Chapter 9
Plant Disease Detection Using Generative AI and Deep Learning Models

N. Bharanidharan
https://orcid.org/0000-0001-9064-8238
Vellore Institute of Technology, India

R. S. Sarweshwaran
Vellore Institute of Technology, India

M. Shomesh
Vellore Institute of Technology, India

S. Tharun
Vellore Institute of Technology, India

Kumar V. Vinoth
Vellore Institute of Technology, India

ABSTRACT

Cassava Plant scientifically Manihot esculenta which is also known as the tapioca plant, is a shrub that is majorly cultivated in many countries. The major cassava plant diseases are Cassava Bacterial Blight (CBB), Cassava Brown Streak Disease (CBSD), Cassava Green Mottle (CGM) and Cassava Mosaic Disease (CMD). The objective of this work is to make the Cassava Plant Disease Prediction from the given input image and tell whether the provided sample is infected with the above mentioned disease or not. To meet the main objective, we build the Model with the Convolution Neural Network, Visual Geometry Group 16, ResNet50, EfficientNetB0, Visual Transformer, and Variational Auto-encoder. Variational Auto-encoder model with various latent dimension size and optimizers are tested. Usage of generative variational autoencoder provides the highest accuracy of 95% while the other tested models are providing accuracy less than 90%.

DOI: 10.4018/979-8-3693-8332-2.ch009

1. INTRODUCTION

Cassava which is termed Manihot esculenta scientifically, is declared a drought plant due to its tolerance to climatic conditions. This plant will be cultivated mainly in subtropical areas of the world. This cassava plant's born place is South America and is currently cultivated mainly in Brazil and many developing countries. The cassava plant witnesses diseases by both components namely the Biotic factor and Abiotic factor. The biotic organisms are the major reason for the disease of the plant and the agents are viruses, nematodes, bacteria, and fungi. The Abiotic factors which can be a threat to the cassava plant are environment-oriented factor stresses, deficiency in nutrients and not containing the required resources for cultivation (McCallum & et. al., 2017).

Cassava plants witness many diseases including Cassava Bacterial Blight (CBB) which can occur because of the Bacteria Xanthomonas axonopodis pv manihotis, which is shortly called Xam. The bacteria found in the plant can cause the loss of a major part in leaves called necrosis which leads to poor quality in production and root damage which will affect the plant growth. This bacteria also affects the stem with a disease called lesions which always poses a high level of risk for plants since it affects the leaf parts with major spots in the leaf. If this disease occurs in the cassava plant, it leads to the death of the cassava plant. This disease is mainly witnessed in a major part of Africa including central and part of West Africa.

The next important disease which is caused by viruses is Cassava Mosaic Disease (CMD) by the viruses with the scientific name Begomo viruses, Potyviruses (McCallum & et. al., 2017**).** It can be seen in many African and east African countries. This disease makes plant production and affects the major parts of the leaves by making distortion and it reduces plant growth. Transportation of the viruses is mainly possible through the vectors of the leaves which is called as whiteflies and similar insects. Another major reason for the loss of plant growth is environmental impacts such as less fertility due to less nutrients in the soil, poor management of crop rotation, and variation in climate.

Infection of the Casava plant by another virus which is similar to potyviruses is also more prevalent. This virus variant is known as Cassava Brown Streak Virus and it is responsible for the reason of the syndrome called Cassava Brown Streak Disease (CBSD) (Legg et. al., 2015). The following viruses are primarily found in Uganda and are classified as Ugandan Cassava Brown Streak Viruses. This disease mainly affects the roots of the cassava plant. Similarly, another virus, known as Cassava Green Mottle Virus (CGM), is found in many parts of Africa and typically affects the leaves, causing them to change color. In some cases, protozoa, which are rarely involved, can also contribute to the disease. Additionally, insects such as grasshoppers and leafhoppers are major vectors of cassava motile disease, exacerbating its spread. Environmental factors, such as stress, also play a significant role in the disease's development. Ultimately, healthy plants can thrive with proper cultivation practices, ensuring the best quality crops.

The Cassava plant is an important crop that will be in growing in many tropical similar countries and it is main food for people who rely on carbohydrates. This plant has special growth in bad or limited soil conditions. The cassava plant can be cultivated in low photosynthesis and it can produce greater results to the farmers. This plant as many significant adaptations called resistance in drought by making its parts to adapt to the environment. They make this by reducing the transpiring level and make the roots adjustable automatically to find the water in low ground level. This enables the plant to withstand more humid climate changes with the help of changing the regulation of the leaf. This behavior allows the plant to survive in high humid climate changes. Cassava is an important crop in developing countries, where it serves as a key food source. The plant's edible parts are used in various food preparations, with its byproducts classified as flour, fufu, gari, and tapioca, all of which are commonly consumed in daily meals. In addition to its culinary uses, cassava is also a vital food resource for livestock in rural areas, helping to reduce food costs and maintenance in these communities. Beyond food, cassava has industrial applications, such as the production of ethanol and starch. The starch derived from cassava is used in a wide range of industries, including paper manufacturing and textiles. Moreover, cassava contributes to the development of renewable energy sources, with its ethanol content being utilized as biofuel, benefiting both local economies and environmental sustainability in nearby regions. (Connor et. al., 2021).

The problems associated with cassava cultivation can be attributed to several factors. The first major issue is the prevalence of diseases, which cause significant damage to plants. As a result, farmers suffer from food shortages. This issue is often linked to poor crop rotation practices, with farmers unaware of the importance of isolating diseased plants and failing to anticipate the impact of climate change on crop health (Parmar et al., 2017). Another challenge lies with the workers, who struggle to identify diseased plants and are often unaware of the benefits of using natural fertilizers, making it difficult to achieve optimal production. Additionally, post-harvest losses are a major concern, particularly in underdeveloped areas. Cassava roots, which are large and require a consistent water supply, are vulnerable to spoilage due to inadequate water management. This results in significant losses for farmers, further exacerbating the challenges of cassava cultivation.

The outline of the remaining paper is presented as follows: the next section describes the basics of deep learning; the third section deals with the literature survey; the fourth section explains the methodology; performance metrics are described in the fifth section; results and discussion are presented in the sixth section followed by conclusion at the last section.

2. DEEP LEARNING ALGORITHMS

Deep Learning is the subset of Machine Learning and Artificial Intelligence, which has advanced concepts when correlated to Machine Learning algorithms. It has numerous networks and nodes interconnected to each other to make the model to learn and perform effectively. Classical machine learning algorithms rely on custom-made feature extraction, where the deep learning models learn the hierarchical patterns automatically and representations from the raw data and learn itself by extracting the features.

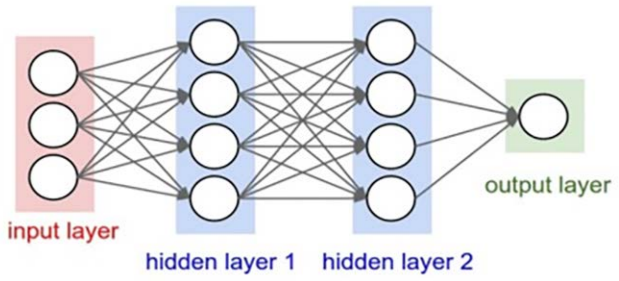
Types of Deep Learning:

1. Feed Forward Neural Network
2. Recurrent Neural Network
3. Convolutional Neural Network

Feed Forward Neural Network:

The Feed Forward Neural Network is a variety of Deep Learning neural network, which ensures that the input nodes process the provided data and give the output in the output nodes, it does not form any cycle. It is a fully connected one. The architecture of the feed-forward neural network is represented in the Fig.1

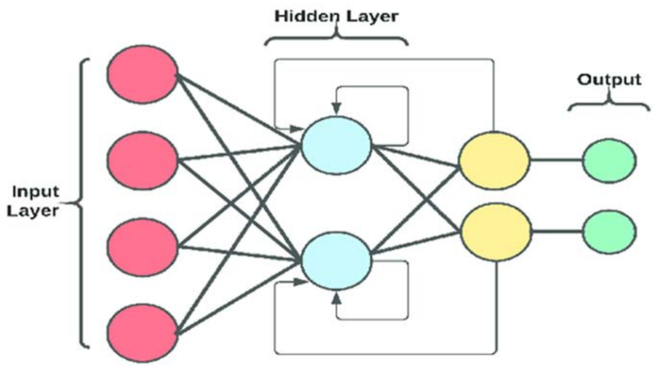
Figure 1. Feed forward neural network-architecture



Recurrent Neural Network:

It is a variety of the feed-forward neural network, in which the neurons present in the hidden layer receive the input with some delay in time, it mainly accesses the preceding learning and trains again to learn from it. In this the output will be used for both the output and the same will be used to learn and train the network. The architecture of the recurrent neural network is represented in Fig. 2

Figure 2. Recurrent neural network-architecture



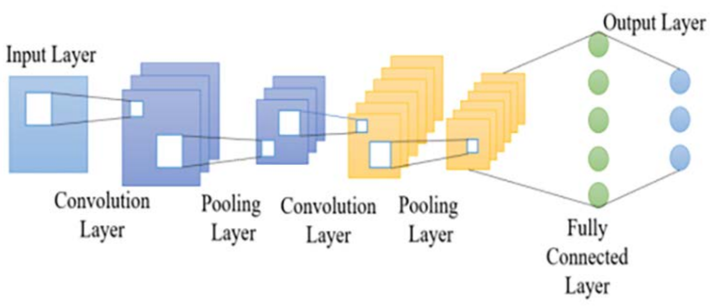
Convolutional Neural Network:

It is a variety of Deep Learning Neural Networks, which is of a special kind, in this neural network the layers are made in such a way that is more complex and such that it can be used to handle the complex tasks to be handled in a simpler manner. It is primarily used for the classification of input images, clustering of input images, and the detection of objects. The architecture of the CNN is shown in the Fig. 3. Considering the Cassava Plant Disease classification, we are using the following categorized models of Convolutional Neural Network (CNN), Visual Geometry Group 16, ResNet50, EfficientNetB0, Visual Transformer.

2.1. Convolution Neural Network (CNN)

Convolution Neural Network is a type of Deep Neural network that consists many of interconnected layers in its architecture, in this the convolution layer does the main operation by working with the matrix convolution filters or the kernels, (Mathulaprangsan & Lanthong 2021), (Thaiyalnayaki et al. 2022). The architecture diagram of the CNN is represented in Fig. 3. Moreover, the architecture of the CNN can be defined as per our need and it can be used to train the model.

Figure 3. General architecture of CNN



The figure above illustrates the basic architecture of a Convolutional Neural Network (CNN). In this architecture, the CNN model extracts features from the image dataset and uses them to train the model. For our specific problem, we developed a custom CNN model with five distinct output nodes, tailored to classify the dataset into five different categories. Then the transformation has been applied to resize the image into 224*224 size such that it works properly. The formula for calculating the output is the following as: Output dimension = ((input_demension – filter size + 2 *padding)/strider) + 1.

Considering our Custom CNN the general feature map is calculated as per the following Equation (1).

$$Y_{i,j,k} = \sum_{c=1}^{C}\sum_{m=1}^{H}\sum_{n=1}^{W} X_{i+m-1,j+n-1,c} \cdot W_{m,n,c,k} + b_k \tag{1}$$

Where:

$X_{i+m-1,j+n-1,c}$ is the input pixel value at channel c.
$W_{m,n,c,k}$ are the weights of the convolutional filter of size HxWxC.
b_k is the bias term for the k-th filter.
The output $Y_{i,j,k}$ is the activation value at position (i, j) for the k – th filter.
Similarly, the Equations for the 5 Layers is given as follows in the Equations (2) – (6).
Layer 1: Convolution + BatchNorm + ReLU + Pooling

Conv1:Z1=ReLU (BatchNorm (Conv2D (X, W1) +b1)) (2)

Input: X□R3×224×224

Output: Z1□R32×112×112 after 2×2 MaxPooling.

Layer 2: Convolution + BatchNorm + ReLU + Pooling

Conv2:Z2=ReLU (BatchNorm (Conv2D (Z1, W2) +b2)) (3)

Input: Z1□R32×112×112

Output: Z2□R64×56×56 after 2×2 MaxPooling.

Layer 3: Convolution + BatchNorm + ReLU + Pooling

Conv3:Z3=ReLU (BatchNorm (Conv2D (Z2, W3) +b3)) (4)

Input: Z2□R64×56×56

Output: Z3□R128×28×28 after 2×2 MaxPooling.

Layer 4: Convolution + BatchNorm + ReLU + Pooling

Conv4:Z4=ReLU (BatchNorm (Conv2D (Z3, W4) +b4)) (5) Input: Z3□R128×28×28

Output: Z4□R256×14×14 after 2×2 MaxPooling.

Layer 5: Convolution + BatchNorm + ReLU + Pooling

```
Conv5:Z5=ReLU (BatchNorm (Conv2D (Z4, W5) +b5)) (6)


Input: Z4R256×14×14


Output: Z5R512×7×7 after 2×2 MaxPooling
```

The Flattening Layer:

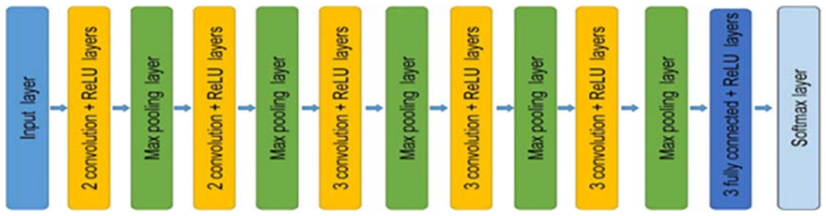
The extracted features have been flattened before passing to the output layer, it is done by the following Equation (7)

$$Flattened\ Output = Flatten(Z_5) \in \mathbb{R}^{512x7x7} = \mathbb{R}^{25088} \tag{7}$$

2.2. Visual Geometry Group 16

The Visual Geometry Group 16 (VGG16) model is a Convolutional Neural Network (CNN) architecture which is proposed by the Visual Geometry Group (VGG), Oxford University. The Visual Geometry Group is a 16-layer CNN model in general it has pretrained with the 1000 classes. The VGG16 model is used to effectively identify cassava plant disease (A. Soujanya et. al., 2023). The general Architecture of the VGG 16 is represented in the Fig.4

Figure 4. VGG 16 architecture


Input layer
2 convolution + ReLU layers
Max pooling layer
2 convolution + ReLU layers
Max pooling layer
3 convolution + ReLU layers
Max pooling layer
3 convolution + ReLU layers
Max pooling layer
3 convolution + ReLU layers
Max pooling layer
3 fully connected + ReLU layers
Softmax layer

The Output Layer for the Cassava Plant Disease is given in Equation (8).

$Z = W_5 \,.\, a + b_5$ (8)

Where:

$W_5 \in \mathbb{R}^{5x4096}$

$b_5 \in \mathbb{R}^5$

2.3. ResNet50

The ResNet50 is the pretrained model of the CNN which stands for the Residual Neural Network which is developed in 2016 to overcome the difficulties in the existing Deep Learning models, and the architecture of the ResNet50 is made in such a way to handle the training process effectively with its complex architecture. (Darwish & et.al 2024). The learning from this network has been adopted on several layers, the following Equation (9) provides the residual definition.

$Y_{pred} = f(x;\theta)$ (9)

Where:

Y_{pred} = predicted output

f = model function (VGG16)

x = input image tensor

ϴ= model parameters (weights and biases).

The ResNet has several variants that were proposed to enhance the accuracy and functioning of the Residual Network to performance. The variants like the ResNetXt, ResNet-v2, ResNet-v3, and much more (J. Cheng, et al., 2022). The original ResNet has 1000 classes and for our plant disease classification we have 5 classes and it has been modified as the five features. For the 5-class classification, the model has been replaced with only 5 classes for the plant disease classification with the following line: model_dict[“ResNet50”].fc = nn. Linear(model_dict[“ResNet50”].fc.in_features, 5). The output for the above line will be Output = Linear (2048, 5). For this the loss function is calculated as mentioned in Equation (10):

Loss = - $\sum_{i=1}^{N}\sum_{c=1}^{C} y_{i,c} \log \check{y}_{i,c}$ Eq. - 10

Where:

N = Batch size

C = Number of classes

$Y_{i,c}$ = Actual label

y = Predicted probability for class c

The Adam optimizer has been used for the model training for adaptive learning rate capabilities, which helps in the faster convergence with $\theta = \theta - \alpha . \frac{m_t}{\sqrt{v_t} + \in}$

Where:

ϴ= Model parameters (weights)

m_t, v_t = Moving averages of gradients and their squares

α = Learning rate

2.4. EfficientNetB0

The EfficientNetB0 is a family of EfficientNet Deep Learning Models which is developed by Google AI, which is designed and developed by google for higher performance and accuracy, efficiency. It has a smaller parameter when considering other models which is the main advantage of it (Lumoring & et. al., 2024). The EfficientNet models leverage a technique called compound scaling, which will uniformly scale the depth, width, and resolution to achieve better accuracy and efficiency. It is the pretrained model which consists of the 1000 classes in general, according to the problem, we can adjust it to solve it. The EfficientNetB0 uses a Mobile Inverted Bottleneck (MBConv) layers along with the Swish activation in its architecture.

2.5. Vision Transformer (ViT)

A Vision Transformer (ViT) is a deep learning model that leverages transformer architecture, which was originally designed for natural language processing (NLP), for image recognition tasks. Unlike traditional Convolutional Neural Networks (CNNs) that use convolutions to process images, the Vision Transformer treats an image as a sequence of patches and processes them similarly to how transformers handle text sequences. The key components of a Vision Transformer include image patching, positional encoding, the transformer encoder, a classification token, and the final prediction. In the image patching step, the input image is divided into fixed-size, non-overlapping patches (for example, 16x16 pixels). These patches are flattened into 1D vectors and linearly embedded into a higher-dimensional space, similar to token embeddings in NLP transformers. Since transformers lack inherent spatial awareness, positional encodings are added to each image patch embedding to retain information about the position of each patch. These positional encodings are included before passing the patch embeddings through the transformer layers.

The embedded patches, now with positional encodings, are fed into a standard transformer encoder, which consists of layers of multi-head self-attention and feed-forward neural networks. The self-attention mechanism enables the model to capture global dependencies across all image patches, irrespective of their spatial locations. The feed-forward networks are applied to each patch independently after attention is computed. Just like in NLP transformers, where a special token (like [CLS]) is used for classification, the Vision Transformer introduces a classification token, often denoted as [CLS]. This token aggregates information from all the patches and is used to predict the final class label of the image. After passing through the transformer encoder, the output corresponding to the classification token is processed by a fully connected layer that generates the class probabilities for the image. The advantages of Vision Transformers include their ability to understand global context, scalability, and flexibility. However, the challenges of using ViTs stem from their computational complexity and training efficiency. ViTs typically require large amounts of training data and computational resources to achieve optimal performance. Without extensive datasets, CNNs may still outperform ViTs, as they tend to generalize better with fewer data. Additionally, due to the complexity of the attention mechanism, Vision Transformers may require more training time, especially when handling high-resolution images.

3. GENERATIVE AI

Generative Artificial Intelligence (AI) is a subset of the Artificial intelligence systems which is designed primarily to create or generate new content for the provided input. This new content includes text, audio, video, images and other such form of media. Unlike from traditional AI, with good Improvements in technology like sensors and image analysis is easy to study. (Bhugra et al.,2024) which will typically focus on analyzing and recognizing the patterns within the data provided, generative AI models can able to produce novel outputs based on the learned data distributions. These models learn from the existing datasets which were used to train them and based on the learning from that it can able to use that information to generate new content which mimics the characteristics of the original data which is used for training and from the real world. For example, generative AI can produce realistic images, based on the prompt we ask for, it can able to generate human-like text, compose music, make videos, and much more. The potential application for the GenAI is vast. Which spans industries such as entertainment, healthcare, finance, and others.

Generative AI works by learning the underlying patterns, structures and that correlate within the data which we use to train them from which it learns all the things. The process of training typically involves the dataset with large volumes of data, in which the model iterates over the provided data to learn a probabilistic model of the distribution. Based on the learning it can able to create or generate new things, similar data which can be in the form of text, image, audio, video, to some other more complex things like the 3D models or even the synthetic voice. Generative AI can be classified into several types based on the approaches and the architectures used to generate the new content. Some of the prominent types of generative AI include the following:

3.1. Generative Adversarial Networks

A variant of the Generative AI model, known as Generative Adversarial Networks (GANs), is one of the most widely recognized and used models in the field of generative models. GAN is a emerging AI tools which reduce the dependency while training the data (Singh et al.,2024). A GAN consists of two neural networks: the Generator and the Discriminator. The Generator creates synthetic data based on random noise, attempting to generate data that closely resembles the real training data. By applying the transformation which changes the style it leads to in efficient augmentation (Lokesh et al.,2024). The Discriminator, on the other hand, evaluates the generated data, distinguishing between real and fake data produced by the Generator. During training, the Generator and Discriminator engage in a continuous "game," where the Generator strives to improve its ability to produce realistic data, while the Discriminator becomes more adept at detecting fake data. This adversarial process continues until the Generator generates data so realistically that the Discriminator can no longer reliably distinguish it from real data (Liu et al., 2020).

3.2. Transformer Models

The transformer model is the type of the Generative AI model which is considered to be one of the most advancements in the context of the generative AI, particularly when considering Natural Language Processing. The transformer model uses a mechanism of self-attention which is used to evaluate the relationships between the worlds in a sequence simultaneously, which allows it for concurrent processing and handling of long-range dependencies. The example for the transformers includes the BERT (Bidirectional Encoder Representations from Transformers) and GPT (Generative Pretrained Transformer, and the image generating model like the DALL-E.

Diffusion Model:

The Diffusion Model is the recent type of the generative AI model that has gained popularity, particularly in the domain of image and video generation. It works by the process of simulation by adding the noise to the data and then learning from it how to reverse the process and reconstruct the original data. The diffusion model starts with the random noise which then gradually refines it to produce the high-quality output of the image/video. It can generate high-capacity images/videos that show impressive performance when doing tasks like inpainting (image completion), super-resolution, and the generation of images from the description provided by the user.

Types of Diffusion Models:
Latent Diffusion Model (LDMs): Which is a type of diffusion model that operates in a lower dimensional latent space to improve the efficiency in terms of computation.
Score-Based Model: It is a variety of models that utilizes score function to guide the process of diffusion process toward the process of generating the realistic output.

3.3. Variational Auto-Encoder:

The Variational Auto-encoders is a type of Generative AI model, which is used often for the generation of new data points and learning from the latent representation, it mainly focuses on learning with a probabilistic mapping from a high-dimensional space to a lower dimensional latent space. A variational auto-encoder is, in the broadest sense, a release of the continuous latent variable model, which is a statistical framework. (Thirunavukkarasu et. al, 2022). The VAE primarily consists of the following two components in it for its working which are:

Encoders: Maps the input data into a latent dimension.
Decoders: Reconstruct the data from the latent dimension.

In general, the VAE uses the ReLU as the activation function for the convolution layer which it consists of each layer. The VAE uses the Loss function to calculate the loss during the training, and validation phase: it consists of two parts:

Reconstruction Loss: This measures how well the generated (reconstructed) image matches with the original input data, this can be mean squared error (MSE) or the binary cross-entropy (BCE) loss.
KL-Divergence Loss: It is a type of loss that encourages the latent dimension distribution to be close to a standard normal distribution, enabling smooth sampling. The loss can be defined in a combined manner in Equation (11).

Loss = Reconstruction Loss + β x KL-Divergence Loss (11)
Where:
β is a weight term that balances the two losses (Reconstruction and KL-Divergence)
The Reconstruction Loss is provided by the following Equation (12)

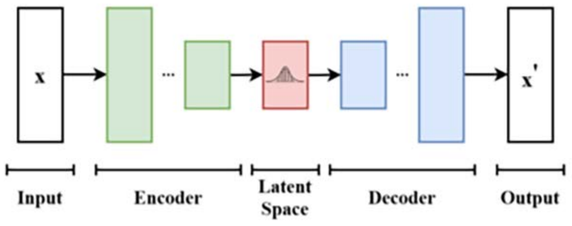
$$\mathscr{L}_{reconstruction} = MSE(x,\hat{x}) = \frac{1}{N}\sum_{i=1}^{N}\|x_i - \hat{x}_i\|^2 \tag{12}$$

Where:
N is the number of samples
x_i is the original input
$\hat{x}_i$ is the reconstructed image.
The KL-Divergence Loss is given by the following Equation (13)

$$\mathcal{L}_{KL} = -\frac{1}{2}\sum_{i=1}^{D}\left(1 + \log\sigma_i^2 - \mu_i^2 - \sigma_i^2\right) \tag{13}$$

Where:
D is the dimensionality of the latent space
μ_i *and* σ_i^2 is the mean and variance of the latent variables from the encoder.
The architecture diagram of the variational autoencoder is provided in Fig. 5

Figure 5. Architecture of the variational auto-encoder



4. LITERATURE SURVEY

(Legg et al. 2015) conducted an extensive review of cassava virus diseases, focusing on their biology, epidemiology, and management strategies. The study discussed the major viruses affecting cassava, such as Cassava Mosaic Virus (CMV) and Cassava Brown Streak Virus (CBSV), outlining their transmission mechanisms, impact on crop yield, and geographical spread. The authors highlighted integrated disease management approaches, including the use of resistant cassava varieties, vector control, and phytosanitary measures. Although not focused on machine learning or accuracy metrics, the research emphasized the effectiveness of breeding programs and biotechnological interventions in reducing disease incidence and improving cassava productivity in affected regions.

(Anitha and Saranya, 2022) Suggested a deep learning approach for identifying and detecting cassava leaf diseases, focusing on improving classification accuracy to support farmers in timely disease management. The authors utilized a Convolutional Neural Network (CNN) model to categorize leaf diseases of cassava into multiple categories, leveraging a dataset containing images of cassava leaves infected by various diseases. Their methodology involved pre-processing the images to improve feature extraction, followed by training the CNN model with optimized hyperparameters. They evaluated their model using performance metrics such as accuracy, precision, recall, and F1-score. The result of the experiment demonstrated that the CNN model achieved a classification accuracy of 90%, the best-performing traditional machine learning algorithms. The study concludes that deep learning models, specifically CNNs, are effective in accurately identifying cassava leaf diseases, which can significantly contribute to precision agriculture by enabling early disease detection and reducing crop losses.

(Moupojou et al. 2023) Introduced FieldPlant, a comprehensive dataset designed to facilitate the detection and classification are performed to plant diseases using deep learning models. The dataset comprises a diverse collection of field plant images captured in real-world agricultural settings, covering multiple plant species and various disease conditions to address the challenges of plant disease detection in dynamic environments. The authors aimed to enhance the accuracy of automated identification of plant disease by providing a rich dataset that captures the complexity of natural field conditions, such as varying lighting, backgrounds, and plant orientations. They employed several state-of-the-art deep learning models, consisting of Convolutional Neural Networks (CNNs) and transfer learning techniques, to benchmark the dataset, achieving high classification performance with models like EfficientNet and ResNet. Their experiments demonstrated that the FieldPlant dataset supports robust training, resulting in improved generalization and accuracy in disease detection tasks, with top models achieving classification accuracies exceeding 90%.

(Paiva-Peredo, 2022) explored the application of deep learning techniques for classifying cassava leaf diseases using an unbalanced field dataset. The study focused on addressing the challenges posed by imbalanced data distributions, which are common in real-world agricultural datasets. The author utilized Convolutional Neural Networks (CNNs) to develop a robust classification model, applying techniques such as data augmentation and class balancing methods to mitigate the effects of data imbalance. The model was trained on a dataset comprising images of cassava leaves affected by various diseases, with a particular emphasis on enhancing the model's ability to generalize across minority classes. Experimental results showed that the deep learning approach achieved satisfactory performance, with the implementation of data balancing techniques significantly improving classification accuracy and recall, particularly for underrepresented disease categories.

(Thaiyalnayaki et. al., 2022) Developed an automatic cassava disease classification system using Convolutional Neural Networks (CNNs) combined with data augmentation techniques. To address the limitations of small and variable cassava leaf datasets, they applied augmentation methods like rotation, flipping, and scaling to increase training data diversity. The CNN model, trained on cassava leaf images with various diseases, achieved high accuracy across multiple disease classes. The results showed that data augmentation improved model performance by reducing overfitting and enabling better generalization to real-world variations. The study concluded that this approach offers an effective, scalable solution for cassava disease detection in agriculture.

(Sapre et al. 2023) proposed a disease classification system for cassava plants using an Artificial Neural Network (ANN). The authors trained the ANN model on cassava leaf images affected by different diseases, focusing on developing an accurate classification framework for plant disease detection. They used image preprocessing techniques, including normalization and feature extraction, to enhance model performance. The ANN achieved a classification accuracy of 92.3%, demonstrating its effectiveness in detecting and categorizing cassava diseases. The study highlighted the potential of ANNs in agricultural applications, offering a robust solution for automated disease detection in cassava plants.

(Satoto et. al., 2021) proposed a Region Proposal Convolutional Neural Network (R-CNN) with data augmentation for identifying cassava leaf diseases. The authors integrated R-CNN with augmentation techniques like rotation, scaling, and flipping to enhance the diversity of training data and improve model robustness. The model was trained to detect and classify various cassava leaf diseases from input images. The experimental results showed a significant improvement in classification performance, with the model achieving an accuracy of around 99%. This approach demonstrated the effectiveness of combining R-CNN and data augmentation in improving cassava disease detection accuracy in agricultural applications.

(Applalanaidu and Kumaravelan, 2021) provided a comparative analysis of various machine learning (ML) and deep learning (DL) algorithms for plant leaf disease detection and classification. They reviewed algorithms such as Convolutional Neural Networks (CNNs), Support Vector Machines (SVMs), and decision trees, noting the superior performance of CNNs in achieving high accuracy, often up to 98%. The paper also highlighted several areas for future research, including diagnosing specific stages of plant diseases, controlling chemical applications based on accurate disease quantification, and developing an online system for plant disease identification. Further investigations were proposed in detecting nutrient deficiencies, analyzing the back of the leaf, using real-time images for disease detection, and addressing challenges like mixed lighting conditions and complex backgrounds. Additionally, the authors suggested exploring automatic disease severity estimation, extending research to other plant parts (e.g., stems), and enhancing pest recognition methods to improve overall plant health monitoring systems.

(Soujanya et al., 2023) proposed an efficient cassava leaf disease classification model using the VGG16 Convolutional Neural Network (CNN) architecture. The study aimed to leverage the deep feature extraction capabilities of VGG16 for the accurate identification of various cassava leaf diseases. The authors applied preprocessing techniques such as resizing and normalization to prepare the dataset, followed by data augmentation to enhance model robustness. The VGG16 model was trained on cassava leaf images, achieving a classification accuracy of 96.8%. The study concluded that VGG16 is highly effective for plant disease classification, offering a reliable solution for early disease detection and management in cassava crops.

(Askr et al., 2024) introduced an explainable ResNet50 model enhanced with copula entropy for predicting cotton plant diseases. The approach focused on making deep learning models more interpretable while maintaining high prediction accuracy. The authors used the ResNet50 architecture due to its strong feature extraction capabilities, integrating copula entropy to improve the model's decision-making transparency. This method enabled a better understanding of the relationships among input features and model predictions. The model was trained on a cotton disease dataset, achieving a good accuracy of 95.5%. The study demonstrated the effectiveness of combining explainable AI techniques with deep learning for precise and interpretable plant disease diagnosis.

(Thirunavukkarasu et al., 2022) proposed an efficient deep learning approach for plant disease detection using Convolutional Neural Networks (CNNs). The study aimed to enhance the accuracy of disease classification by leveraging advanced deep-learning techniques. The authors applied image preprocessing and data augmentation to improve the diversity of the training set, followed by using a CNN model optimized for plant disease identification. The model was trained on a diverse dataset of plant leaf images affected by various diseases. Experimental results showed that their approach achieved a high classification accuracy of 97.2%, demonstrating the effectiveness of deep learning in automated plant disease detection and supporting precision agriculture.

(Aravind and Harini, 2021) Developed a cassava leaf disease classification model using deep learning techniques. They employed a Convolutional Neural Network (CNN) architecture to automatically identify and classify cassava leaf diseases from images. The study involved preprocessing steps like image resizing, normalization, and data augmentation to enhance model training and improve accuracy. The CNN model was trained on a dataset of cassava leaf images representing various diseases, achieving a classification accuracy of 94.7%. The results demonstrated that the deep learning approach is effective for plant disease detection, offering a reliable answer for early diagnosis and management of cassava leaf diseases in agricultural fields.

Mathulaprangsan and Lanthong (2021) developed a cassava leaf disease recognition system using Convolutional Neural Networks (CNNs). The study aimed to automate the detection of cassava diseases to support farmers in early diagnosis. The authors utilized a CNN model trained on a dataset of cassava leaf images, incorporating preprocessing techniques like image resizing, normalization, and data augmentation to improve model performance. The CNN architecture was optimized for classifying multiple cassava leaf diseases, achieving a high classification accuracy of 93.5%. The results demonstrated the effectiveness of CNNs in plant disease recognition, providing a reliable solution for enhancing agricultural disease management practices.

5. METHODOLOGY

The data set contains 17,938 of images which is obtained from the Kaggle dataset, with 5 classes which are split into 70:15:15 where 70 percentage for training, 15 percentage for validation, and 15 percent for Testing the model by constructing the Variational Auto-encoders which is generative Artificial intelligence. The 70% of the Training Image set consists of 5 classes (0 to 4) with 625 images, 1,259 images, 1,404 images, 7,738 images, and 1,530 images. The validation image is split into 5 classes (0 to 4) which consist of about 153 images, 288 images, 295 images, 1,630 images, and 325 images respectively. The 15% of the testing dataset is split into the 5 classes (0 to 4) which consist of about 143 images, 284 images, 294 images, 1,659 images, and 311 images respectively. The single image dimension consists of 800 X 600 with a width is 800, a height of 600 pixels, and horizontal and vertical resolutions is 96 dpi(Dots Per Inch) respectively. The dataset has been collected from the natural sunlight during the daytime with medium lighting. The model is designed for training by the following components which were encoder for the purpose of input data mapping and for the distribution of the latent, another component will be considered as a Decoder (Kingma et. al., 2019) which will perform with the help of the representation of the latent to make reconstruction of the given input data. Based on some important modules of VAE will be considered during the training the model which is called latent space or latent dimension for the usage to make image compression (Chamain et. al., 2022) of the input data, VAE consists of encoders which will allow the control random of input by preparing the distribution with help of mean and variance from the space.

Another module that is performed in the VAE is a minimization of the Divergence of the Kullback-Leibler which will make the module VAE understand the relations of the distribution of the latent very closely with the help of the already defined values distribution with the help of the Gaussian function. Another process is to check the loss of reconstruction that identify the difference among the entered data input to the reconstructed images to make the model prepare a good understanding of features that were underlying in data The last process is Sampling where the VAE has the sample of latent distribution from the process of decoder it will make good facilitate the generation of the data. The overall flow of the model is represented in Fig.6

Figure 6. Image showing the flowchart

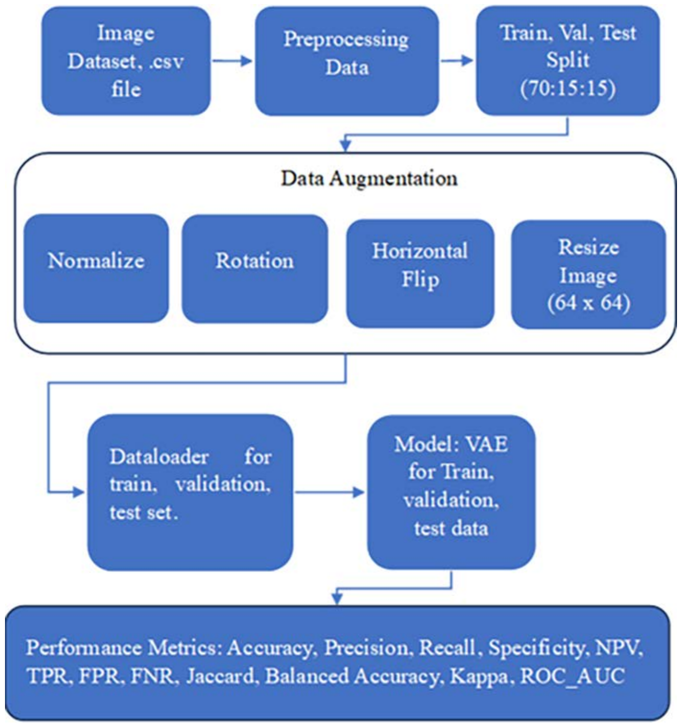


Image Dataset, .csv file
Preprocessing Data
Train, Val, Test Split (70:15:15)
Data Augmentation
Normalize
Rotation
Horizontal Flip
Resize Image (64 x 64)
Dataloader for train, validation, test set.
Model: VAE for Train, validation, test data
Performance Metrics: Accuracy, Precision, Recall, Specificity, NPV, TPR, FPR, FNR, Jaccard, Balanced Accuracy, Kappa, ROC_AUC

5.1. Data Cleaning

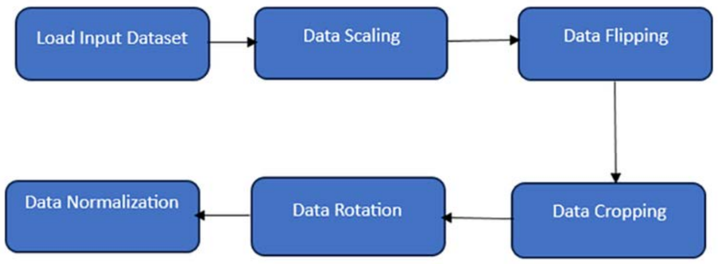
The first process is to verify the input data processing which will help to clean the values where to clean the null values which is important to Train any module to attain good output or results particularly when VAE has done the representation of latent for some meaningful learning from input. Considering our dataset which consists of the images and the .csv file, we verify and ensure that all the images present in the image directory are also available in the CSV file, if the extra information is available in the dataset means it is removed as per the image files which our dataset consist of. Some methods are adapted to make a good data clean process which is Scaling for the normalize the following uploading input data for it range of binary values. It is used to make good convergences while training by changing the input data, it also helps the model to identify the dominating learnings that prevent the big ranges of features or attributes, Scaling verifies before working with large data like images and tables. The next process is Flipping to changing the input images or from data set by Horizontal and vertical to avoid overfitting and increase the efficiency in generalization, Performs the new method called invariance which is suitable for image classification or reconstruction with spatial Orientation to the module.

The next process cropping the images by random or using parts from the center which helps to prevent overfit with changing the input parts also helps the model to identify the robustness unless the model predicts with the same positions. The next process is Rotation which will change the input data with the rotation of some randoms with some already defined range, it also helps in invariance which can be used in orientation with the object which is irrelevant and more diversities to the dataset which model is ready for Training. The next process is Normalization which makes the input data to able to adjust with the distribution of scale, important while calculating the gradient which can get output of stable and improves the models to fast converges. The next process is Adding noise to the model for the robustness of the data which the decoder will identify the differences between the input data and reconstruct from methods like Gaussian or salt-and-paper methods to avoid overfitting.

5.2. Data Augmentation

Data augmentation increases the diversity of the training data without collecting new samples. Techniques such as scaling, flipping, and rotating introduce variations in the input data, forcing VAEs to learn the representation of latent dimension. For the dataset, we applied the following augmentation flipping the images which ensures that the model becomes orientation invariant, rotates, and resizes (64 x 64), while adding random noise promotes robustness by training the decoder to reconstruct clean data from noisy inputs. The flowchart describes the Data Augmentation Process after the data has been preprocessed in Fig. 7.

Figure 7. Flow chart describing data augmentation Process



5.3. Learning Rate

Learning price scheduling is any other effective enhancement to improve VAE training. The choice of gaining knowledge of price notably influences the convergence speed and balance of optimization. Using a static getting-to-know price during training would possibly cause suboptimal results, as the best mastering rate modifications as the version approaches convergence. The learning rate for the VAE model has been chosen according to the optimizer's work.

5.4. Normalization

Normalization methods, like **Batch Normalization (BN)**, play a crucial role in stabilizing and accelerating the training of deep learning models. BN normalizes the activations of each layer, ensuring they have a mean of zero and a standard deviation of one, which reduces internal covariate shifts and prevents large weight updates. This stabilization helps in faster convergence, reduces the risk of vanishing or exploding gradients, and improves the model's generalization capabilities, making it particularly useful in deep architectures like Variational Autoencoders (VAEs). In VAEs, normalization not only speeds up training but also works alongside regularization techniques such as **dropout**, which helps the model generalize better by preventing overfitting. Together, these techniques contribute to more efficient and robust models, especially when working with large datasets.

5.5. Model Loss

In a Variational Autoencoder (VAE), the model loss plays a crucial role in guiding the network during training by evaluating how well the model is learning to generate data. Regarding the calculation of the Variational Auto-encoders, we described the model loss previously in equations (11)-(13).

5.6. Training VAE Model

5.6.1. Training Phase

Training and validation are essential phases in building a sturdy Variational Autoencoder (VAE). During education, the encoder learns to map input statistics into a latent area represented through probabilistic distributions, even as the decoder reconstructs the entry from sampled latent vectors. The optimization purpose is to reduce the Evidence Lower Bound (ELBO), which is composed of the reconstruction loss and the Kullback-Leibler (KL) divergence loss. Reconstruction loss measures how correctly the model recreates the enter, at the same time as KL-divergence guarantees the latent area distribution approximates a general regular distribution, selling generalization and regularization. Even in the training phase, we trained the model with about 7 various optimizers which were the Adam, LAMB, NovoGrad, RAdam, RMSProp, LARS, and ADAGRAD, whose results will be discussed under the Results and Discussion section.

5.6.2. Validation Phase

The validation phase is a crucial phase in monitoring the model performance during the time of training, it provides a checkpoint to ensure the model is generalizing well to the unseen data. It helps in detecting the issues in the model if it has any like overfitting (the model correctly predicts the trained data but wrongly predicts the unseen data) and underfitting (the underlying pattern that the models failed to predict). With the help of the validation set, the model can be tuned based on the performance over the unseen data, improving its generalizing ability before finalizing it.

5.6.3. Testing Phase

It is a phase which will be done after the training and validation phase is completed on the model, also it is the checking out section of a model to measure the performance of the overall model completely from the unseen/hidden data during the phase of training, from the original dataset, and checked against the unseen data to check how it performs with that data. This phase assesses the generalization capability of the version and presents the impartial estimate of its reconstruction satisfactory and latent area regularization. During trying out, the input statistics is exceeded through the encoder and decoder with no gradient updates.

6. PERFORMANCE METRICS OF THE MODEL

The following metrics for the constructed model performance are evaluated, which is very crucial to identify the critical things which allow us to visualize the representation where the VAE model while reconstruction data of the input and it will used to identify the good practice to saw the dimension of the latent. The following metrics which are given below are considered.

6.1. Accuracy

The following accuracy metrics are very important for the model which helps to identify the overall right prediction which from all predictions by the model. The formula provided in Equation (14)

$$Accuracy = \frac{True\ Positive\ rate + True\ Negative\ Rate}{True\ Positive + True\ Negative + False\ Positive + False\ Negative} \tag{14}$$

6.2. Precision

The following metrics which will validate the only positive values with all the predicted values from the model. The formula provided in Equation (15)

$$Precision = \frac{True\ Positive\ rate}{True\ Positive + False\ Positive} \tag{15}$$

6.3. Recall

The following metrics have calculated the total number of positive values which is important for the model to identify positive values. The formula provided in Equation (16)

$$Recall = \frac{True\ Positive\ rate}{True\ Positive + False\ Negative} \tag{16}$$

6.4. Specificity

The following metrics have calculated the total number of negative values which is important for the model to identify imbalances from the data inputs. The formula provided in Equation (17)

$$Specificity = \frac{True\ Negaive\ rate}{True\ Negative + False\ Positive} \tag{17}$$

6.5. Negative Predictive Value

The following metrics have calculated a number of negative values which is important for the model to identify in low-frequency outcomes. Formula provided in Equation (18)

$$NPV = \frac{True\ Negative\ rate}{True\ Negative + False\ Negative} \tag{18}$$

6.6. F1 Score

The following metrics have calculated the metrics like precision and recall from the model which us to validate the performance in noisy classes. The formula provided in Equation (19)

$$F1\ Score\ =\ 2\ X \frac{Precision\ X\ Recall}{Precision + Recall} \tag{19}$$

6.7. Matthews Correlation Coefficient (MCC)

The following metrics have calculated the full elements of the confusion matrix to understand the constructed model's performance. The formula provided in Equation (20)

$$MCC = \frac{True\ Positive}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{20}$$

6.8. Jaccard Index

The following metric Jaccard Index calculates the overlap between the actual (true) positive values and the model class that is predicted. The formula provided in Equation (21)

$$Jaccard = \frac{True\ Positive}{True\ Positive + False\ Positive + False\ Negative} \tag{21}$$

6.9. Balanced Accuracy

The following metrics have calculated the metrics like all positive and negative values from the model which as noisy dataset. The formula provided in Equation (22)

$$Balanced\ Accuracy = \frac{Snsitivity + Speficicity}{2} \tag{22}$$

Where: Sensitivity = True Positive Rate (TPR), Specificity = True Negative Rate (TNR)

6.10. False Positive Rate (FPR)

The following metrics have calculated the negative values which are predicted as positive in the model. The formula provided in Equation (23)

$$False\ Positive\ Rate = \frac{False\ Positive}{False\ Negative + True\ Positive} \tag{23}$$

6.11. False Negative Rate (FNR)

The following metrics have calculated the positive values which are ignored by the model which is important to identify the correct response otherwise it would be risk. The formula provided in Equation (24)

$$\text{False Negative Rate} = \frac{\text{False Negative}}{\text{False Negative} + \text{True Positive}} \tag{24}$$

6.12. ROC-AUC (Area Under Operating Characteristics Curve)

The following metrics have shown the exchange values among true positives and false positives with some certain conditions. The AUC has a range of 0.5 to 1.0, where 0.5 is considered as performance of random and 1.0 considered as performance of perfect

6.13. Confusion Matrix

The following parameters will be very useful in computing important metrics like recall, precision, and other metrics.

Whereas

TP = True Positive (Correctly predicted positive reconstructed values)

TN= True Negative (Correctly predicted negative reconstructed values)

FP= False Positive (Incorrectly predicted positive values from reconstructed data)

FN= False Negative (Incorrectly predicted negative values from reconstructed data)

6.14. Cohen's Kappa

The following metrics have calculated the statistics value of which bond of two models and it is used the identify the frequency of consistent values from the prediction which can be far more than the expected value of chances of random. The formula provided in Equation (25)

$$\text{Kappa} = \frac{P_0 - P_e}{1 - P_e} \tag{25}$$

7. RESULTS AND DISCUSSIONS

7.1. Deep Learning Model Performance

Based on the training of deep learning models, the values presented in Table 1 show that the EfficientNet model performs well compared to other models. While Visual Transformers show potential, they perform poorly on this dataset according to the key metrics in Table 1. In contrast, ResNet, CNN, and VGG16 deliver average performance. Overall, the key metrics indicate that EfficientNet achieves the best results during the training phase.

Table 1. Training metrics of deep learning models

	(CNN)	VGG16	ResNet	EfficientNet	Visual Transformers
Accuracy (%)	89.96	76.53	94.91	94.70	61.76
Precision (%)	89.82	74.55	94.90	94.70	44.68
Recall (%)	89.69	76.53	94.91	94.70	61.67
F1-Score (%)	89.96	75.13	94.90	94.69	48.72

During the validation phase, the same four performance metrics were considered, with EfficientNet demonstrating superior performance compared to other models. ResNet, VGG16, and CNN also show good performance but do not surpass EfficientNet, with their results falling slightly above average in the validation metrics. In contrast, Visual Transformers exhibit the lowest values across all models, as shown in Table 2.

Table 2. Validation metrics of deep learning models

	(CNN)	VGG16	ResNet	EfficientNet	Visual Transformers
Accuracy (%)	73.54	75.13	77.21	82.65	60.68
Precision (%)	71.72	70.29	77.00	82.71	40.49
Recall (%)	73.54	76.74	77.22	82.65	60.68
F1-Score (%)	70.98	65.96	76.81	84.06	50.92

In the testing phase, EfficientNet achieves the best performance among all models. CNN follows with strong test results, though it does not outperform EfficientNet. ResNet and VGG16 also deliver good performance, but they fall short of EfficientNet, showing above-average results. Lastly, Visual Transformers have the lowest values across all models, as presented in Table 3.

Table 3. Testing metrics of deep learning models

	(CNN)	VGG16	ResNet	EfficientNet	Visual Transformers
Accuracy (%)	73.91	72.28	72.84	80.16	62.63
Precision (%)	73.59	68.00	76.11	80.21	72.59
Recall (%)	73.91	72.28	72.84	80.16	62.36
F1-Score (%)	73.61	69.58	73.68	79.73	49.01

7.2. VAE Model and its Performance

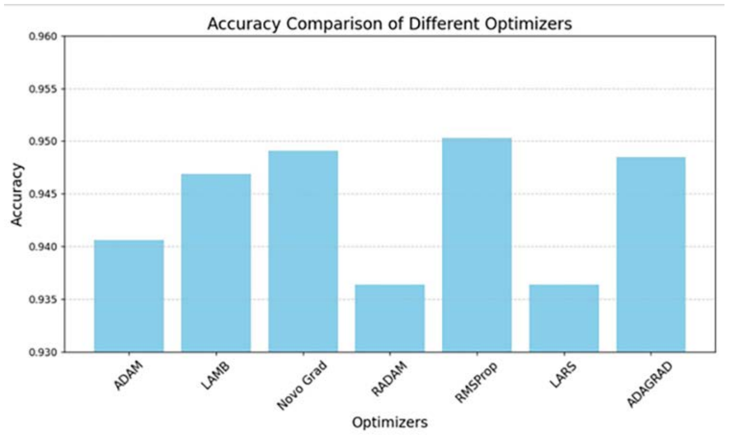
The deep learning Models are good but not to the expectation so the generative AI model Variational Auto-encoder (VAE) is used to predict the same data set, so the results are good when compared to the deep learning models. In Variational Auto-encoder different types of optimizers are used to predict which optimizer has a very good performance based on various metrics as shown the Table 4 which are calculated for the training phase. The RMSProp has the best overall performance than another optimizer, the representation of the other optimizer is good but not better than the RMSProp optimizer in the training phase.

Table 4. Performance metrics during the training phase with various optimizers

	ADAM	LAMB	Novo Grad	RAdam	RMSProp	LARS	ADAGRAD
Accuracy	0.9406	0.9469	0.9491	0.9364	0.9503	0.9364	0.9485
F1 Score	0.2992	0.4725	0.5140	0.0000	0.5122	0.0002	0.4778
Precision	0.7271	0.6885	0.6500	0.0000	0.7273	0.4019	0.7174
Recall	0.1229	0.3597	0.4251	0.0000	0.3953	0.0001	0.3581
Specificity	0.9949	0.9885	0.9845	1.0000	0.9895	1.0000	0.9901
NPV	0.9600	0.9550	0.9987	0.9345	0.9575	0.9345	0.9989
TPR	0.4000	0.3300	0.3500	0.0000	0.3800	0.0000	03500
FPR	0.0100	0.0100	0.7000	1.0000	0.0100	0.0000	0.0089
FNR	0.6000	0.6890	0.0100	1.0000	0.6500	1.0000	0.7080
MCC	0.5311	0.5000	0.5050	0.0000	0.5000	0.0040	0.5000
Jaccard Index	0.3512	03000	0.3459	0.0000	0.3000	0.0000	0.3138
ROC AUC	0.6929	0.6750	0.7048	1.0000	0.6800	0.5000	0.6741
Balanced Accuracy	0.6919	0.6750	0.7048	1.0000	0.6800	0.5000	0.6741
Kappa	0.4900	0.4000	0.4884	0.0000	0.4500	0.0000	0.4538

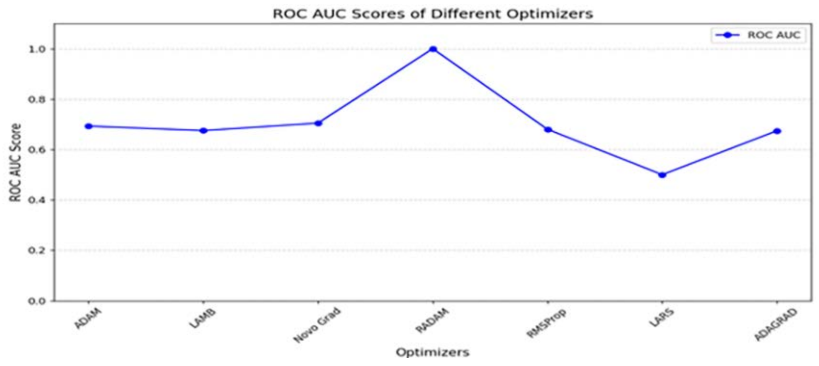
Fig.8 represents the accuracy of the training phase which is a critical metric for identifying the model's overall performance. From this figure, it can be witnessed that RMSProp optimizer has the highest accuracy than other optimizers like Novo Grad, Lamb, and AdaGrad optimizer. These optimizers provided good accuracy however lesser than the RMSProp optimizer whereas other optimizers like RAdam, Lars, and Adam are performing less than the other optimizers.

Figure 8. The bar chart showing the comparison among accuracy across different optimizers in the training set



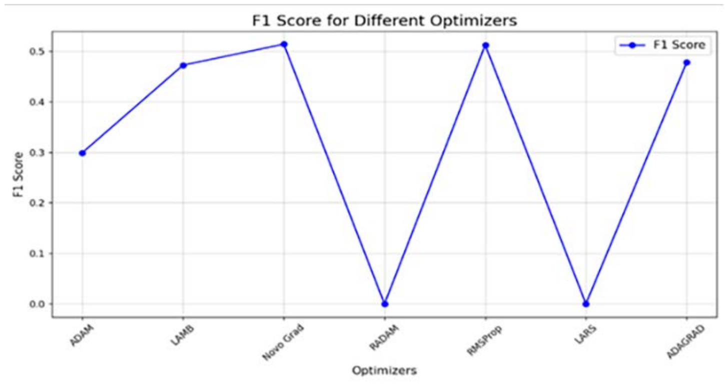
Based on the above Table. 4 which consists of the performance metrics of the VAE with the different optimizers, the graph which represents the ROC-AUC accuracy for it is shown in Fig. 9.

Figure 9. The plot showing the ROC AUC Scores during the training phase



From the above plot, we can infer that the RAdam achieves the highest ROC AUC score of 1.0, which indicates that it performed exceptionally well among the classes, the other optimizers perform moderately including the Novo Grad, ADAM show similar performance as the score is similar, the LAMB optimizer has a slightly lower performance compared to ADAM but still, it maintains a decent score around the value of 0.6. The other optimizers show he Lower performance among the above optimizers. But by considering all other metrics NovoGrad is a preferable one. Likewise, from Table 4 which consists of the performance metrics of the VAE with the different optimizers we plot the graph which represents the training phase F1-score as presented in Fig. 10.

Figure 10. The plot showing the training phase F1-Score



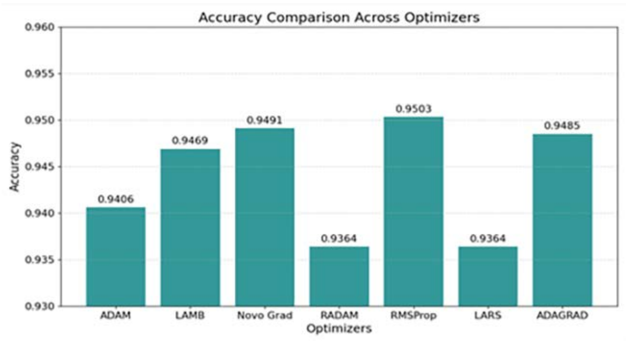
Considering both the ROC AUC scores from the previous analysis and the F1 Scores from this graph, Novo Grad remains the most consistent and effective optimizer for achieving balanced and accurate predictions. Similar to Table 4, we present Table 5 which is based on the Validation Phase of the Variational Auto-encoder Model, the validation phase also consists of the various optimizers.

Table 5. Performance metrics during the validation phase with various optimizers

	ADAM	LAMB	Novo Grad	RAdam	RMSProp	LARS	ADAGRAD
Accuracy	0.9406	0.9469	0.9491	0.9364	0.9503	0.9364	0.9485
F1 Score	0.5332	0.4725	0.5140	0.0000	0.5122	0.0002	0.4778
Precision	0.7271	0.6885	0.6500	0.0000	0.7273	0.4019	0.7174
Recall	0.1229	0.3597	0.4251	0.0000	0.3953	0.0001	0.3581
Specificity	0.9949	0.9885	0.9845	1.0000	0.9895	1.0000	0.9901
NPV	0.9662	0.9500	0.0000	0.9365	0.9575	0.9365	0.0000
TPR	0.4042	0.3500	0.0000	0.0000	0.4000	0.0000	0.0000
FPR	0.0111	0.0120	0.0000	1.0000	0.0100	0.0000	0.0000
FNR	0.5002	0.6500	0.0000	1.0000	0.6500	1.0000	0.0000
MCC	0.5123	0.5000	0.0000	0.0000	0.5000	0.0055	0.0000
Jaccard Index	0.3500	0.3000	0.3459	0.0000	0.3500	0.0000	0.3138
ROC AUC	0.6992	0.6750	0.7048	1.0000	0.6800	0.5000	0.6741
Balanced Accuracy	0.6992	0.6750	0.7048	1.0000	0.6800	0.5000	0.6741
kappa	0.4999	0.4000	0.4884	0.0000	0.5000	0.0000	0.4538

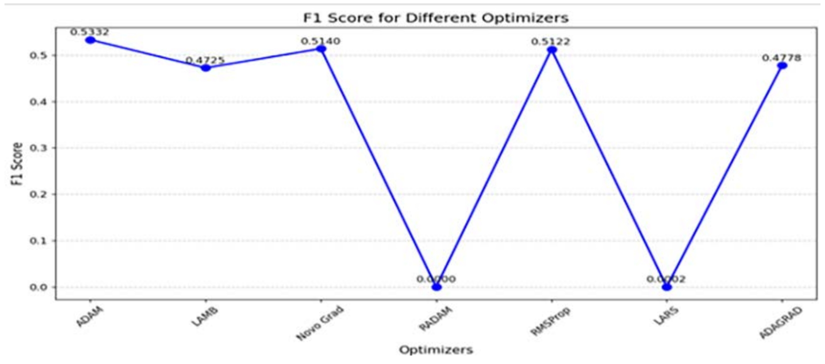
Based on the table.5 which is calculated for the validation phase the RMSProp has the best overall performance than another optimizer in the given performance metrics, the representation of the other optimizer is good but not better than the RMSProp optimizer in the Validation phase. We now present a bar plot comparing the accuracy across different optimizers, shown in Figure 11. In the validation phase, the accuracy which is a key metric for evaluating the model's overall performance, the RMSProp optimizer achieves the highest accuracy among all optimizers. The NovoGrad, Lamb, and AdaGrad optimizers also show good accuracy, though slightly lower than RMSProp. Meanwhile, the RAdam, LARS, and Adam optimizers perform less effectively, with accuracy values lower than the other optimizers.

Figure 11. The Bar chart showing the comparison among the accuracies across different optimizers in the validation set



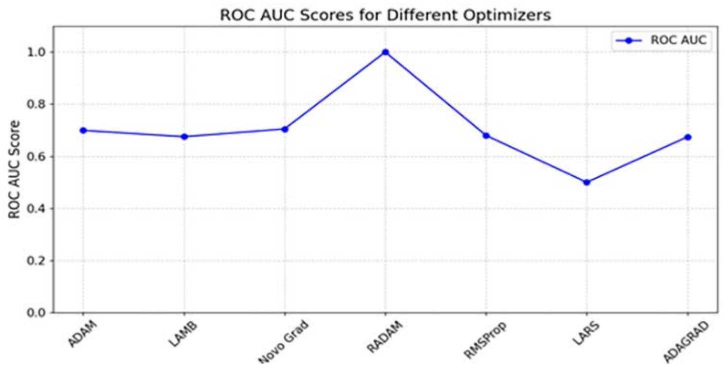
In addition to accuracy, the F1 Score is another crucial metric used to assess the model's overall performance. From the graphical representation in Fig. 10, it is clear that the Adam, RMSProp, and NovoGrad optimizers exhibit strong performance in terms of F1 Score during the Training Phase. Similarly, the F1 Score graph for the Validation Phase has been plotted and is shown in Fig. 12 below.

Figure 12. Validation F1-Score for optimizers



The same graph has now been plotted, considering the ROC-AUC scores for different optimizers in the VAE model, as shown in Fig. 13 below.

Figure 13. ROC AUC-Score for optimizers in the validation phase



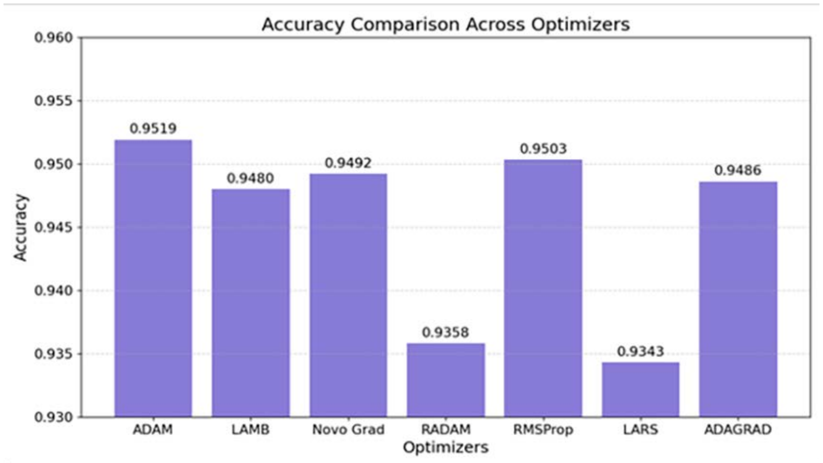
Similar to the previous two tables (Table 4 and Table 5), Table 6 presents the results based on the Testing Phase of the Variational Autoencoder (VAE) model. This table also includes the various optimizers used during the validation phase, as shown below:

Table 6. Performance metrics during the testing phase with various optimizers

	ADAM	LAMB	Novo Grad	RAdam	RMSProp	LARS	ADAGRAD
Accuracy	0.9519	0.9480	0.9492	0.9358	0.9503	0.9343	0.9486
F1 Score	0.5211	0.4728	0.5046	0.0000	0.5092	0.0003	0.4755
Precision	0.7271	0.6822	0.6411	0.0371	0.7231	0.4020	0.7256
Recall	0.4142	0.3618	0.4160	0.0000	0.3929	0.0001	0.3536
Specificity	0.9949	0.9884	0.9845	1.0000	0.9894	1.0000	0.9906
NPV	0.9616	0.9574	0.9621	0.9358	0.9587	0.9343	0.9560
TPR	0.4142	0.3618	0.4160	0.0000	0.3929	0.0001	0.3536
FPR	0.0118	0.0116	0.0155	0.0000	0.0106	0.0000	0.0094
FNR	0.5858	0.6382	0.5840	1.0000	0.6071	0.9999	0.6464
MCC	0.5168	0.4733	0.4915	-0.0007	0.5106	0.0062	0.4843
Jaccard Index	0.3523	0.3096	0.3347	0.0000	0.3415	0.0001	0.3119
ROC AUC	0.7012	0.6751	0.7003	0.5000	0.6912	0.5001	0.6721
Balanced Accuracy	0.7012	0.6751	0.7003	0.5000	0.6912	0.5001	0.6721
kappa	0.4975	0.4482	0.47900	-0.0000	0.4854	0.0002	0.4518

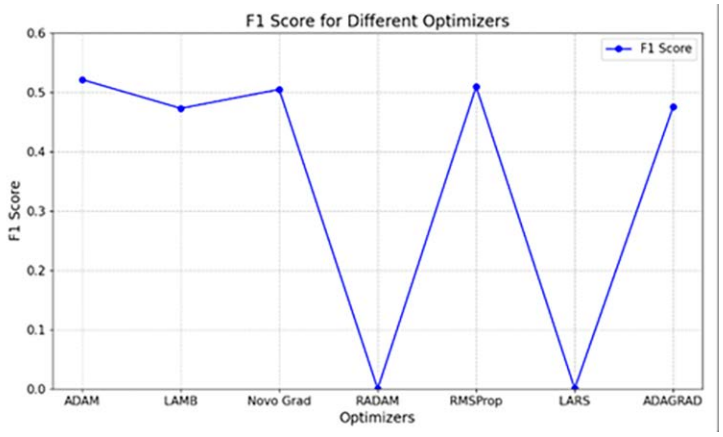
Based on the data in Table 6, we have plotted a bar chart comparing the accuracies of different optimizers, as shown in Fig. 14. In the Testing Phase, which is a critical metric for assessing the model's overall performance, the RMSProp and Adam optimizers achieve the highest accuracy. NovoGrad, Lamb, and AdaGrad also show good accuracy, though they fall slightly behind RMSProp. Meanwhile, the RAdam and LARS optimizers demonstrate lower performance compared to the others.

Figure 14. The Bar chart showing the comparison among the accuracies across different optimizers in the test set.



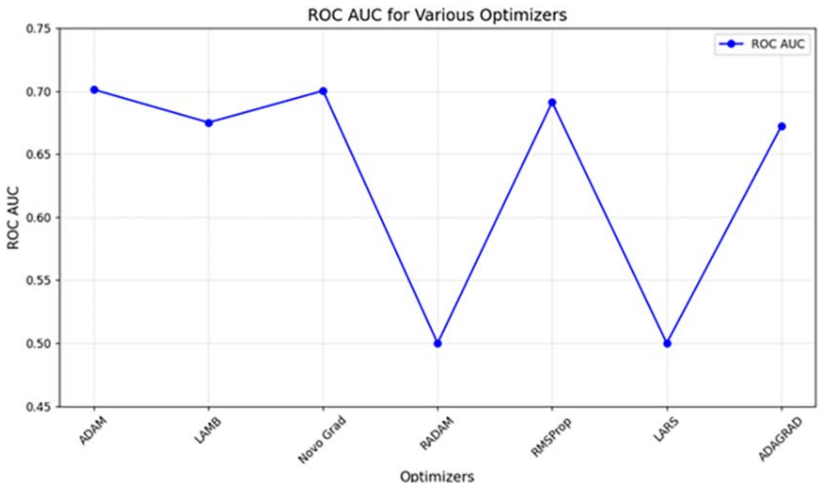
In addition to accuracy, the F1 Score is another important metric used to evaluate the model's overall performance. From the graphical representation in Fig. 15, it is evident that the NovoGrad and RMSProp optimizers demonstrate strong performance in terms of the F1 Score during the Testing Phase.

Figure 15. F1-Score Comparison over test dataset



From Fig. 13, the graphical representation indicates that for the ROC-AUC score metric, the RMSProp and NovoGrad optimizers outperform others, showing higher values, which is a critical metric. However, the overall performance of the RAdam optimizer falls short of expectations. In the Testing Phase, Fig. 16 provides a detailed interpretation of the optimizers that performed well in terms of the ROC-AUC score.

Figure 16. ROCAUC Curve for various optimizers among the test dataset

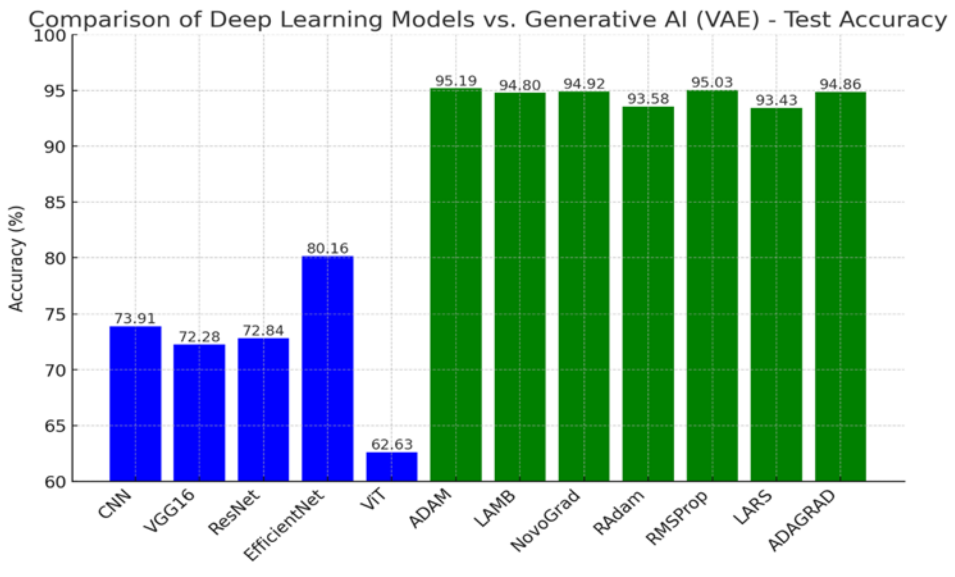


The significance that can be seen based on the above two charts is as follows: The F1 score (Fig. 15) evaluates the balance in the data by considering the relationship between the precision and the recall. For our dataset using the ADAM, Novo Grad, and the RMSProp optimizers we can see that they perform relatively well, maintaining F1 scores above the value of 0.45. The RADAM and LARS show poor performance almost near zero, which suggests that the optimizers RADAM and LARS are not suitable for this task. The ROCAUC Score (Fig. 16) measures how well the model distinguishes between the available classes. The ADAM, Novo Grad and RMSProp have the highest AUC value (~0.7) indicating that they performed well with the dataset and the classes, while RADAM and LARS show poor performance (~ 0.5) from which we can infer that the model's weak discriminative ability with those optimizers. From the above F1 Score and the ROCAUC graph, we can conclude that the ADAM and the Novo Grad are the most effective optimizers for our chosen dataset.

7.3 Comparison Between Deep Learning and Generative AI

From Fig.17 which compares the test accuracy of Deep learning and Generative AI We can understand the best test accuracy is generated by Generative AI with the most suitable optimizer (RMSProp(95.03%) and Adam(95.19%)) and In deep learning method only the EfficientNet has the good Testing Accuracy(80.16%) Based on the Dataset we conclude the Generative AI (VAE) gives the best result to find the generative Modeling and good Anomaly detection.

Figure 17. Comparison of deep learning and generative AI test accuracy



8. CONCLUSION

This research explores the fundamentals of Deep Learning, its algorithms, and the architectures that power its functioning. It also delves into the concept of Generative AI, discussing its various types, including Generative Adversarial Networks (GANs), Transformer Models, and Variational Autoencoders (VAEs), with a focus on how these models operate. The study then applies these principles to develop a model for Cassava Plant Disease Prediction, specifically a 5-class classification model. Throughout the training, validation, and testing phases, several metrics were considered to evaluate the model's performance: Accuracy, F1 Score, Precision, Recall, Specificity, NPV, TPR, FPR, FNR, MCC, Jaccard Index, ROC AUC, Balanced Accuracy, and Kappa. The evaluation, based on three primary performance indicators—Accuracy, F1 Score, and ROC AUC—reveals that the RMSProp optimizer performs the best. With an accuracy of approximately 0.9503, RMSProp demonstrates superior capability in making correct predictions. Accuracy is a key metric in tasks requiring consistent and reliable performance, making RMSProp the ideal choice for most applications. RAdam, on the other hand, excelled in the ROC AUC metric, achieving a near-perfect score of 1.0. This suggests that RAdam is particularly effective in distinguishing between positive and negative classes, making it highly suited for classification tasks that demand clear class separation. NovoGrad and LAMB also showed reasonable performance in terms of the F1 Score, ranging between 0.5 and 0.6. While they demonstrated balanced recall and precision, their performance in other metrics, such as accuracy, did not surpass RMSProp or RAdam. Based on the findings, the recommendation is to use RMSProp as the optimized algorithm for tasks requiring reliable and consistent performance across multiple metrics. RMSProp proves to be highly flexible and reliable for a wide range of machine learning applications. However, for tasks that prioritize class separation and require the highest performance in ROC AUC, RAdam presents a strong alternative.

Despite the promising results, this study has several limitations. First, while RMSProp achieved the highest accuracy (0.9503), accuracy alone does not fully capture model performance, especially in imbalanced datasets. The reliance on accuracy as a primary metric may overlook misclassification costs, particularly in critical agricultural applications. Second, although RAdam demonstrated a perfect ROC AUC score, suggesting strong class separation, real-world variability in cassava disease images may lead to performance degradation when applied to unseen data. Overfitting to the training dataset remains a potential issue, especially given the high ROC AUC score. Third, NovoGrad and LAMB, despite their balanced precision and recall, had lower accuracy, making them less suitable for general use. However, their F1 Scores (0.5–0.6) suggest potential in scenarios where recall and precision balance are more critical than accuracy alone. Additionally, the study does not explore hybrid optimization strategies, which could combine strengths from multiple optimizers. The absence of external validation on independent datasets limits the model's generalizability. Computational resource constraints may also impact scalability, as training deep learning models with different optimizers requires significant processing power. Future research should explore advanced techniques like meta-learning and optimizer adaptation for further improvements and combining of Deep Learning methods with Generative AI.

REFERENCES

Anitha, J., & Saranya, N. (2022). Cassava leaf disease identification and detection using deep learning approach. *International Journal of Computers, Communications & Control*, *17*(2). Advance online publication. DOI: 10.15837/ijccc.2022.2.4356

Applalanaidu, M. V., & Kumaravelan, G. (2021, February). A review of machine learning approaches in plant leaf disease detection and classification. In *2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV)* (pp. 716-724). IEEE. DOI: 10.1109/ICICV50876.2021.9388488

Aravind, S., & Harini, S. (2021). Cassava leaf disease classification using Deep Learning. *NVEO-NATURAL VOLATILES & ESSENTIAL OILS Journal| NVEO*, 9375-9389.

Bhugra, S., Srivastava, S., Kaushik, V., Mukherjee, P., & Lall, B. (2024). Plant Data Generation with Generative AI: An Application to Plant Phenotyping. *Applications of Generative AI*, 503-535.

Chamain, L. D., Qi, S., & Ding, Z. (2022). End-to-end image classification and compression with variational autoencoders. *IEEE Internet of Things Journal*, *9*(21), 21916–21931. DOI: 10.1109/JIOT.2022.3182313

Cheng, J., Tian, S., Yu, L., Gao, C., Kang, X., Ma, X., Wu, W., Liu, S., & Lu, H. (2022). ResGANet: Residual group attention network for medical image classification and segmentation. *Medical Image Analysis*, *76*, 102313. DOI: 10.1016/j.media.2021.102313 PMID: 34911012

Askr, H., El-dosuky, M., Darwish, A., & Hassanien, A. E. (2024). Explainable ResNet50 learning model based on copula entropy for cotton plant disease prediction. *Applied Soft Computing*, *164*, 112009.

Kingma, D. P., & Welling, M. (2019). An introduction to variational autoencoders. Foundations and Trends® in Machine Learning, 12(4), 307-392.

Legg, J. P., Kumar, P. L., Makeshkumar, T., Tripathi, L., Ferguson, M., Kanju, E., & Cuellar, W. (2015). Cassava virus diseases: Biology, epidemiology, and management. [Academic Press.]. *Advances in Virus Research*, *91*, 85–142. DOI: 10.1016/bs.aivir.2014.10.001 PMID: 25591878

Liu, B., Tan, C., Li, S., He, J., & Wang, H. (2020). A data augmentation method based on generative adversarial networks for grape leaf disease identification. *IEEE Access : Practical Innovations, Open Solutions*, *8*, 102188–102198. DOI: 10.1109/ACCESS.2020.2998839

Lokesh, G. H., Chandregowda, S. B., Vishwanath, J., Ravi, V., Ravi, P., & Al Mazroa, A. (2024). Intelligent Plant Leaf Disease Detection Using Generative Adversarial Networks: A Case-study of Cassava Leaves. *The Open Agriculture Journal*, *18*(1), e18743315288623. DOI: 10.2174/0118743315288623240223072349

Lumoring, N., Zakkiyah, A. Y., Kurniadi, F. I., & Minor, K. A. (2024, August). Classification of Tomato Maturity Using Transfer Learning EfficientnetB0 Algorithm. In 2024 8th International Conference on Information Technology, Information Systems and Electrical Engineering (ICITISEE) (pp. 47-52). IEEE. DOI: 10.1109/ICITISEE63424.2024.10730095

Mathulaprangsan, S., & Lanthong, K. (2021, December). Cassava leaf disease recognition using convolutional neural networks. In 2021 9th International Conference on Orange Technology (ICOT) (pp. 1-5). IEEE. DOI: 10.1109/ICOT54518.2021.9680655

McCallum, E. J., Anjanappa, R. B., & Gruissem, W. (2017). Tackling agriculturally relevant diseases in the staple crop cassava (Manihot esculenta). *Current Opinion in Plant Biology*, *38*, 50–58.

Kelleher, J. D. (2019). *Deep learning*. MIT press.

Moupojou, E., Tagne, A., Retraint, F., Tadonkemwa, A., Wilfried, D., Tapamo, H., & Nkenlifack, M. (2023). FieldPlant: A dataset of field plant images for plant disease detection and classification with deep learning. *IEEE Access : Practical Innovations, Open Solutions*, *11*, 35398–35410. DOI: 10.1109/ACCESS.2023.3263042

Paiva-Peredo, E. (2022, December). Deep learning for the classification of cassava leaf diseases in unbalanced field data set. In *International Conference on Advanced Network Technologies and Intelligent Computing* (pp. 101-114). Cham: Springer Nature Switzerland

Parmar, A., Sturm, B., & Hensel, O. (2017). Crops that feed the world: Production and improvement of cassava for food, feed, and industrial uses. *Food Security*, *9*(5), 907–927. DOI: 10.1007/s12571-017-0717-8

Sapre, M., Jatti, V. S., Tiwari, P., Kodachakki, N., & Undale, A. (2023). Disease Classification in Cassava Plant by Artificial Neural Network. In *Machine Learning and Optimization for Engineering Design* (pp. 75–84). Springer Nature Singapore. DOI: 10.1007/978-981-99-7456-6_6

Satoto, B. D., Syarief, M., & Khotimah, B. K. (2021, December). Region Proposal Convolutional Neural Network with augmentation to identifying Cassava leaf disease. In *2021 4th International Seminar on Research of Information Technology and Intelligent Systems (ISRITI)* (pp. 74-79). IEEE. DOI: 10.1109/ISRITI54043.2021.9702829

Singh, A. K., Rao, A., Chattopadhyay, P., Maurya, R., & Singh, L. (2024). Effective plant disease diagnosis using Vision Transformer trained with leafy-generative adversarial network-generated images. *Expert Systems with Applications*, *254*, 124387. DOI: 10.1016/j.eswa.2024.124387

Soujanya, A., Cherukuvada, S., Murugan, V., Ramya, S., Agalya, K., & Gayathri, S. (2023, December). An Efficient Classification of Cassava Leaf Disease using VGG 16. In 2023 International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems (ICSES) (pp. 1-5). IEEE. Askr DOI: 10.1109/ICSES60034.2023.10465476

Thaiyalnayaki, K., Raghul, S., & Ramachandran, S. (2022, April). Automatic classification of cassava using data augmentation and CNN. In *AIP Conference Proceedings* (Vol. 2405, No. 1). AIP Publishing. DOI: 10.1063/5.0072729

Thirunavukkarasu, J., Oindrilla, K. J., Sangeetha, M., & Swetha, E. (2022, July). An Efficient Deep Learning Approach for Plant Disease Detection. In *2022 International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems (ICSES)* (pp. 1-7). IEEE. DOI: 10.1109/ICSES55317.2022.9914063