classes of combinations [plant, disease], including some healthy plants.

Key words: Activation functions, Convolutional Neuronal Networks, Deep Learning, Plant diseases

1. INTRODUCTION

One of the major challenges that the agricultural factor has to face almost inevitably is the different types of diseases that affect crops from time to time. A lot of these diseases are caused by pests. Several efforts to prevent plant diseases have been in practice over time [1-2]. However, excessive use of these chemicals has started to show its adverse effects. Contrary to the former, accurate and timely detection of plant diseases is recognized as crucial for precision agriculture with neuronal architecture [3], so the performance of the latter becomes a challenge considering the factor influencer's [4].
When making the neural architecture or a deep neural network to perform well in practice, activation functions are needed. However, their choice also affects the network in terms of optimization to retrieve better results [5]. One such decision is the type of layer that composes the neural network, particularly involving the type of activation functions that will be implemented as hidden layers. Several activation functions have been introduced in machine learning for many practical applications. They are used to solve nonlinear complex problems with the real data to train the network and to produce the useful results [6].
The rest of the paper is organized as follows. Firstly, in section 2, we present the most recent related works. Then in section 3, we provide the theoretical overview of the activation functions used in the context of our application in order to identify the optimized choices for a better convergence of the neuronal network in the case of use of these experiments. After, in section 4, an analysis and a comparison of the performance of five activation functions, mainly ReLU, Leaky ReLU, PReLU, Sigmoid and Tanh are done on the convolutional neural network. We use a dataset of 20,639 images of diseased and healthy plant leaves collected under controlled conditions to see how overall accuracies vary accordingly on different activation functions in the network. Based on our results from that analysis, we make the prediction by giving input images onto the trained CNN with different activation functions on the layers and conclude which are best suited for different type of layers for CNN: hidden or output one's case of plant disease detection.

2. RELATED WORKS

This section present the related work carried out in this perspective. First, the authors on [7], conclude that the ReLU and the Softmax are the dormant activation functions used in practical Deep Learning applications.

Next in [8], the authors of the article note that the Softmax function is generally used at the level of the output layer, on the other hand the reread function is chosen as a hidden layer in the majority of deep neuron network architectures, this choice is motivated by the results obtained and the performance released.

Then, in [9], the sigmoid function will transform an input value into an output between 0 and 1. Any input larger than 1 will be transformed to 1, and inputs smaller than 0 will be transformed to 0. In the case where it is applied in a neural network, the saturation varies between 1 and 0 and generates a sensitivity to change at the midpoint. in this logic it explains the training behavior the of a neural network, especially one with many layers, it becomes increasingly more difficult for the neural network to adapt, considering its weights, and thus improve performance. The sigmoid function can also cause neural networks to suffer from the vanishing gradient problem since error is back propagated through the layers and decreases dramatically with each hidden layer [10].

Next, on the demonstration detailed on [11], the Multilayer Perceptron Neural Network (MLP) has been configured on a series of activation functions with the aim of evaluating their behavior. These are: Identity, Tahn, Logistics, Exponential and Sine. The results obtained show that the performance of a multilayer perceptron can be affected by the choice of an activation function. Tahn's and logistics activation functions far surpassed other models.

After, the authors of [12], present a demonstration of a new function allowing adaptive scaling based on a principle of attenuation of changes in covariates during training, and that the observed advantages in terms of performance of Adaptive Blending Units (ABUs) also rely heavily on the ability of the activation function to adapt during training.

So, the type of activation function associate to the number of hidden layers are very important to evaluate the performance of neural networks. Considering this the authors on [13], construct many artificial neural network models using different covariations of hyper parameters and evaluate their performance. The results obtained proved that neural networks with scaled exponential linear units and five hidden layers provide the best performance

Finally, result present in [14] shows that the function to be used in the case of dead neurons is Leaky ReLU, moreover the ReLU function is not optimized to be used in the hidden layers but remains interesting for it to be used at the start of the simulation. Otherwise Parametric ReLU function has a small slope, which can also avoid the problem of ReLU death. knowing that for negative inputs an output slope presents a learning parameter while in the case of the Leaky ReLU function, the latter is rather a hyper parameter. [15].

Our contribution consists in challenging the performance of prediction model using a comparison of the most regularization methods namely: ReLU, Leaky ReLU, PReLU, Sigmoid and Tanh applied to plant disease detection with the aim of optimizing the crop production.

3. ACTIVATION FUNCTIONS

In this theoretical section, the activation functions present in the literature will be detailed. their mathematical presentation, if they are the right choice to use, analyze in depth the weaknesses and strengths of each function in order to take a first step in a long road of understanding the neural network but the mathematical analysis of these nonlinearities can provide an appropriate basis and intuition for what could happen after. Inside neural networks, you have to understand the activation functions not only as a tool to make Neuronal Network run, beside that their theoretical descriptions and practical use.

3.1. LINEAR ACTIVATION

The activation function is a mathematical function applied to a signal at the output of an artificial neuron. The term activation function comes from the biological equivalent of activation potential, a stimulation threshold which, once reached, results in a response from the neuron [16].

The linear activation function is defined mathematically as:

$$y = x, \quad (1)$$

3.2. BINARY STEP FUNCTION

The binary step function is the choice of reference for an activation function in the process of creating a binary classifier, on the other hand when the variable contains several classes, the function is limited [16].

It is defined mathematically (for the threshold at 0) as:

$$\text{B. Step}(x) = \begin{cases} 1 & \text{if } x > 0 \\ -1 & \text{if } x \leq 0 \end{cases} \quad (2)$$

Its gradient can be written as follows:

$$\frac{d(\text{B}\cdot\text{Step }(x))}{dx} = \begin{cases} \text{Undefined} & \text{if } x = 0 \\ 0 & \text{if otherwise} \end{cases} \quad (3)$$

This function cannot be differentiated at0 and it’s gradient is equal to zero at the other values points. This leads to the discovery of a softer, differentiable approximation known as the sigmoid through which the gradients could flow backward.

3.3. SIGMOID

The performance of the sigmoid function is generally proven in classification problems [16].

Mathematically, sigmoid is defined as:

$$\text{sigmoid }(x) = \sigma(x) = \frac{1}{1+e^{-x}} \quad (4)$$

The sigmoid output is in the range [0, 1] and, therefore, can be interpreted as a probability. Its gradient can be formulated in expressions of the activation forms:

$$\frac{d(\sigma(x))}{dx} = \frac{e^{-x}}{(1+e^{-x})^2} = \sigma(x) \cdot (1 - \sigma(x)) \quad (5)$$

3.4. TANH (HYPERBOLIC TAN)

The hyperbolic tangent activation function is a function centered on zero, doubly saturating far from zero allowing to facilitate the modeling of the inputs whose values are extremely negative, positive or neutral [16].

It is mathematically defined as:

$$\tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \quad (6)$$

And its gradient can be written as:

$$\frac{d(\tanh(x))}{dx} = \frac{4}{(e^{x}+e^{-x})^2} = 1 - \tanh^2(x) \leq 1.0 \quad (7)$$

The hyperbolic tangent function has a characteristic which despite its importance does not make it successful in neural network architectures, its gradient only tends towards 1 when the input is equal to 0. in this case the function generates neurons dead during computation. The latter also presents a criterion relating to the activation weight.

3.5. THE RELU FAMILY

The ReLU family of activation functions is very popular and used in the majority of neural network architectures, below we will present the different characteristics of the functions of this family and give more details on the choice of use. [16].

A. ReLU

The most widely used activation function these days is the ReLU - Rectified Linear Unit - which is a piecewise linear function. Its advantage is that it replaces any negative input value with 0 and any positive value with itself, roughly max (0, x). And for its gradient, it becomes zero for negative values and is equal to 1 for positive values. This property is very interesting in the learning phase because it makes it possible to avoid and correct the vanishing gradient problem.Mathematically, ReLU is defined as:

$$\text{ReLU}(x) = max(0, x) = \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases} \quad (8)$$

Its gradient is given as:

$$\frac{d(\text{Re }LU(x))}{dx} = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x < 0 \\ \text{undefined} & \text{if } x = 0 \end{cases} \quad (9)$$

Its main limitation is that some gradients can be fragile and therefore die (cancel out) and this is due to its zero value for negative values. To address this, a modified version of ReLU, the Leaky ReLU is offered.

B. Leaky ReLU

In order to solve the Dead ReLU problem. A new activation function called Leaky ReLU [16] was proposed to set the first half of ReLU 0.01x instead of 0. Advantages of this function is that Their gradient for the negative input will not die compared to ReLU. on the other hand, the inconvenient is that the function is not consistent with a negative input. This leads to a zero gradient and slower training. To solve this problem, a leak is added to the activation instead of absolute zero.

The Leaky ReLU function is defined mathematically as:

$$\text{LReLU}(x) = \begin{cases} x & \text{if } x \geq 0 \\ 0.01x & \text{if } x < 0 \end{cases} \quad (10)$$

This leak increases the range of the ReLU. The gradient of the leaking ReLU is obtained as:

$$\frac{dL\text{Re }LU(x)}{dx} = \begin{cases} 1 & \text{if } x \geq 0 \\ 0.01 & \text{if } x < 0 \end{cases} \quad (11)$$

Furthermore, the most used activation functions among a large number of variants are the parameterized function ReLU and the exponential ReLU.

C. PReLU (Parametric ReLU)

The parametric ReLU known under the name of PReLU is another part of the activation functions of the ReLU family, the learning of the negative part is progressive while the positive part is linear. The PReLU is mathematically defined as follows:

$$\text{PReLU}(x_i) = \begin{cases} x_i & \text{if } x_i \geq 0 \\ \alpha_i x_i & \text{if } x_i < 0 \end{cases} \quad (12)$$

Where α_i is the back-propagation in other terms the gradient of the error for each neuron of a neural network used in training the parameter adjusting the negative slope. Here, the indexi refers to the ith channel in the CNN. For each entity map, α_i is shared, which reduces the risk of over-adjustment. Leaky ReLU uses α_i= 0.01, The slope of the negative part make parametric function learns progressively.There is a trade-off between computational costs when using PReLU. The PReLU gradient is mathematically defined as:

$$\frac{dPReLU(x_i)}{dx_i} = \begin{cases} 1 & \text{if } x_i \geq 0 \\ \alpha_i & \text{if } x_i < 0 \end{cases} \quad (13)$$

The Parametric ReLU function finds its utility and becomes the go-to choice in the event that the Leaky ReLU fails to deal with the problem of dead neurons, this limitation does not allow information to be generated at subsequent layers in an efficient manner.

4. CASE STUDY

Plant diseases affect crop production, causing significant losses to farmers and threatening food security. Using deep learning, we are trying to create a model capable of detecting diseases in the plant and using evaluation metrics, such as accuracy functions. The aim was to compare the transfer learning model using or not Siamese network particularly in low data regime.

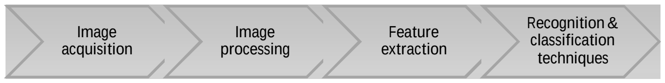


Figure 1: General steps applied to plant disease identification

As shown in Figure 1, the identification process begins with an image acquisition step to proceed to capture images of healthy and diseased plants under different shooting conditions. Then a thorough analysis is needed to process and edit the image and prepare it for the next step, such as image enhancement, segmentation and filtering. In particular, image segmentation methods, such as thresholding, are frequently used to detect outlines and boundaries of images used in the dataset.

As shown in Figure 2, a Convolutional Neural Network contains three main parts which are convolution, pooling and fully connected layers. The two first parts convolution and pooling layers' act as feature extractors from the input images while the third proceeded as a classifier. The aim of convolution is to automatically extract features from the inputs images. The dimensionality of these features allowing to project data from a large space to al smaller space is then reduced by the pooling layer to combat what is called the scourge of large dimensions. At the end of the model, the fully connected layer with a Sigmoid or Hyperbolic Tangent activation function makes use of the learned high-level features to classify the input images into predefined classes. In summary, the model is composed of two main parts: the first part is the self-taught feature extraction model and the second part is the classification model.

4.1. LOADING THE DATA

We used the Plant Village dataset, an open datasetof 20 639 composed of diseased and healthy plant leaves imagesgathered onlaboratory conditions. PlantVillage Dataset taken both in the laboratory and in real conditions from the crop fields. The plant images cover 13 species of crops including: pepper, potato and tomato. It contains images 27 classes (17-10, disease-healthy).

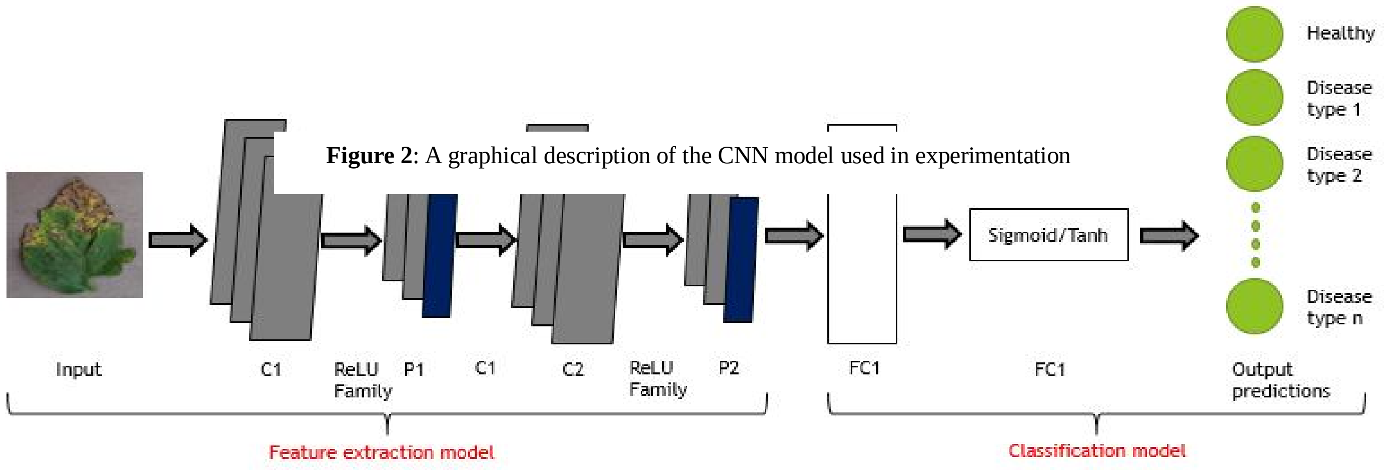
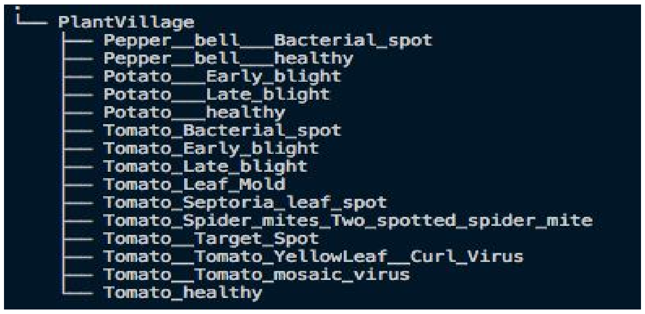


Figure 2: A graphical description of the CNN model used in experimentation



```
└── PlantVillage
    ├── Pepper__bell___Bacterial_spot
    ├── Pepper__bell___healthy
    ├── Potato___Early_blight
    ├── Potato___Late_blight
    ├── Potato___healthy
    ├── Tomato_Bacterial_spot
    ├── Tomato_Early_blight
    ├── Tomato_Late_blight
    ├── Tomato_Leaf_Mold
    ├── Tomato_Septoria_leaf_spot
    ├── Tomato_Spider_mites_Two_spotted_spider_mite
    ├── Tomato__Target_Spot
    ├── Tomato__Tomato_YellowLeaf__Curl_Virus
    ├── Tomato__Tomato_mosaic_virus
    └── Tomato_healthy
```

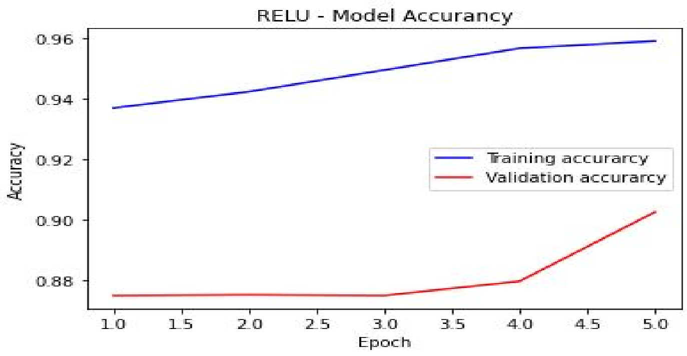
Figure 3: Plant disease dataset description

4.2. DATA PREPROCESSING

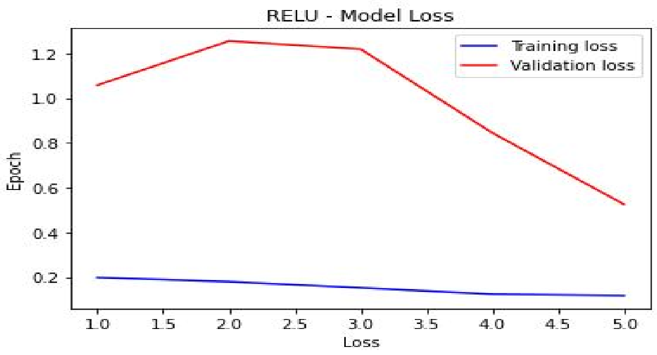
Firstly, we proceed to preprocessing the data by scaling the data points. Secondly, we achieved a split on the data using 80% of the images for training and 20% for testing. Finally, we created an image generator object. This image enhancement approaches were applied for the upgrade of the distribution of pixels over an extensive range of intensities, linear contrast stretching was applied on the images. During the image acquisition process, some undesirable noise information was added to the image due to nonlinear light intensity conceded as noise.

4.3. EXPERIMENTATION RESULTS

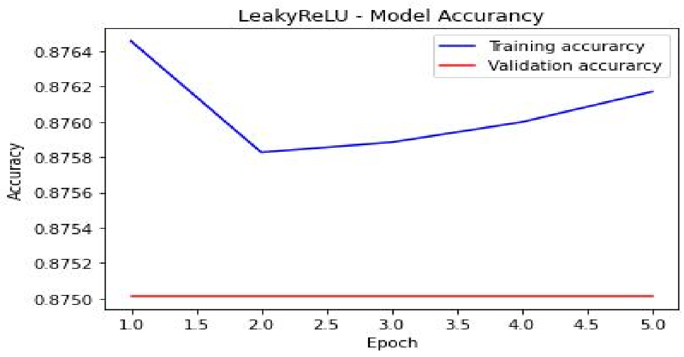
After training and validating the model, we obtain our results. Training accuracy is usually the accuracy when the model is applied on the training data. When the model is applied on test data from different classes, it is known as validation accuracy. The figure bellow shows a graph, which contains training and validation accuracy of our model by activation functions.



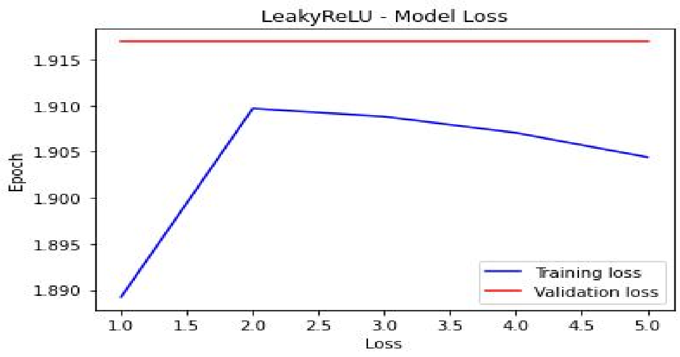
(a)



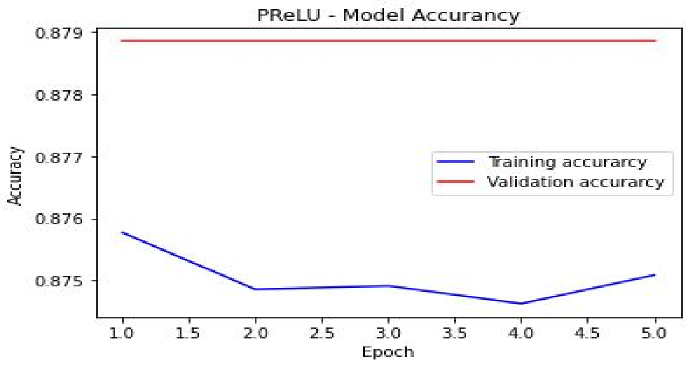
(b)



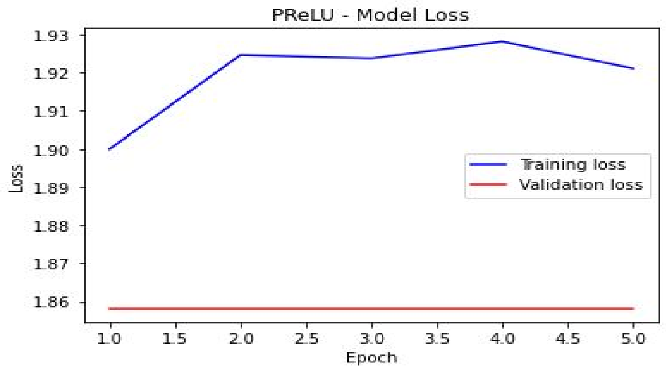
(c)



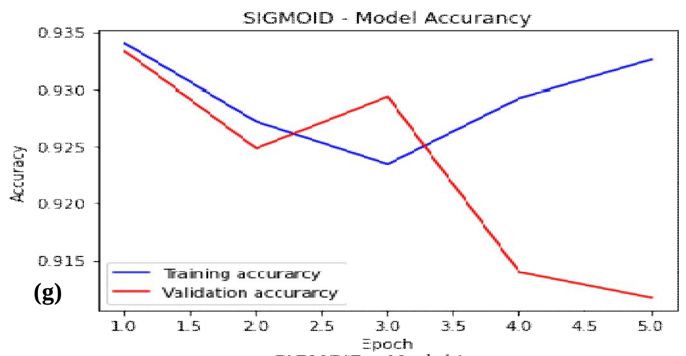
(d)



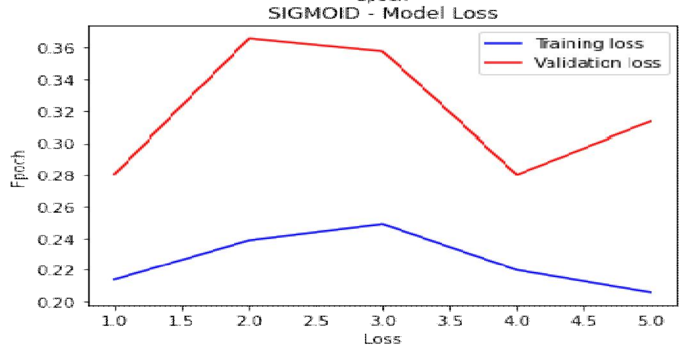
(e)



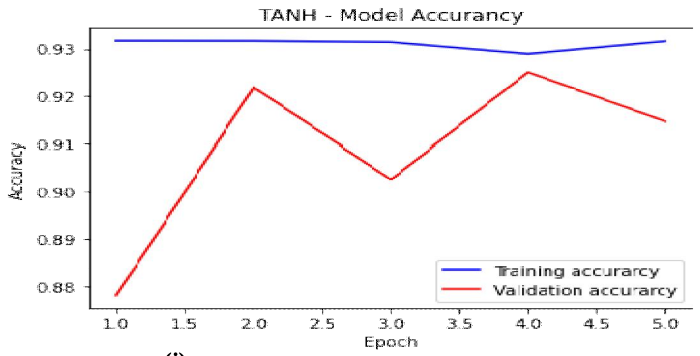
(f)



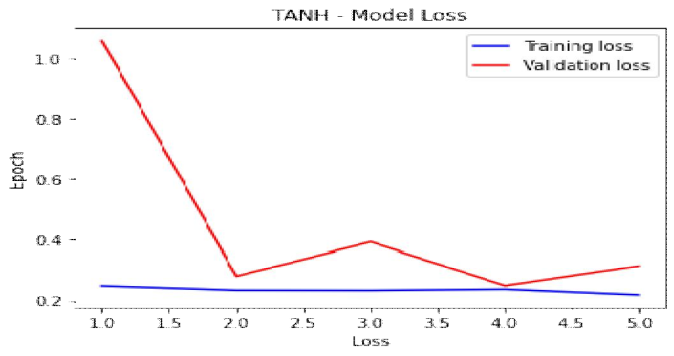
(g)



(h)



(i)



(j)

Figure 4: Training and Validation accuracygraph with ReLU, LeakyReLU, PReLU, Sigmoid and Tanh (a) (c) (e) (g) (i) Vs Training and Validation loss graph with ReLU, LeakyReLU, PReLU, Sigmoid and Tanh (b) (d) (f) (h) (j)

As shown in Figure 4 – (a), the curve of the ReLU function graph increases as the iteration number increases on validation and the sigmoid curve tip after a certain epoch number.

Table 1: Calculating model accuracy for 5 epochs

Activation Function	Model Accuracy	Model Loss
RELU	**96.22%**	**10.70**
PRELU	87.48%	16.24
LEAKY RELU	90.78%	13.82
Sigmoid	93.26%	20.60
Tanh	93.16%	21.95

As shown in Table 1, ReLU with and accuracy of 96.22% outperforms Leaky ReLU and PReLU according to the selected accuracy functions and loss functions respectively. Otherwise sigmoid and Tanh accuracy present a small difference of 0.1% for model accuracy and 1.35% on loss functions.

5. RESULT DISCUSSION

As with the results finding in the experimental analysis, ReLU, PReLU, Leaky ReLU, Tanh and Sigmoid activation functions give better results in terms of validation error and accuracy on plant disease dataset. With ReLu layer, overall performance and speed of the network taken into account computational cost is more powerful. We found that a CNN model with ReLU hidden layers (5 hidden layers here) gives best performing results and improves overall performance better in terms of accuracy and speed. This ReLU advantages in Convolutional Neuronal Network at number of hidden layers is suitable to effective and fast retrieval of images from the databases,which explains the use of this function in a large number of architectures. To conclude, we can say that the model combined with Relu on hidden layer and sigmoid on output layer is the one which yields the best results on the model used to detect plant disease. In further work, we can generalize these results to more complex classification tasks onto different datasets with different analyses. i.e. epoch variation and variant number of layers [17-18].

6. CONCLUSION

To study the effect of activation functions on classification accuracy using convolutional neuronal network. We make a model and test the accuracy performance affected by the choice of these functions in different level of the architecture of layers. The experimentation results valid that activation functions affect classification accuracy about more 8.74%

accuracy. Sigmoid Function used in the output layer combined with Relu in the hidden layers outperforms the other functions and are recommended to be used in classification case of plant disease detection.

In future works, we plan to focus on the use of new images and improved CNN models to improve classification performance. In addition, the prospect will focus to come up with a new and improved activation feature. The main objective of further research will be to improve the existing application of smart mobile devices capable of optimally detecting various plant diseases. This application, which will provide an automatic diagnosis of plant diseases with visual diagnosis, in this case will allow the user to detect plant diseases taking into account the real conditions of taking pictures of the latter.

REFERENCES

1. J. Boulent, S. Foucher, J. Théau, and P. L. St-Charles. **Convolutional Neural Networks for the Automatic Identification of Plant Diseases**, Front. Plant Sci., vol. 10, no. July, 2019.
2. Fuentes, S. Yoon, S. C. Kim, and D. S. Park, **A robust deep-learning-based detector for real-time tomato plant diseases and pests recognition**, Sensors (Switzerland), vol. 17, no. 9, 2017.
3. M. Nagaraju and P. Chawla. **Systematic review of deep learning techniques in plant disease detection**, Int. J. Syst. Assur. Eng. Manag., 2020.
4. K. Arai, S. Kapoor, and R. Bhatia, **Intelligent Computing**, Springer, pp. 37-50, vol. 3, 2020.
5. N. Networks and U. Michelucci. **Applied Deep Learning**. Springer, pp. 31-80, 2018.
6. Dureja and P. Pahwa. **Analysis of Nonlinear Activation Functions for Classification Tasks Using Convolutional Neural Networks**. Springer Singapore, 2019.
7. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall. **Activation Functions: Comparison of trends in Practice and Research for Deep Learning**, arXiv, pp. 1–20, 2018.
8. S. Pattanayak and S. Pattanayak. **Pro Deep Learning with TensorFlow**. Springer, pp. 89–117, 2017.
9. H. Habibi, A. Elnaz, J. Heravi, P. Application, and T. Detection. **Guide to Convolutional Neural Networks**, Springer, pp. 61-81, 2017.
10. C. Sagana, M. D. R, P. Keerthika, M. Sangeetha, M. Thangatamilan, and K. Devendran. **Comparative Analysis of different Activation Function for Pattern Classification Problem**,IJCA, vol. 13, no. 4, pp. 312–325, 2020.
11. J. Chen and Y. Wang. **Comparing Activation Functions in Modeling Shoreline Variation Using Multilayer Perceptron Neural Network**,Water, 12, 1281, 2020.
12. K. Arai, S. Kapoor, and R. Bhatia, **Intelligent Computing**, Springer, vol. 3. 2020.
13. J. Moon, S. Park, S. Rho, and E. Hwang. **A comparative analysis of artificial neural network architectures for building energy consumption forecasting**, International Journal of Distributed Sensor Network, vol. 15, no. 9, 2019.
14. S. Sharma and S. Sharma. **Activation Functions in Neuronal Networks**, International Journal of Engineering Applied Sciences and Technology, vol. 4, no. 12, pp. 310–316, 2020.
15. S. Qian, H. Liu, C. Liu, S. Wu, and H. San. **Adaptive activation functions in convolutional neural networks**, Neurocomputing, vol. 0, pp. 1–9, 2017.
16. W. Pedrycz and S. Chen, **Deep Learning: Algorithms and Applications**. Springer, pp. 04-19, 2020
17. M. Türkoğlu and D. Hanbay. ***Plant disease and pest detection using deep learning-based features***, Turk J Elec Eng& Comp Sci, pp. 1636–1651, 2019.
18. J. G. A. Barbedo. **Impact of dataset size and variety on the effectiveness of deep learning and transfer learning for plant disease classification**, Comput. Electron. Agric., vol. 153, no. October, pp. 46–53, 2018.