GSJ: Volume 11, Issue 8, August 2023, Online: ISSN 2320-9186
www.globalscientificjournal.com

Pre-Trained CNN Architectures in Yam Diseases Detection. A Deep Learning Approach

Peter E. Okimba[1], Virginia E. Ejiofor[2]
1 Taraba State Polytechnic, Suntai
2 Nnamdi Azikiwe University, Nigeria
rhemapok30@gmail.com, ve.ejiofor@unizik.edu.ng

Abstract

Every society's agricultural sector has long been harmed by the widespread crop diseases and mite damage. The following claim based on this premise: "Faster and an accurate prediction of leaf diseases in crops could be enable to design an early treatment strategy while significantly lowering economic losses. Researchers have been able to significantly increase the performance and accuracy of object identification and recognition systems thanks to recent advances in deep learning (DL). In this study, a DL approach for spotting illnesses on various plants' leaves is devised using pretrained CNN architecture (VGG16). The study involved in selecting a suitable pre-trained model for image classification tasks. The accuracy scores indicate the potential of these architecture, but further experimentation and fine-tuning may be required to achieve optimal performance on specific datasets and tasks. Above all, the model can help in monitoring yam plants by analyzing images captured through cameras or drones. It can alert farmers and researchers to the presence of diseases or attacks at an early stage, enabling timely interventions to minimize crop damage.

Keywords: CNN, Deep Learning, VGG16, Yam

1. Introduction

Hunger and food insecurity are the leading global disasters with about 815 million (approximately 10.7%) of the global population in extreme hunger or suffering from chronic undernourishment in 2016 [1]. Accordingly, FAO ascertain that, all cases of undernourishment occur in developing and least-developed countries except for 1.35% in the developed countries and 26.9% of these undernourishment cases occur in Africa.

Similarly, [2] in Nigeria, the prevalence of hunger and undernourishment is 1.58% (12.9 million) to the global rate with Northern Nigeria accounting for 8.9 million undernourishment cases caused by conflict-related displacements. With two-thirds of active force of Nigeria's 170 million population [3] employed in agriculture, which contributes 40 percent of her national GDP [4], one begins to wonder why 1.58% of this population is undernourished. Regrettably, the answer lies on ineffectiveness of her agricultural practices, which are still largely subsistent, un-mechanized, and the absence of any form of computer-assisted technology.

Although, Nigeria's efforts towards achieving United Nation's Millennium Development Goals (MDGs) was graded positive because of her 38.1 index change in reducing the number of people undernourished since 1990-2016 [1]. However, Food and Agriculture Organizations of United Nations (FAO), International Fund for Agricultural Development (IFAD) and World Food Program (WFP) are not optimistic that the current trend will achieve zero hunger [3].

This is, of course, the second target of the Sustainable Development Goals (SDGs), unless there is a deliberate intervention in her agricultural practice, which is afflicted by inadequate crop disease management, processing and storage issues, and other issues. Through the use of deep residual learning in yam disease identification, such strategic intervention in crop disease management is conceivable.

According to statistics, subsistence crop producers provide more than 80% of agricultural output [5] [6]. They possess tiny plots of land on which they raise a few commercial crops with the help of family and some paid labor. Furthermore, there are numerous losses as a result of insect and disease attack, et al., [7]. Agriculture experts have devoted research efforts toward preventing crop loss caused by widespread illnesses. Pesticide treatment has traditionally been used to combat pest attacks, which has been aided by integrated pest management (IPM) strategies [8].

The importance of yam in West Africa's food economy cannot be overstated. Because it is one of the most important sources of energy grown in the tropics, efforts to manage its diseases must be expanded. Yam crop is Nigeria's top staple food and the world's eighth most staple food, with Nigeria producing 60.5% (36 million metric tons) of the global production of 59.5 million metric tons [4]. Yam is a Dioscorea family edible tuber crop grown via its seedling (tuber). White yam (Dioscorea rotundata), water yam (Dioscorea alata), and yellow yam (Dioscorea Cayenensis) are the three most prevalent yam species in Nigeria. "Yam in Nigeria is also processed into various staple, intermediate, and end product forms that are used for direct consumption by animals, used as the basic ingredient for snacks, or made into flour used to make instant puree," writes [9]. Because Nigeria is the world leader in yam production, reducing crop disease losses will increase yield and move the world closer to food sufficiency, in keeping with the second aim of the UN SDGs. Traditional crop disease diagnosis, on the other hand, is innately impacted by opinions and limited to nations that can aid and sustain the necessary human framework et al.,[10]; et al., [11]. It has been noted in Nigeria that good disease diagnosis and control, which leads to the maximization of production capacity, ideally puts the country on the path to food sufficiency with numerous chain values from her export potential.

Millions of West Africans rely on yam (Dioscorea spp.) as a staple root crop. According to [12], It is primarily grown for food security and livelihood systems in the Derived and Southern Guinea Savannas. This sub-region produces approximately 57 million tons of yams (nearly 93% of global production) on 4.7 million hectares annually, primarily in five countries: Benin, Côte d'Ivoire, Ghana, Nigeria, and Togo. Farmers engage in yam cultivation for household food supply; income generation through marketing ware yams; and production of planting material to meet their own needs and generate some income from the sale of surplus seed yams. Yam production is marred by many constraints, the key among them being the scarcity of high quality seed yam of local popular and improved varieties, high levels of post-harvest losses (almost 40%), high production costs (high cost of seed, labour at land clearing and harvest, and staking all contributing almost 70% of the total production costs). Low and declining soil fertility, moisture stress as well as pests and diseases, mainly viruses, fungi and nematodes, were also earmarked as limiting factors to increased yam production. These obstacles not only undermine food production and farmers' ability to generate sustainable incomes, but also disproportionately impact on rural women. In

Nigeria, the most cultivated species are the D. rotundata (white yam), D. cayenesis, (yellow or guinea yam) and D. alata (water yam) [9].

For early disease detection, [14] conceived the machine learning algorithms for on-site classification of both diseased (Figure 1.1) and heathy walnut leaves. The purpose of this study was to build a robust CNN model that is able to classify images of leaves, depending on whether or not these are infected by anthracnose, and therefore determine whether a tree is infected. On image collection, a total of 4491 images of leaves, both with and without anthracnose, were gathered from a walnut crop field located in the Volos region, Greece. The anthracnose-infected leaf images amounted to 2356, which is slightly more than the healthy leaf images which amounted to 2135. Python's random.shuffle generator was used to ensure variability. The set of images were used both in grayscale and RGB mode, a fast Fourier transform was implemented for feature extraction, and a CNN architecture was selected based on its performance. Finally, the best performing method was compared with state-of-the-art convolutional neural network architectures.

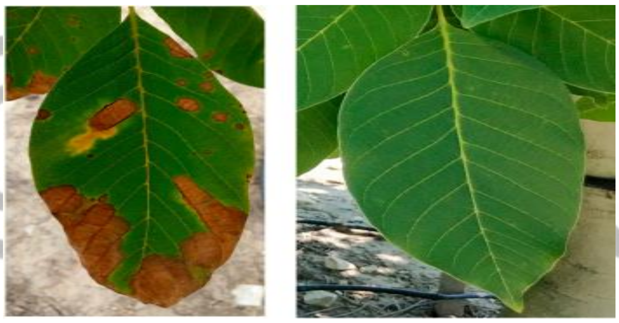


Figure 1.1: Anthracnose on Walnut leaves and Health Leaves et al., [14]

[15] concentrated on identifying tomato leaf disease using deep convolutional neural networks by transfer learning. The utilized networks are based on the pretrained deep learning models of AlexNet, GoogLeNet, and ResNet. The diseases considered here include; early blight, yellow leaf curl disease, Family, Geminiviridae, corynespora leaf spot disease, leaf mold disease, Tomato Mosaic Virus, late blight, septoria leaf spot, and two-spotted spider mite. First, they compared the relative performance of these networks by using SGD and Adam optimization method, revealing that the ResNet with SGD optimization method obtains the highest result with the best accuracy, 96.51%. The following accuracy was generated using the following architectures/optimizers: AlexNet/SGD (95.83%), AlexNet/Adam-(13.86%), GoogLeNet/SGD (95.66%), GoogLeNet/Adam (94.06%), ResNet/SGD (96.51%), and ResNet/Adam (94.39%). Then, the performance evaluation of batch size and number of iterations affecting the transfer learning of the

ResNet was conducted. A small batch size of 16 combining a moderate number of iterations of 4992 is the optimal choice in this work. The findings imply that, for a certain task, neither a big batch size nor a large number of iterations may increase the target model's accuracy. The structure was then fine-tuned using the best combined model. In recognizing tomato leaf disease, fine-tuning ResNet layers from 37 to "fc" yielded the maximum accuracy (97.28%).

2. Methodology

As this study is aimed at solving a peculiar food problem, it became necessary to interview some yam growers at cultivation sites in Otuocha, Anambra State, Nigeria. Difficulty in early disease detention – Most farmers decried their inability to detect early signs and symptoms of yam diseases before they spread to other yam stead. They also emphasized that the speed at which diseases of yam spread require early detection to cub the effects. Generally, most farmers blamed the poor knowledge of disease prognosis as the major problem in the management of yam diseases. These were some of the problems discovered from the interview. The implementation of DL for yam disease detection and diagnosis was done using Convolutional Neural Network architectures in Deep learning models such as; VGG 16, VGG 19, LeNet-5 (Adam Optimizer), LeNet-5 (SGD Optimizer), MobileNetV2, InceptionV3 and InceptionResNetV2 were the methods used to build and carryout diagnosis of yam diseases at early stage of yam growth. The specifications for the algorithm, in this instance pseudocodes, the input/output format, and the software module (Google Colab Notebook) are explained below.

3. Program Module Specification

Google Colab Notebooks runs on graphical processing units (GPUs) instead of the ubiquitous central processing units (CPUs). Despite the enormous computation power of CPUs, they were adjudged inefficient for simultaneous multiple computations. More so, since DL models can be time consuming during training, even the CPUs gave way for GPUs as a result of its parallelism. GPU is a smaller version of the computer in entirety but it is programmed to work on a particular issue. Unlike the CPU that operates of numerous tasks at the same time, GPU works on a specific issue. This is due to the fact that its motherboard is merged with video ram, and a thermal design for ventilation and cooling.

Lambdalabs maintained that in order to assess GPU effectiveness during neural networks' training, throughput should be chosen as the metric of measurement. Training throughput measures the number of samples (e.g. tokens, images, etc.) processed per second by the GPU. Using throughput instead of Floating Point Operations per Second brings GPU performance into the realm of training neural networks. Training throughput is strongly correlated with time to solution — since with high training throughput, the GPU can run a dataset more quickly through the model and teach it faster. In order to maximize training throughput it's important to saturate GPU resources with large batch sizes, switch to faster GPUs, or parallelize training with multiple GPUs. Additionally, it's also important to test throughput using state of the art model implementations across frameworks as it can be affected by model implementation. Figure 1.2 shows the training throughput of several GPUs.

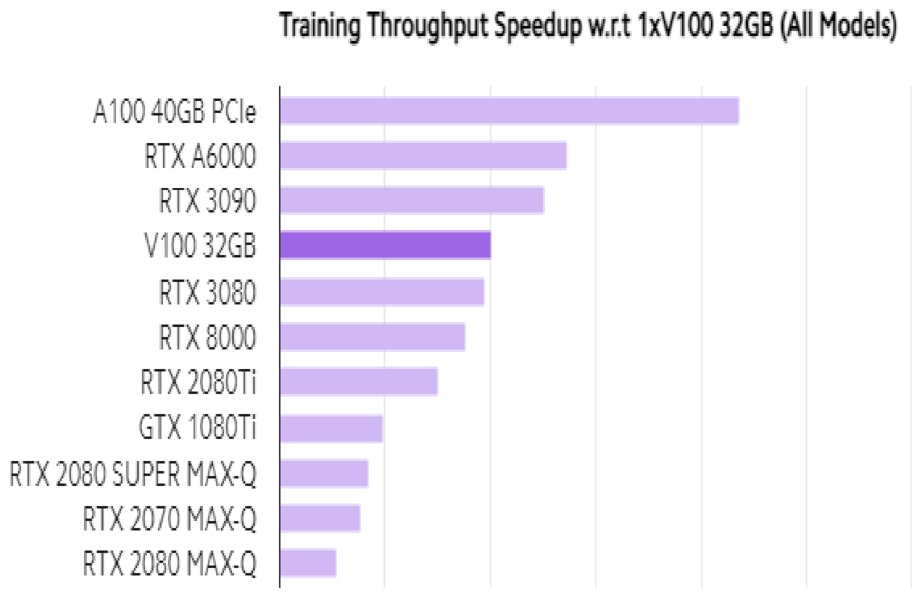
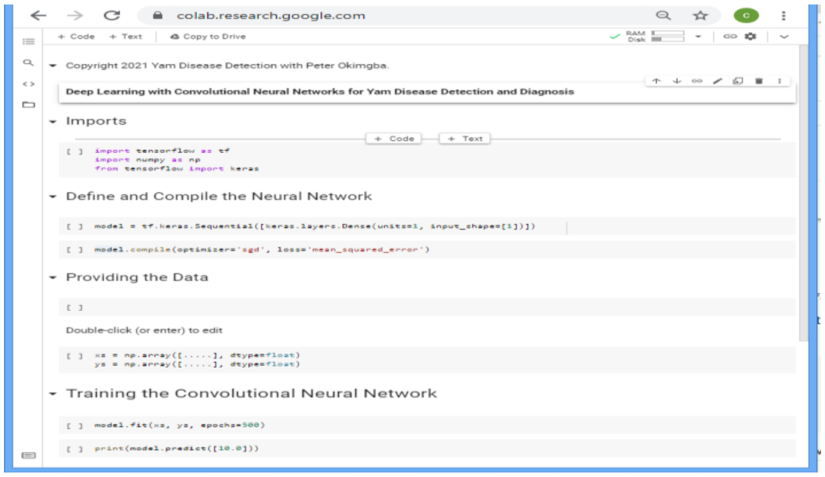


Figure: 1.2: Training throughput of several GPUs [13]

Control Centre/Main Menu

The implementation of DL for yam disease detection and diagnosis was done using the Google Colab notebook (Figure 1.3) – a powerful IDE for writing executable Python codes.



Copyright 2021 Yam Disease Detection with Peter Okimgba.

Deep Learning with Convolutional Neural Networks for Yam Disease Detection and Diagnosis

Imports

```
import tensorflow as tf
import numpy as np
from tensorflow import keras
```

Define and Compile the Neural Network

```
model = tf.keras.Sequential([keras.layers.Dense(units=1, input_shape=[1])])
```

```
model.compile(optimizer='sgd', loss='mean_squared_error')
```

Providing the Data

Double-click (or enter) to edit

```
xs = np.array([.....], dtype=float)
ys = np.array([.....], dtype=float)
```

Training the Convolutional Neural Network

```
model.fit(xs, ys, epochs=500)
```

```
print(model.predict([10.0]))
```

Figure 1.3: Google Colab Notebook for DL Implementation

4. Input/Output Format

As graphically described in the high level model section, the inputs to the CNN are images collected from farm sites were yam are cultivated. The images of diseases affecting the popular types of yams in these areas were collected using normal cameras as well as phone cameras. Note that after preprocessing, an image size of 224 X 224 will be fed batch per iteration to the CNN architectures. Figure 1.3 depicts a diseased leaf of yam while Figure 1.4 shows a collection of several leaves. Figure 1.5 shows a collection of tuber disease captured from storage barns.

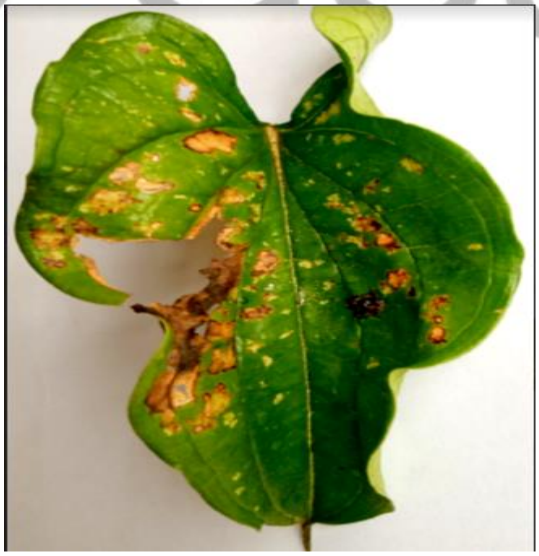


Figure 1.3: A diseased leaf of yam

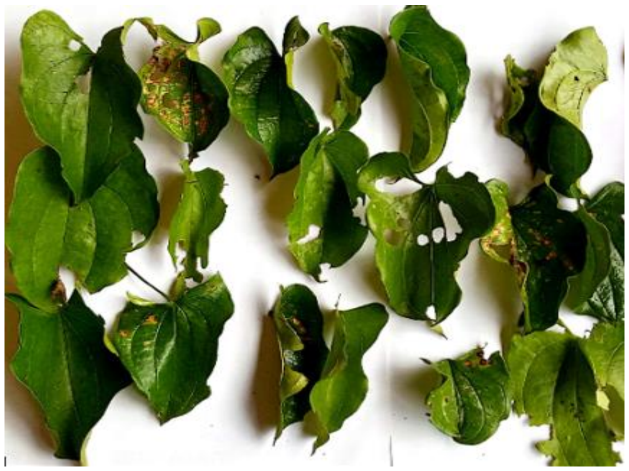


Figure 1.4: Collection of yam diseased leaves

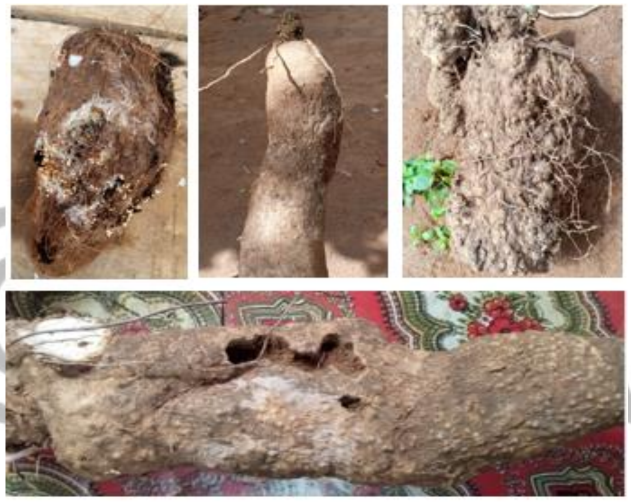


Figure 1.5: Collection of diseased tubers

The output of the system are the accuracies generated by testing the performance CNN architectures. Several properties of the confusion matrix, which include False Negative Rate (FNR), False Positive Rate (FPR), True Positive Rate (TPR) and True Negative Rate (TNR) were presented.

5. Pre-trained CNN Architectures with 6021 Images

VGG16

VGG16 is a convolutional neural network (CNN) architecture that was introduced by the Visual Geometry Group (VGG) at the University of Oxford. It is widely used for image classification tasks and has achieved excellent performance on various benchmark datasets. The VGG16 architecture consists of 16 layers, including 13 convolutional layers and 3 fully connected layers. The key characteristic of VGG16 is its simplicity and uniformity in architecture design. All convolutional layers have a small receptive field of 3x3, and the pooling layers have a fixed size of 2x2 with stride 2, which helps to reduce the spatial dimensions of the feature maps. The total number of parameters in VGG16 is 14,865,222. These parameters represent the weights and biases of the network, which are learned during the training process. The trainable parameters, which can be updated during training, amount to 150,534. While applying the VGG16 CNN Architecture to the yam dataset, Figure 1.6 showed the plot of Training Accuracy against Validation Accuracy, Figure 1.7 showed the plot of training loss against validation loss, while Figure 1.8 depicts the confusion matrix.

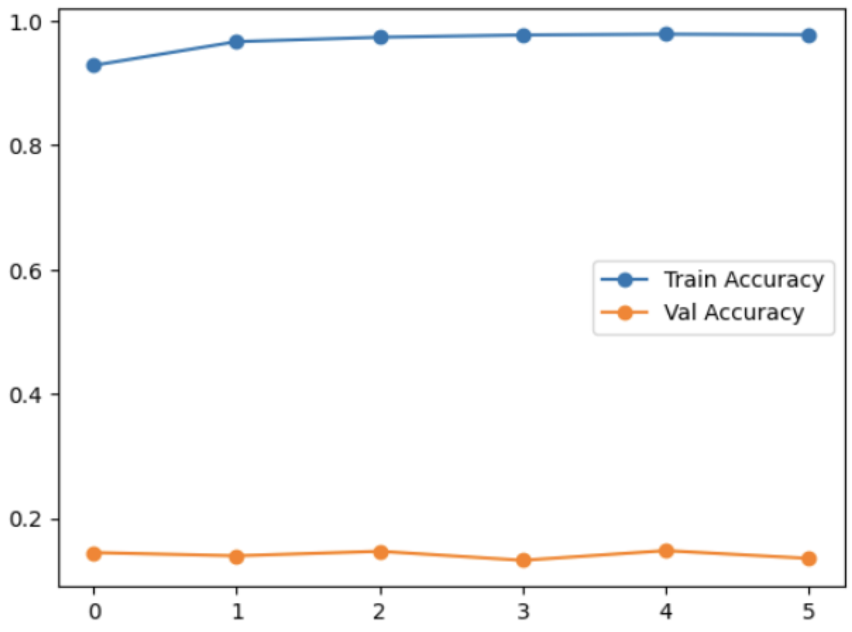


Figure 1.6: Plot of Training Accuracy against Validation Accuracy for VGG16 CNN Architecture

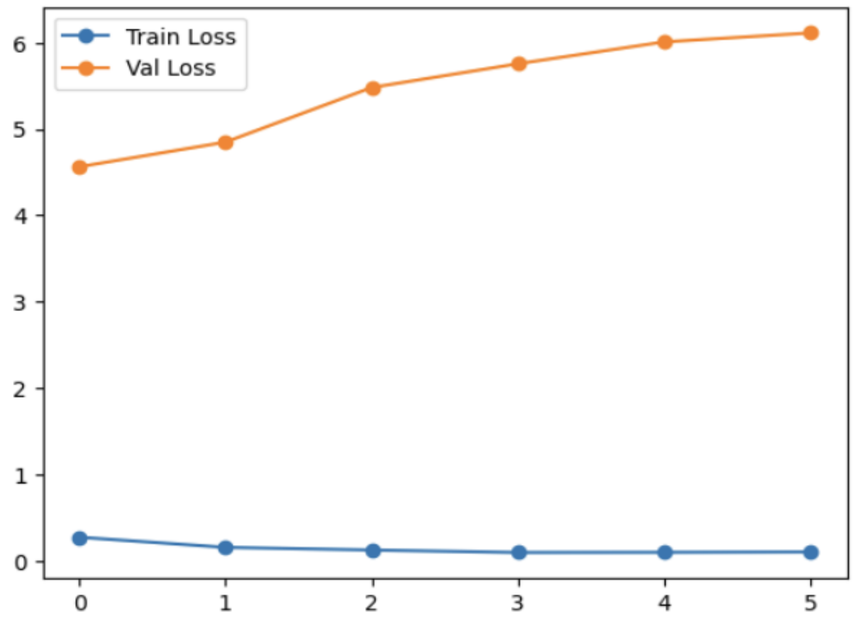


Figure 1.7 Plot of Training Loss against Validation Loss for VGG16 CNN Architecture

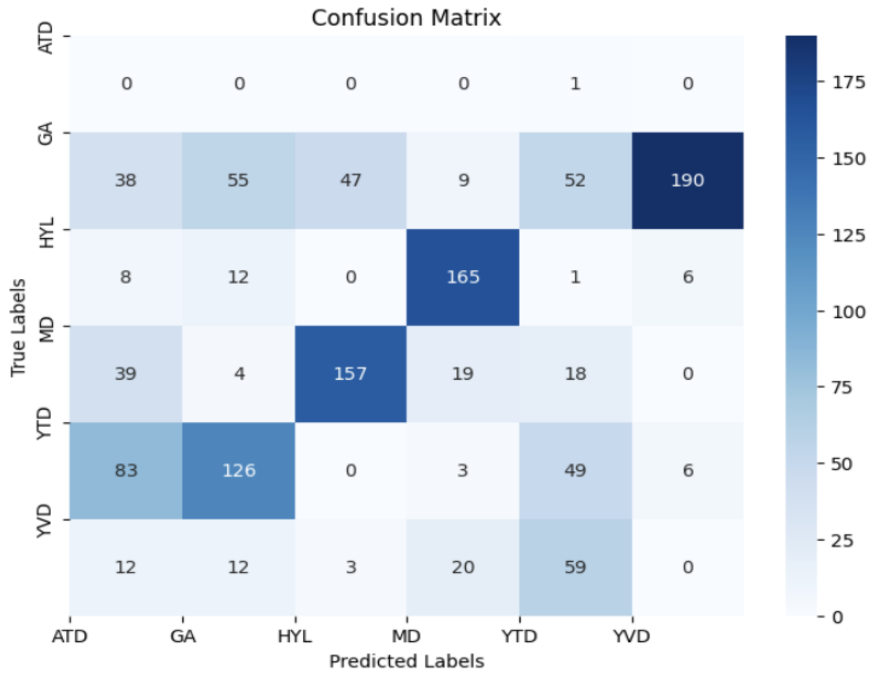


Figure 1.8 Confusion Matrix of VGG16 CNN Architecture

Class ATD – The precision for class ATD is 0.00, indicating that the model did not correctly classify any samples as ATD. The recall is also 0.00, suggesting that the model failed to identify any true positive samples for class ATD. The F1-score, which is the harmonic mean of precision and recall, is 0.00, indicating poor performance for class ATD. The support represents the number of samples for class ATD, which is 1 in this case.

Class GA – The precision for class GA is 0.26, indicating that the model correctly classified approximately 26% of the samples as GA. The recall is 0.14, suggesting that the model captured approximately 14% of the true positive samples for class GA. The F1-score is 0.18, providing a balanced measure of the model's performance for class GA. The support represents the number of samples for class GA, which is 391.

Class HYL – The precision for class HYL is 0.00, indicating that the model did not correctly classify any samples as HYL. The recall is 0.00, suggesting that the model failed to identify any true positive samples for class HYL. The F1-score is 0.00, indicating poor performance for class HYL. The support represents the number of samples for class HYL, which is 192.

Class MD: The precision for class MD is 0.09, indicating that the model correctly classified approximately 9% of the samples as MD. The recall is 0.08, suggesting that the model captured approximately 8% of the true positive samples for class MD. The F1-score is 0.08, providing a balanced measure of the model's performance for class MD. The support represents the number of samples for class MD, which is 237.

Class YTD – The precision for class YTD is 0.27, indicating that the model correctly classified approximately 27% of the samples as YTD. The recall is 0.18, suggesting that the model captured approximately 18% of the true positive samples for class YTD. The F1-score is 0.22, providing a balanced measure of the model's performance for class YTD. The support represents the number of samples for class YTD, which is 267.

Class YVD – The precision for class YVD is 0.00, indicating that the model did not correctly classify any samples as YVD. The recall is 0.00, suggesting that the model failed to identify any true positive samples for class YVD. The F1-score is 0.00, indicating poor performance for class YVD. The support represents the number of samples for class YVD, which is 106.

The accuracy of the model is 0.10, meaning that only 10% of the predictions made by the model are correct. The macro average F1-score is 0.08. These metrics provide insights into the model's performance for each individual class. It appears that the model struggles to accurately classify samples for several classes, such as ATD, HYL, and YVD, as indicated by the low precision, recall, and F1-scores. Improvements may be needed in the model architecture, data representation, or data collection process to enhance performance for these specific classes.

6. Research Findings of the Pretrained CNN Architectures

By analyzing table 1.1 column-by-column, we can observe the variations in architectural complexity, parameter counts, training times, input shapes, early stopping usage, and accuracy levels for the different pre-trained CNN architectures.

Table 1.1: Pre-trained CNN Architectures and Accuracies

CNN	**Total Params**	**Trainable Params**	**Non-trainable Params**	**Training Time (Mins)**	**Input Shape**	**Early Stopping**	**Accuracy (Test set)**
VGG 16	14,865,222	150,534	14,714,688	16.670	224, 224	Yes (Epochs 6)	10.34%
LeNet-5 (Adam Optimizer)	46,661,506	46,661,506	0	22.493	224, 224	Yes (Epochs 12)	17.84%
LeNet-5 (SGD Optimizer)	46,661,506	46,661,506	0	15.889	224, 224	Yes (Epochs 9)	16.72%

The CNN column represents the names of the pre-trained CNN architectures used in the analysis. Which is the VGG 16, LeNet-5 (Adam Optimizer) and LeNet-5 (SGD Optimizer). The "Total Params" column indicates the total number of parameters in the CNN architecture. It represents the overall complexity and size of the model. The "Trainable Params" column shows the number of parameters that are updated during the training process. These parameters are learned and optimized based on the given dataset. The trainable parameter values for the architectures listed in the table are relatively small, with the highest being 46,661,506 (LeNet-5 with both Adam and SGD optimizers). These parameters are typically pre-trained or fixed and contain valuable knowledge transferred from previous training. The "Input Shape" column describes the dimensions of the input images expected by the CNN architecture. The input shapes mentioned in the table are either 224 by 224 for VGG 16. The "Early Stopping" column indicates whether early stopping was applied during the training process. Early stopping is a technique to prevent overfitting by stopping the training if the performance on a validation set starts to degrade. The values in this column indicate "Yes (Epochs X)," where X represents the number of epochs at which early stopping was triggered. The last column, "Accuracy (Test set)," shows the accuracy achieved by the CNN architecture on the test set. The value in this column was 10.34% (VGG 16) to 17.84% (LeNet-5 with Adam Optimizer).

VGG 16 is a Convolutional Neural Network (CNN) architecture. It has a total of 14,865,222 parameters, out of which 150,534 are trainable parameters. The training time for this model is 16.670 minutes. It takes an input of size 224 by 224. Early stopping is enabled, and it stops training after 6 epochs. LeNet-5 with Adam Optimizer is a CNN architecture. It has a total of 46,661,506 parameters, all of which are trainable. The training time for this model is 22.493 minutes. The input shape is 224 by 224. Early stopping is employed after 12 epochs. The accuracy achieved on the test set is 17.84%. LeNet-5 architecture was also implanted using the SGD optimizer instead of Adam. The training time for this model is 15.889 minutes. The input shape remains the same at 224 by 224. Early stopping is applied after 9 epochs. The accuracy achieved on the test set is 16.72%.

1.2 Pre-trained CNN Architectures (Contd.)

CNN	**Train Loss**	**Train Accuracy**	**Validation Loss**	**Validation Accuracy**	**Test Loss**	**Test Accuracy**
VGG 16	0.1032	0.9782	6.1119	0.1354	6.8035	0.1030
LeNet-5 (Adam Optimizer)	1.7956	0.1812	1.8064	0.1813	1.8077	0.1784
LeNet-5 (SGD Optimizer)	1.7945	0.1758	1.7979	0.1698	1.6023	0.1672

For VGG 16, during training, the model achieved a low training loss of 0.1032 and a high training accuracy of 0.9782. However, on the validation set, it obtained a higher validation loss of 6.1119 and a lower validation accuracy of 0.1354. On the test set, the model had a test loss of 6.8035 and a test accuracy of 0.1030. For VGG 19, similar to VGG 16, the training loss was low (0.1033) and the training accuracy was high (0.9805). However, the validation loss was 5.6626, and the validation accuracy was 0.1458. On the test set, the model achieved a test loss of 5.8098 and a test accuracy of 0.1222. For LeNet-5 with the Adam optimizer, the training loss was relatively high

(1.7956), and the training accuracy was low (0.1812). The validation loss and accuracy were similar to the training metrics, with a validation loss of 1.8064 and a validation accuracy of 0.1813. On the test set, the model achieved a test loss of 1.8077 and a test accuracy of 0.1784. LeNet-5 with the SGD optimizer had similar metrics to the LeNet-5 with Adam optimizer. The training loss was 1.7945, and the training accuracy was 0.1758. The validation loss was 1.7979, and the validation accuracy was 0.1698. On the test set, the model achieved a test loss of 1.6023 and a test accuracy of 0.1672.

7. Conclusion

Model Performance: The accuracy scores on the test set for the pre-trained CNN architecture 10.34%. This accuracy indicates the effectiveness of the model in classifying the target variable. It is important to note that this accuracy score may vary depending on the specific dataset and task.

Model Complexity: The complexity of the model can be determined by the number of parameters it has. Higher parameter count indicates more complex model. It is of note that, model complexity alone does not guarantee better performance.

Training Time: Shorter training times can be advantageous in scenarios where computational resources or time constraints are a concern. However, it is important to consider that training time alone does not determine the model's quality or effectiveness.

Input Shape: The input shape of the images used for training the model varies between 150 x 150 and 299 x 299. Larger input shapes provide more detailed information to the model but may also require more computational resources. It is crucial to choose an appropriate input shape based on the specific requirements of the image classification task.

Early Stopping: The use of early stopping, as indicated by "Yes" in the table, can help prevent overfitting and improve generalization by stopping the training process when the model's performance on a validation set starts to degrade. It is observed that early stopping was employed the architecture, except for LeNet-5 with the SGD optimizer. Early stopping can contribute to improved performance and prevent overfitting.

References

[1] Food and Agriculture Organization, World Food Programme, and European Union. (2018). Global report of food crises 2018. Retrieved from http://www.fao.org /fileadmin /user_upload / fsin/docs/global_ report /2018 /GRFC _2018 _Full _report EN.pdf. Accessed April, 2019.

[2] Food and Agriculture Organization, IFAD and World Food Programme (2015). The State of Food Insecurity in the World. Retrieved from Food security indicators http://www.fao.org/economic/ess/ ess-fs/ess-fadata/it/#. VRuyjOEZbqc). Accessed April, 2020.

[3] United Nations Development Programme Nigeria (2016). UNDP Nigerian Annual Report 2016. Retrieved from https://www.undp.org/..../nigeria/..../ UNDP%202016%20 Annual% Report. Accessed, April, 2020.

[4] Food Security Portal. (2014) Food Security Portal-Nigeria. Retrieved from http:// www.foodsecurityportal.org/nigeria Accessed April, 2020.

[5] UNEP (2013). Smallholders, Food Security, and the Environment. Rome: International Fund for Agricultural Development (IFAD). Availableonlineat: https://www.ifad.org/documents/10180/666cac2414b643c2876d9c2d1f01d5dd.

[6] Mohanty, S. P., Hughes, D. P., and Salathé, M. (2016). Using deep learning for image-based plant disease detection. *Front. Plant Sci*. 7:1419. Doi: 10.3389/fpls.2016.01419

[7] Harvey, C. A., Rakotobe, Z. L., Rao, N. S., Dave, R., Razafimahatratra, H., and Rabarijohn, R. H. (2014). Extreme vulnerability of smallholder farmers to agricultural risks and climate change in Madagascar. *Philos.Trans.R.Soc. Lond.BBiol.Sci.* 369: 20130089. doi:10.1098/rstb.2013.008

[8] Ehler, L. E. (2006). Integrated pest management (ipm): definition, historical Development and implementation, and the other IPM. *Pest Manag. Sci*. 62, 787-789. doi:10.1002/ps.1247

[9] Amusa, N. A., Adegbite, A. A, Muhammed S. and Baiyewu R. A. (2003). Yam diseases and its management in Nigeria. *African Journal of Biotechnology*, vol. 2 (12), pp. 497-502, December 2003.

[10] Bock, C., Poole, G., Parker, P., and Gottwald, T. (2010). Plant disease severity estimated visually, by digital photography and image analysis, and by hyperspectral imaging. *Crit. Rev. Plant Sci.* 29, 59–107. doi: 10.1080/07352681003617285

[11] Ramcharan, A., McCloskey, P., Baranowski, K., Mbilinyi, N., Mrisho, L., Ndalahwa, M., Legg, J. and Hughes, D. P. (2019). A Mobile-Based Deep Learning Model for Cassava Disease Diagnosis. Frontiers in Plant Science, Volume 10 (272), pp 1- 8. DOI: 10.3389/fpls.2019.00272

[12] Sanginga, N. and Mbabu, A. (2015). Root and Tuber Crops (Cassava, Yam, Potato and Sweet Potato). Feeding Africa, pp. 1 – 26.

[13] Exxactcorp. deep-learning-benchmarks-for-tensorflow. Assessed February 10, 2021. https://www.exxactcorp.com/blog/Benchmarks/nvidia-a100-deep-learning-benchmarks-for-tensorflow

[14] Anagnostis, A., Asiminari, G., Papageorgiou, E. and Bochtis, D. (2020). A Convolutional Neural Networks Based Method for Anthracnose Infected Walnut Tree Leaves Identification. *Applied Sciences*, 10(469). DOI: 10.3390/app10020469.

[15] Zhang J., Kong F., Wu J., Han S., and Zhai Z. (2018). Automatic image segmentation method for cotton leaves with disease under natural environment. Journal of Integrative Agriculture 2018, 17(8), pp. 1800–1814.