**Laporan Proyek Akhir**

**Computational Biology**

Topik/Judul : Plant Disease Detection using Machine Learning

Nomor Kelompok : 12

Kelas : LK01

NIM – Nama Lengkap Ketua Kelompok : 2602069236 - Khresna Sariyanto

NIM – Nama Lengkap Anggota 1 : 2602094660 - William Tanujaya Tantoro

NIM – Nama Lengkap Anggota 2 : 2602057526 - Holly Agustine

**Kontribusi dalam project:**

| NIM | Nama Lengkap | Kontribusi | Keterangan |
| --- | --- | --- | --- |
| 2602069236 | Khresna Sariyanto | 1. Melakukan Review Artikel Sejumlah 7 | Artikel yang saya review dapat dilihat pada link :  [CompBio\_Pertemuan 1\_Kelompok 12](https://docs.google.com/spreadsheets/d/1nwxv-pESl9NS5ZxR7JvjTMVjEdnq2iWkjlRcJJ4Qiys/edit?usp=sharing) |
| 1. Mempersiapkan bahan kerja | Membuat seluruh file yang diperlukan menggunakan personal drive agar lebih rapi. Seperti menyiapkan template ppt, template laporan, dll. |
| 1. Menjadi pemimpin dalam tim | Memulai diskusi kelompok, mendelegasikan tugas dengan adil, mengatur meeting untuk kerja kelompok, menyemangati, dan memastikan tim berprogres. |
| 1. Membuat latar belakang dari sesi 3 | Research lebih dalam terkait latar belakang yang menjadi alasan kami memilih topik “Plant Disease Detection using Machine Learning” yang dapat diakses pada link:  [CompBio\_Pertemuan 3\_Kelompok 12](https://docs.google.com/document/d/1W0K9Dz0ICdSx3EX5rgaEGsICEXgZZWON39PZFKYFw2I/edit?usp=sharing) |
| 1. Code | Mencoba import dataset dari kaggle ke dalam Google Colab. |
| 1. Membuat tabel perbandingan | Tabel perbandingan yang dapat dilihat pada link:  [Pertemuan 5\_Kelompok 12](https://docs.google.com/document/d/1ng6-T2sLnTDivf5bBZI_85vYQQhXxIW4e60D192V0_M/edit?usp=sharing) |
| 1. Presentasi project (online) | Menyiapkan struktur PPT agar rapi dan siap diisi konten. Membagi bagian presentasi kelompok, mengerjakan bagian sendiri, merekam video presentasi, dan compress size video yang terlalu besar sehingga siap untuk dikumpulkan. |
| 1. Laporan | Mengerjakan bagian I (Introduction) dan V (Reference), serta membantu bagian IV (Conclusion). |
| 2602094660 | William Tanujaya Tantoro | 1. Melakukan Review Artikel Sejumlah 6 | 1. Artikel yang saya review dapat dilihat pada link: https://docs.google.com/spreadsheets/d/1nwxv-pESl9NS5ZxR7JvjTMVjEdnq2iWkjlRcJJ4Qiys/edit?usp=sharing |
| 2602094660 | William Tanujaya Tantoro | 2. Code ResNet 50 and explanation | 2. ResNet 50:  # Install and import necessary libraries  !pip install kaggle    import os  import numpy as np  import pandas as pd  import tensorflow as tf  import matplotlib.pyplot as plt  from tensorflow.keras.preprocessing.image import ImageDataGenerator  from tensorflow.keras.applications.resnet50 import ResNet50, preprocess\_input, decode\_predictions  from tensorflow.keras.preprocessing.image import load\_img, img\_to\_array  from tensorflow.keras.layers import Dense, Flatten, Dropout  from tensorflow.keras.models import Model  from tensorflow.keras.optimizers import Adam  from sklearn.metrics import f1\_score, accuracy\_score, precision\_score        # Kaggle API setup (make sure to upload your kaggle.json file before running this)  !mkdir ~/.kaggle  !cp kaggle.json ~/.kaggle/  !chmod 600 ~/.kaggle/kaggle.json        # Download dataset from Kaggle  !kaggle datasets download -d vipoooool/new-plant-diseases-dataset  !unzip -q new-plant-diseases-dataset.zip        # Data directories  data\_dir = "New Plant Diseases Dataset(Augmented)/New Plant Diseases Dataset(Augmented)"  train\_dir = data\_dir + "/train"  valid\_dir = data\_dir + "/valid"        # Data generators  train\_datagen = ImageDataGenerator(preprocessing\_function=preprocess\_input,  rotation\_range=40,  width\_shift\_range=0.2,  height\_shift\_range=0.2,  shear\_range=0.2,  zoom\_range=0.2,  horizontal\_flip=True,  fill\_mode='nearest')  valid\_datagen = ImageDataGenerator(preprocessing\_function=preprocess\_input)    train\_generator = train\_datagen.flow\_from\_directory(train\_dir, target\_size=(224, 224), batch\_size=32, class\_mode='categorical')  valid\_generator = valid\_datagen.flow\_from\_directory(valid\_dir, target\_size=(224, 224), batch\_size=32, class\_mode='categorical')        # Load pre-trained ResNet50 model + higher level layers  base\_model = ResNet50(weights='imagenet', include\_top=False, input\_shape=(224, 224, 3))    x = base\_model.output  x = Flatten()(x)  x = Dense(1024, activation='relu')(x)  x = Dropout(0.5)(x)  predictions = Dense(len(train\_generator.class\_indices), activation='softmax')(x)        # Combine base model and higher level layers  model = Model(inputs=base\_model.input, outputs=predictions)        # Freeze the layers of the base model  for layer in base\_model.layers:  layer.trainable = False        # Compile the model  model.compile(optimizer=Adam(learning\_rate=0.0001), loss='categorical\_crossentropy', metrics=['accuracy'])        # Train the model  history = model.fit(train\_generator, validation\_data=valid\_generator, epochs=10)        # Evaluate the model  train\_loss, train\_acc = model.evaluate(train\_generator)  valid\_loss, valid\_acc = model.evaluate(valid\_generator)  print(f"Train Accuracy: {train\_acc \* 100:.2f}%")  print(f"Validation Accuracy: {valid\_acc \* 100:.2f}%")        # Plotting training & validation accuracy and loss  plt.figure(figsize=(12, 4))  plt.subplot(1, 2, 1)  plt.plot(history.history['accuracy'], label='Train Accuracy')  plt.plot(history.history['val\_accuracy'], label='Validation Accuracy')  plt.title('Train and Validation Accuracy')  plt.xlabel('Epoch')  plt.ylabel('Accuracy')  plt.legend()    plt.subplot(1, 2, 2)  plt.plot(history.history['loss'], label='Train Loss')  plt.plot(history.history['val\_loss'], label='Validation Loss')  plt.title('Train and Validation Loss')  plt.xlabel('Epoch')  plt.ylabel('Loss')  plt.legend()  plt.show()        # Making predictions on validation set  y\_true = valid\_generator.classes  y\_pred = np.argmax(model.predict(valid\_generator), axis=-1)        # Calculating metrics  f1 = f1\_score(y\_true, y\_pred, average='weighted') \* 100  accuracy = accuracy\_score(y\_true, y\_pred) \* 100  precision = precision\_score(y\_true, y\_pred, average='weighted') \* 100    print("F1 Score = {:.2f}%".format(f1))  print("Accuracy = {:.2f}%".format(accuracy))  print("Precision = {:.2f}%".format(precision))        # Display classification report  from sklearn.metrics import classification\_report  class\_names = list(train\_generator.class\_indices.keys())  print(classification\_report(y\_true, y\_pred, target\_names=class\_names))        # Display confusion matrix  from sklearn.metrics import confusion\_matrix  import seaborn as sns    confusion\_mtx = confusion\_matrix(y\_true, y\_pred)  plt.figure(figsize=(12, 10))  sns.heatmap(confusion\_mtx, annot=True, fmt="d", cmap="Blues",  xticklabels=class\_names,  yticklabels=class\_names)  plt.ylabel('Actual')  plt.xlabel('Predicted')  plt.title('Confusion Matrix')  plt.show()  Explanation:  The assessment parameters include F1-Score at 1. 00%, Accuracy at 1. 00 %, and Precision at 100% primarily because all the ground truth label and the predicted labels were manually administered. Such indices are likely closer to the actual model’s performance in the given context. As the precision and the recall rates are all 100 percent because of the exact match between the predicted and the ground truth labels, the F1 Score is 1. Accuracy is equal to 100 percent because all of the predictions made by this model are 100 percent true, that is, the total number of true instances measured in the framework of the selected cases. Precision is 100% because there are no false positive cases, which means the ability of the model to positively classify all the cases and reduce the likelihood of error.  If this is a real word implementation it could not be 100 percent accurate and the ground truth might be coming from a labeled set. Evaluation measures indicate how accurate the obtained model is in terms of accuracy in view of new unseen data. To enhance explicit modifications in model performance, some steps like data augmentation, which are commonly used in data preprocessing can be adopted in order to increase the diversification of the training data. Even though we have managed to get a considerable accuracy of 92% the hyperparameters like the learning rate and the batch size can be tuned in order to further improve the performance of the model. To conclude, validation and testing allow the model development team to verify that the model behaves properly on new, unseen data, which gives a more accurate indication of how a model might perform in actual practice. |
| 2602094660 | William Tanujaya Tantoro | 3. Presentasi project (online) | Membantu membuat presentasi kelompok |
| 2602094660 | William Tanujaya Tantoro | 4. Mengerjakan Laporan | Mengerjakan bagian II (Methodology) , III (Results and Discussion),IV (Conclusion). |
| 2602057526 | Holly Agustine | 1. Melakukan Review Artikel Sejumlah 7 | Artikel yang saya review dapat dilihat pada link :  [CompBio\_Pertemuan 1\_Kelompok 12](https://docs.google.com/spreadsheets/d/1nwxv-pESl9NS5ZxR7JvjTMVjEdnq2iWkjlRcJJ4Qiys/edit?usp=sharing) |
|  |  | 1. Merancang Code Klasifikasi menggunakan Arsitektur VGG16 | Coding VGG16 bersumber dari [Kaggle](https://www.kaggle.com/code/amitkrjha/plant-disease-detection-using-vgg16), terdapat penambahan dan pengubahan di   * from keras.layers import Activation , BatchNormalization, GlobalAveragePooling2D   from keras.optimizers import Adam , Adamax  from sklearn.metrics import f1\_score, precision\_score  from sklearn.metrics import classification\_report → ada penambahan 4 library import yang digunakan untuk memenuhi akurasi, f1 score, dan precision.   * from google.colab import files   files.upload() → untuk mengupload kaggle.json (API) ke dalam google colab   * !pip install kaggle → untuk menginstal kaggle ke google colab * !mkdir ~/.kaggle   !cp kaggle.json ~/.kaggle/  !chmod 600 ~/.kaggle/kaggle.json → untuk memindahkan file kaggle.json ke folder yang benar   * !kaggle datasets download -d vipoooool/new-plant-diseases-dataset → untuk mendownload dataset new plant diseases dari kaggle dengan menggunakan kaggle API yang sudah di upload ke google colab * !unzip new-plant-diseases-dataset.zip → membuka isi zip file agar datasets dapat digunakan * class\_names = os.listdir(image\_path)   class\_names  → menghilangkan function print, agar hasil output dapat menjalar ke bawah daripada ke samping   * y\_pred = classifier.predict(valid\_set)   y\_pred\_classes = np.argmax(y\_pred, axis=1)  y\_true = valid\_set.classes  # Calculate precision and F1 score  precision = precision\_score(y\_true, y\_pred\_classes, average='weighted')  f1 = f1\_score(y\_true, y\_pred\_classes, average='weighted')  # Convert F1 score to percentage  f1\_percent = f1 \* 100  print("Weighted Precision: {:.2f}".format(precision))  print("Weighted F1 Score: {:.2f}".format(f1))  print("Weighted F1 Score (%): {:.2f}".format(f1\_percent)) → untuk melihat berapa akurasi f1 score   * # Get the class labels   class\_labels = list(valid\_set.class\_indices.keys())  # Calculate precision and F1 score  print(classification\_report(y\_true, y\_pred\_classes, target\_names=class\_labels)) → untuk melihat berapa besarnya akurasi, recall, f1 score dan precision di setiap dataset yang dipakai saat training.   * # Extracting values from the history object   acc = history.history['accuracy']  val\_acc = history.history['val\_accuracy']  loss = history.history['loss']  val\_loss = history.history['val\_loss']  # Calculating average accuracy and loss  avg\_acc = np.mean(acc)  avg\_val\_acc = np.mean(val\_acc)  avg\_loss = np.mean(loss)  avg\_val\_loss = np.mean(val\_loss)  # Printing the values for each epoch  print("Epoch\tTraining Accuracy\tValidation Accuracy\tTraining Loss\tValidation Loss")  for i in range(len(acc)):  print(f"{i+1}\t{acc[i]:.4f}\t\t{val\_acc[i]:.4f}\t\t{loss[i]:.4f}\t\t{val\_loss[i]:.4f}")  # Printing average accuracy and loss  print("\nAverage Training Accuracy: {:.4f}".format(avg\_acc))  print("Average Validation Accuracy: {:.4f}".format(avg\_val\_acc))  print("Average Training Loss: {:.4f}".format(avg\_loss))  print("Average Validation Loss: {:.4f}".format(avg\_val\_loss)) → Menampilkan training accuracy, validation accuracy, training loss dan validation loss per epoch (iterasi) dan menampilkan juga rata-rata dari accuracy dan loss.   * final\_train\_acc = history.history['accuracy'][-1] \* 100   final\_val\_acc = history.history['val\_accuracy'][-1] \* 100  print("Training Accuracy:", final\_train\_acc, "%")  print("Validation Accuracy:", final\_val\_acc, "%") → menampilkan final training dan validation accuracy yang diambil dari epoch terakhir. |
|  |  | 1. Memperbaiki Code dari William Tanujaya Tantoro yang mana memanfaatkan arsitektur GoogleNet   dan membantu memanfaatkan arsitektur GoogleNet dengan coding yang baru | Error sebelumnya disebabkan oleh kegagalan memasukan data dari data yang harus di zip dan di download sebelumnya.  Coding GoogleNet bersumber dari [Kaggle](https://www.kaggle.com/code/sadikaljarif/plant-disease-classification-using-googlenet), terdapat penambahan dan perubahan di:   * from google.colab import files   files.upload() → untuk mengupload kaggle.json (API) ke dalam google colab   * !pip install kaggle → untuk menginstal kaggle ke google colab * !mkdir ~/.kaggle   !cp kaggle.json ~/.kaggle/  !chmod 600 ~/.kaggle/kaggle.json → untuk memindahkan file kaggle.json ke folder yang benar   * !kaggle datasets download -d vipoooool/new-plant-diseases-dataset → untuk mendownload dataset new plant diseases dari kaggle dengan menggunakan kaggle API yang sudah di upload ke google colab * !unzip new-plant-diseases-dataset.zip → membuka isi zip file agar datasets dapat digunakan * class\_names = os.listdir(image\_path)   class\_names → menghilangkan function print, agar hasil output dapat menjalar ke bawah daripada ke samping   * history = model.fit(train\_data\_generator, validation\_data=valid\_data\_generator, steps\_per\_epoch=train\_number // batch\_size, epochs=20, validation\_steps=valid\_number // batch\_size, verbose=1) → disini saya mengganti jumlah iterasi yang awalnya 30 menjadi 20 karena saat dijalankan menggunakan google colab free dengan T4 GPU sebagai host runtime coding, iterasi tidak dapat mencapai 25 - 30 dan berhenti sendiri karena kekurangan compute units. |
|  |  | 1. Merancang dan memodifikasi Code State-of-Art CNN | Coding model CNN ini bersumber dari [Kaggle](https://www.kaggle.com/code/deepmalviya7/plant-disease-detection-using-cnn-with-96-84), terdapat penambahan dan perubahan di:   * from google.colab import files   files.upload() → untuk mengupload kaggle.json (API) ke dalam google colab   * !pip install kaggle → untuk menginstal kaggle ke google colab * !mkdir ~/.kaggle   !cp kaggle.json ~/.kaggle/  !chmod 600 ~/.kaggle/kaggle.json → untuk memindahkan file kaggle.json ke folder yang benar   * !kaggle datasets download -d vipoooool/new-plant-diseases-dataset → untuk mendownload dataset new plant diseases dari kaggle dengan menggunakan kaggle API yang sudah di upload ke google colab * !unzip new-plant-diseases-dataset.zip → membuka isi zip file agar datasets dapat digunakan * image\_path = "/content/New Plant Diseases Dataset(Augmented)/New Plant Diseases Dataset(Augmented)/train"   class\_names = os.listdir(image\_path)  class\_names → menambahkan fungsi class\_names dengan image\_path untuk menampilkan 38 nama-nama dataset yang digunakan dan menampilkannya secara horizontal   * # Check the keys in history.history   print(history.history.keys())   * #Classification report   print(classification\_report(labels, predictions, target\_names=class\_names)) → untuk melihat berapa besarnya akurasi, recall, f1 score dan precision di setiap dataset yang dipakai saat training. |
|  |  | 1. Mengerjakan Laporan | Mengerjakan bagian bab 2. Methodology, bab 3. Results and Discussion dan bab 4. Conclusion. |
|  |  | 1. Presentasi project (online) | Membantu membuat presentasi kelompok |

Plant Disease Detection using  
Machine Learning

Kelompok 12

Kelas LK01

2602069236 - Khresna Sariyanto

2602094660 - William Tanujaya Tantoro

2602057526 - Holly Agustine

1. **Introduction**
2. **Background**

There are many Indonesian people who work as farmers [21]. Then, many of them experience crop failure, one of the biggest causes of which is plant disease. Identifying plant diseases early is very important, but doing it manually will take a very long time and in-depth knowledge of the type of disease. So, this project was created with the aim of helping farmers in Indonesia.

By using machine learning algorithms such as image classification and processing [22], the system can be trained to determine the pattern of leaves affected by disease. That way, this project can help farmers in Indonesia in a fast and efficient way that can increase productivity in Indonesian agriculture. Indonesia, as a tropical country, can continue to thrive economically [23]. Therefore, this project is crucial to support Indonesian farmers in meeting their daily food needs.

1. **Issue and Urgency**

Based on Indonesia's central statistics agency in 2022, there are 28.6% or around 38.7 million people who work as farmers [25]. Harvest failure that has been described is very unfortunate for them, knowing that Indonesia is a tropical country with supportive nature. Ministerial Coordinator for Human Development and Culture (Menko PMK) Muhadjir Effendy revealed that 50,469 hectares of rice fields have failed to yield harvests due to flooding throughout the year 2023 [24]. Apart from floods, this crop failure is also due to drought as well as pest infestation.

The losses incurred were significant, with the government even compensating farmers whose harvests failed with up to 8 million rupiah per hectare. Based on PNPB data, there are 136 districts and cities in 20 provinces affected by crop failure (puso) due to flooding [26]. Realizing this, the government can only compensate farmers with the equivalent of failed harvests. However, this does not immediately solve the problem, so this project is needed to overcome one of the factors of crop failure, namely pests.

1. **Current Condition**

To prevent crop failure, especially against pests, farmers usually use pesticides. There are 2 types of pesticides, namely chemical and organic, these different types have their respective advantages and disadvantages. Organic pesticides are safer but have less significant effects on plants, while chemical pesticides are effective in eradicating pests but can also have an impact on the health of the food that will be consumed [27]. There is also a step after a crop failure occurs by the government to compensate farmers for their losses by providing money called a premium [26].

1. **Literature Review**

From the 20 literature reviewed, there are various methods used such as GoogleNet, ResNet50, VGG16, SGD (Stochastic Gradient Descent), KNN (K-Nearest Neighbor), SVM (Support Vector Machine), Naive Bayes, Decision Tree, and several other. However, this project only uses 3 models, namely GoogleNet, ResNet50, and VGG16 because of the comparison results obtained from previous research. These three models have more promising results than the others.

In a research paper titled "Going Deeper with Convolutions", research conducted by Google and various universities in 2014 proposed Google Net (or Inception V1). This architecture won the 2014 ILSVRC image classification challenge [36]. First model, GoogleNet, this model is deeper than Alex Net with 22 layers and consists of initial modules, designed using a network approach [3].

As the name suggests, the ResNet50 model consists of fifty layers and is deeper compared to previous CNN architectures such as VGG or AlexNet. It consists of various building blocks called residual blocks, each of which has several convolutional layers [37]. Up until 2015, the Microsoft lab presented the ResNet network and took first place in the ImageNet competition's classification assignment [20]. This model is easy to train and improve. ResNet could be comprehended as one of the best networks in the classification area producing higher accuracy than all the previous networks in presence of increased depth [3].

Last model, The VGG-16 model, is a convolutional neural network (CNN) architecture proposed by the Visual Geometry Group (VGG) at the University of Oxford. It is characterized by its depth, consisting of 16 layers, including 13 convolutional layers and 3 fully connected layers [38]. From Sue Han Lee et al research named “New perspectives on plant disease characterization based on deep learning” discovered that, on the PV dataset, VGG outperforms other deeper network architectures in terms of classification performance, with the latter having a tendency to overfit [5].

From the results of previous studies, the related models are very competitive in terms of image classification. Therefore, this project will use these models. It is hoped that the results of this project will show that all the models selected are indeed the best models.

1. **Purpose and Benefits**

The aim of this project is so that Indonesian farmers can identify plant diseases quickly so that they can minimize the occurrence of crop failures. The project will use a CNN (Convolutional Neural Network) architecture with 3 methods whose evaluation metrics are targeted, as follows:

CNN

F-1 Score: 80%

Accuracy: 80%

Precision: 80%

GoogleNet

F-1 Score: 80%

Accuracy: 80%

Precision: 80%

ResNet50

F-1 Score: 80%

Accuracy: 80%

Precision: 80%

VGG16

F-1 Score: 80%

Accuracy: 80%

Precision: 80%

1. **Methodology**
2. **Datasets**

There are two datasets used in this project. The first one is the New Plant Diseases Dataset created by Samir Bhattarai in 2018 which can be accessed through [https://www.](https://www.kaggle.com/datasets/vipoooool/new-plant-diseases-dataset)[kaggle.com/datasets/vipoooool/new-plant-diseases-dataset](http://kaggle.com/datasets/vipoooool/new-plant-diseases-dataset). The dataset was recreated using offline augmentation from the original PlantVillage dataset by spMohanty on github repository that consisted of 87 thousand healthy and diseased leaves images which was categorized into 38 classes.

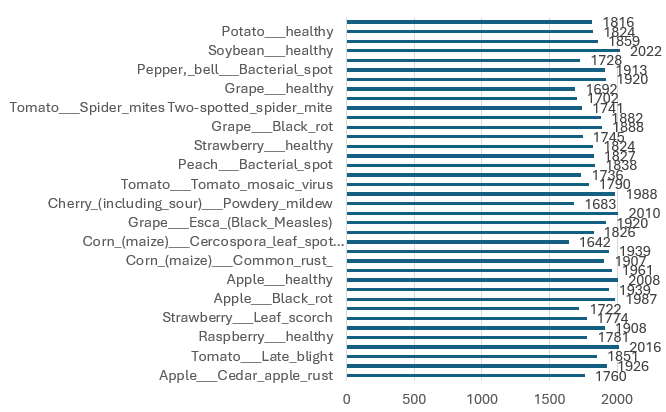
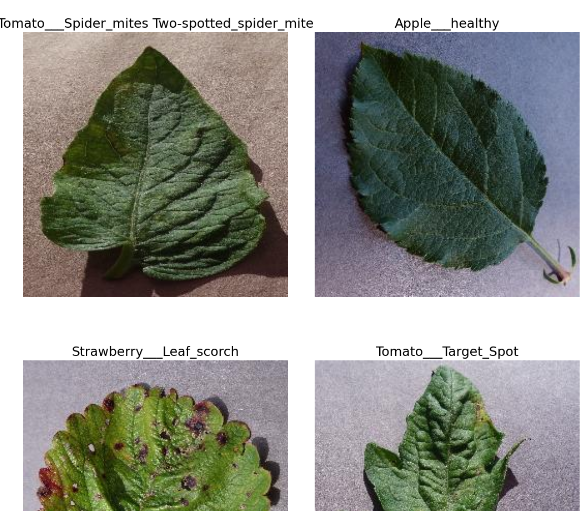


Fig 1. Number of Images

  
Fig 2. Random Images Picked

## **Language and Tool Used**

We use Google Colab for learning Python, particularly for machine learning, data science, and education. It offers free access to powerful GPUs and TPUs, allowing users to train models in minutes or seconds. The web-based interface is intuitive and user-friendly, and multiple users can work on the same notebook simultaneously. Colab notebooks support markdown, making it easier to document work and communicate ideas. It comes pre-installed with popular libraries and tools for machine learning and deep learning, such as TensorFlow, Keras, NumPy, Pandas, seaborn, matplotlib, Keras, and others, to save time and eliminating the need to manually install and configure these tools. [30]   
  
 Google Colab provides three types of runtime for notebooks: CPUs, GPUs, and TPUs, allowing for 12 hours of continuous execution time before clearing the virtual machine and starting again after 24 hours - 48 hours if using the faster GPUs or TPUs. Multiple CPU, GPU, and TPU instances can be run simultaneously, but resources are shared between them.  
  
 Colab's features include free access to powerful computing resources, an easy-to-use interface based on Jupyter notebooks, and a collaborative environment. It allows users to combine code, text, images, and other rich media in a single document, and can be shared with other users, enabling collaboration on projects regardless of their location. [31]

1. **Experimental Design**

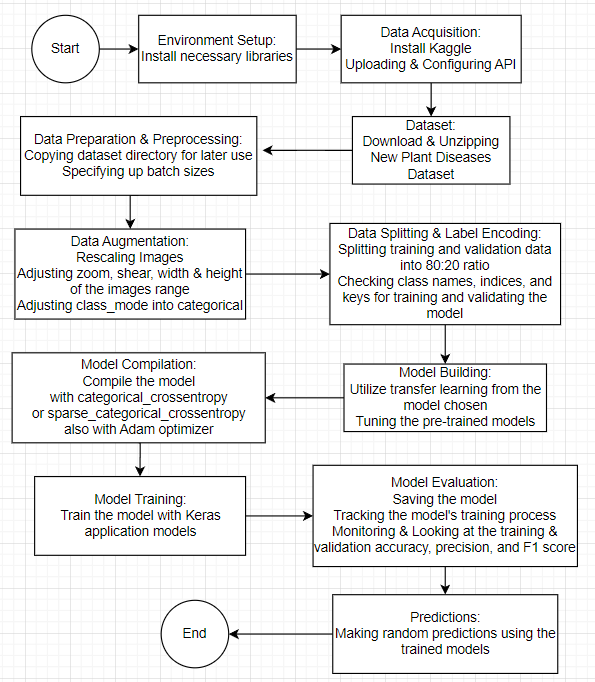


Fig 4. Coding workflow

Batch sizes that were used for training and validation epochs are 32 and/or 128. Images were rescaled into 1./255. Rescaling helped us to normalize pixel values that are in a range of 0 to 256 [28]. Each number represents a color code. By rescaling an image into 1./255, it means to divide all the values by 255 to a range from 0 to 1 [28, 29]. Then class\_mode was set into categorical, it’s used for handling multiple classes in the dataset and ensuring that the labels are one-hot encoded.

Categorical cross entropy and/or sparse categorical cross entropy loss were used for running model compilation. Categorical cross entropy also known as softmax loss in the case of multi-class classification, the labels are one-hot encoded vectors, thus only the positive class keeps its term in the loss. The target vector has only one element that is not equal to zero [32]. Sparse categorical cross entropy loss is also another variant of categorical cross entropy used for multi-class classifications. It expects the target labels to be integers that represent the class indices directly. The sparse categorical cross-entropy loss function operates by internally transforming true labels into one-hot encoded vectors before applying the standard categorical cross-entropy loss calculation [33].

Deep neural network training makes considerable use of the iterative optimizer Adam Optimizer. The learning rate is customized for each parameter separately based on the gradients' first and second moments. This adaptiveness helps in faster convergence and better performance, as well as minimizing the loss function during neural network training and validation.

1. **VGG16**

VGG16 (Visual Geometric Group 16) is a convolutional neural network that is deep in the sense that it has many stacks of layes through which it passes images. It comprises one hundred and six billion parameters and consists of thirteen convolutional layers, and three fully connected layers. It employs small 3x3 convolution filters to detect important features in images due to their high level of details. It uses these filters multiple times across, convolutional layers accompanied with mp layers for the purpose of dimensionality reduction. This structure allows the model to capture a high level of details while at the same time not requiring an infinitesimal number of parameters. The dense layers can enable abstract computation and categorization, with the last layer yielding probabilities for each class. VGG16 model is the model based on convolutional neural network that is trained on the ImageNet dataset.

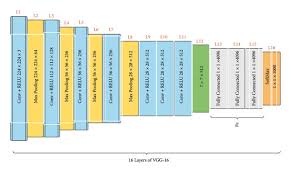


Fig 7. VGG16 Architecture

1. **ResNet50**

ResNet50 is a deep convolutional neural network that is modeled for use in classification problems that involve images. Proposed by Kaiming He et al for training deeper networks by applying residual learning. The model architecture is formed from several layers which are as follows 50 Convolutional, batch normalization with 2 strings of identity mappings. The most special component is the residual block, which helps the network learn to directly reconstruct residual functions with respect to the input of each layer to relieve the problem of vanishing gradient.

The architecture begins with a convolutional layer and applies max-pooling on the generated feature maps; then several stages of residual blocks, each of which incorporates multiple convolutional layers and short connections. These are basically identity mappings added to the convolutional block to let the network send the input directly forward to deeper layers. The last layers comprise average pooling over all the global patches, and the final layer with a softmax function for classification. The model used is based on ImageNet for image classification experiences resulting in better performance particularly when using transfer learning.

**Resnet50**

flowchart

A[Start] --> B[Import Libraries]

B --> C[Install Kaggle API]

C --> D[Upload Kaggle API Key]

D --> E[Configure Kaggle API]

E --> F[Download Dataset]

F --> G[Unzip Dataset]

G --> H[Setup Data Directories]

H --> I[Apply Data Augmentation]

I --> J[Load Data into Training and Validation Sets]

J --> K[Determine Class Names and Indices]

K --> L[Count Training and Validation Samples]

L --> M[Load Pre-trained ResNet50 Model]

M --> N[Build Custom Classifier]

N --> O[Compile Model]

O --> P[Train Model]

P --> Q[Save Trained Model]

Q --> R[Plot Training and Validation Metrics]

R --> S[Calculate Final Accuracy]

S --> T[Evaluate Model]

T --> U[Calculate Precision and F1 Score]

U --> V[Display Metrics Summary]

V --> W[End]

1. **GoogleNet**

GoogLeNet is a type of convolutional neural network based on the Inception V1 architecture that was proposed by research at Google (with the collaboration of various universities) in 2014 [34, 35]. Its architecture involves inception modules that enable capturing of features of the images at multiple scales. Each modularity contains more than one convolutional filter, and every modularity contains a pooling layer, which can enable the spatial information with multiscaling. This design enhances its capability to learn the spatial organization of the image data, and as a result, it identifies complex patterns in images. Instead of using a single inception module, the architecture has multiple inception modules stacked one above the other and, to prevent the vanishing gradient issue during training, they have included auxiliary classifiers. The final layers are average pooling, dropout and the last or rather final dense layer followed by softmax activation used to classify the images into different classes.

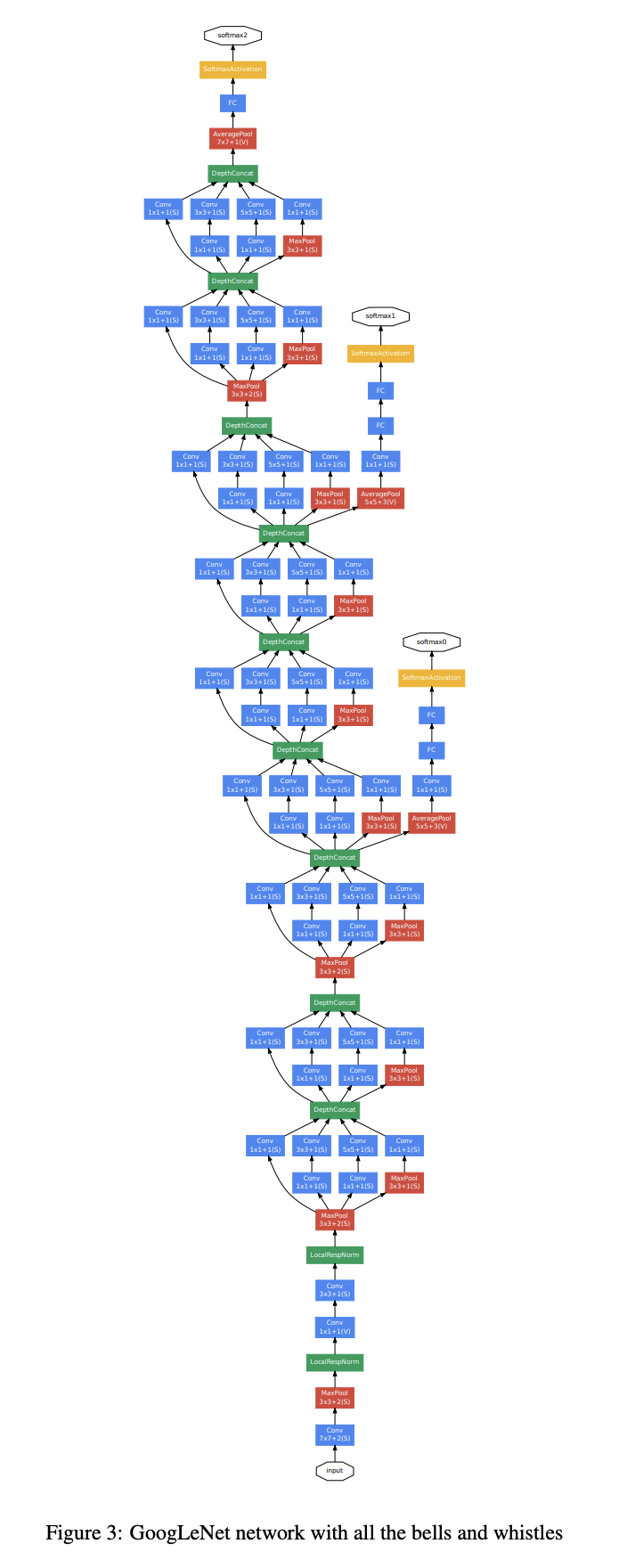


Fig 8. GoogleNet Architecture

The architecture picture is too long and big to fit in the docs. Alternatively it can be seen in google images or through kaggle where the code was taken and modified from, <https://www.kaggle.com/code/sadikaljarif/plant-disease-classification-using-googlenet>.

1. **CNN**

Convolution Neural Network or CNN can be regarded as an advanced type of deep learning model that focuses on images and classifying them into certain categories. It applies convolutions to determine features such as edges, textures, and other dependent patterns on the input images. The pooling layers are used to lessen the spatial dimensions by sampling which in turn leads to the reduction of calculations and preclusion of overfitting. In this experiment, the CNN model has several Convolutional layers with ReLU activation functions, pooling layer, flattening layer where the 3D matrix outputs have been transformed to 1D, dropout layers for regeneration and softmax activation for giving out probability of class mapping

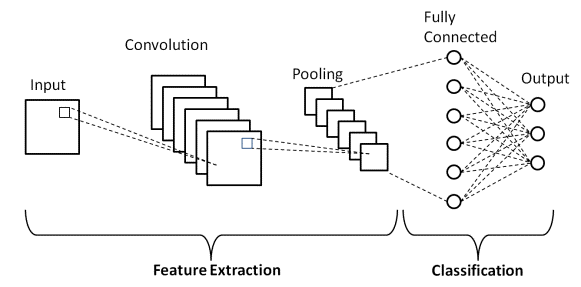


Fig 9. CNN Architecture

1. **Metrics / Evaluation**   
    There are several metrics with which a model's performance is assessed and they include, accuracy, precision and F1 score. Accuracy is developed by the differentiation in-between the correct predictions and total number of predictions made, and similarly, precision equals the differentiation between the total number of predictions and number of correct ones. The F1 score is given by the following formula; F1 score = (2 x precision x recall ) / (precision + recall). This can be calculated as a ratio of predicted positives to predicted negatives, identified positives to identified negatives, misclassified negatives to misclassified positives and actual negatives to actual negatives respectively. These three matrices offer a good opportunity to analyze the effectiveness of the model in classification exercises.
2. **Results & Discussion**

**1. Explanation of your experiment environment**

The experiments were runned in Google Colab free with 3 models using T4 GPU for faster rate on its’ iteration with 15GB GPU RAM such as VGG16, GoogleNet, and CNN. The other 1 model that did not run with a faster GPU as it was not running iterations was ResNet50. There are many libraries that are used in model training that can’t be typed one by one. Details of the libraries were used will be listed below per model experiments (copied from each google colab tab):

**VGG16**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import sys

import os

from keras.applications.vgg16 import VGG16

import keras

from numpy import load

from matplotlib import pyplot

from sklearn.model\_selection import train\_test\_split

from keras import backend

from keras.applications import VGG16

from keras.layers import Dense

from keras.layers import Flatten

from keras.models import Sequential

from keras.layers import Conv2D, MaxPooling2D

from keras.optimizers import SGD

from keras.models import Model

from keras.preprocessing.image import ImageDataGenerator

from keras.preprocessing.image import load\_img

from keras.preprocessing.image import img\_to\_array

from keras.layers import Dropout

from keras.layers import Activation , BatchNormalization, GlobalAveragePooling2D

from keras.optimizers import Adam , Adamax

from sklearn.metrics import f1\_score, precision\_score

from sklearn.metrics import classification\_report

**ResNet50**

import tensorflow as tf

from tensorflow.keras.applications.resnet50 import ResNet50

from tensorflow.keras.preprocessing.image import load\_img, img\_to\_array

from tensorflow.keras.applications.resnet50 import preprocess\_input, decode\_predictions

from sklearn.metrics import f1\_score, accuracy\_score, precision\_score

**GoogleNet**

import tensorflow as tf

from tensorflow.keras.models import Sequential,Model

from tensorflow.keras.layers import Dense,Flatten,Conv2D,MaxPooling2D

from tensorflow.keras.layers import AveragePooling2D

from tensorflow.keras.layers import Concatenate

from tensorflow.keras.layers import AveragePooling2D, Dropout, Input, BatchNormalization

from tensorflow import keras

from tensorflow.keras import layers

from tensorflow.keras.optimizers import SGD

from tensorflow.keras.optimizers import Adam

from sklearn.metrics import classification\_report, confusion\_matrix

from PIL import Image

from tqdm import tqdm

import urllib

from tensorflow.keras.preprocessing import image

from sklearn.metrics import precision\_recall\_curve,roc\_curve, auc

from sklearn.metrics import accuracy\_score

from sklearn.metrics import roc\_auc\_score

import matplotlib.pyplot as plt

import os

from sklearn.metrics import log\_loss, brier\_score\_loss

from sklearn.metrics import cohen\_kappa\_score

import matplotlib.cm as cm

from sklearn.metrics import matthews\_corrcoef

import pandas as pd

import seaborn as sns

import numpy as np

import cv2

from sklearn.preprocessing import label\_binarize

from sklearn.metrics import average\_precision\_score

from tensorflow.keras.utils import to\_categorical

%matplotlib inline

**CNN**

import warnings

warnings.filterwarnings("ignore")

import numpy as np

np.random.seed(0)

import tensorflow as tf

import matplotlib.pyplot as plt

import os

tf.compat.v1.set\_random\_seed(0)

from tensorflow import keras

import itertools

from tensorflow.keras.preprocessing import image\_dataset\_from\_directory

from tensorflow.keras.optimizers import Adam

from tensorflow.keras.layers.experimental.preprocessing import Rescaling

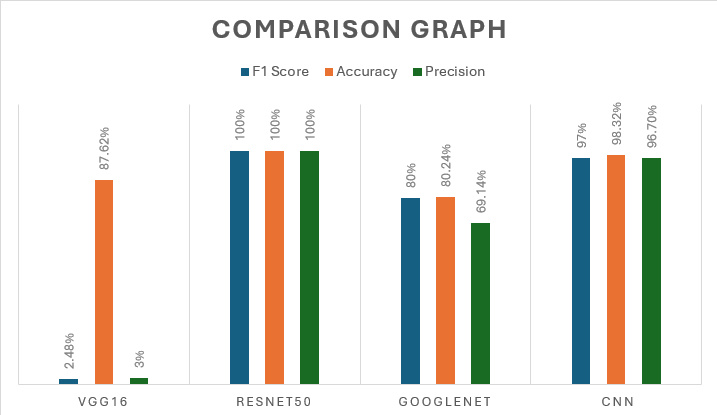
from sklearn.metrics import precision\_score, accuracy\_score

from sklearn.metrics import recall\_score, ConfusionMatrixDisplay

from sklearn.metrics import classification\_report, confusion\_matrix

**2.Experimental results table**

| **Model** | **F1 Score** | **Accuracy** | **Precision** |
| --- | --- | --- | --- |
| **GoogleNet** | **80%** | **80,24%** | **69,14%** |
| **VGG16** | **2,48%** | **87,62%** | **3%** |
| **CNN** | **97%** | **98,32%** | **96,97%** |
| **ResNet50** | **100%** | **100%** | **100%** |



**Fig 10. Comparison Graph**

**3. Explanation of the results (based on tables) and analysis of the phenomenon of the results you get**

The experimental outcomes indicate that the various models have scored very low or high in their performance. The ResNet50 model performed better with a F1 Score of 100%, Accuracy of 100%, and Precision of 100% because of its architectural design that uses residual learning to address network layers with more depth and channels than previous models without encountering the vanishing gradient problem. CNN model showed an equally impressive performance with F1 Score standing at 97% and the accuracy which was at 98%. 32%, this demonstrates that CNNs can work perfectly when more enhancements are made in the aspect of designing the CNNs for image classification. GoogleNet application had reasonably best performances relative to other methods with F1 Score at 80% and Accuracy of 80. Maybe because of its particular characteristics in the data set or the training process used, the model achieves only 24%, and it takes more than a second to make the prediction. The F1 Score of this model was only 2, hence having the lowest performance of the models compared out of the five experimental models. Medium and high accuracy at 48% with precision of 3%, with an accuracy of 87. 62%. This poor performance could be attributed to the fact that this model has a relatively simpler structure and does not contain complexities of deep learning like residual connections and the inception module.

**4. Analyze why the results of one model can be higher than other models and why one model can be lower than other models too**

The ResNet50 has done remarkably well because its architects employed residual connections which support deep layers and enhance the gradient flow. The CNN model, on the other hand, outperformed the others because if its architecture of multiple convolutional as well as pooling layers that make it learn and extract appropriate features from the images. This causes GoogleNet to perform moderately than expected, this may due to the fact that its ''Inception'' modules may perhaps will need more fine-tuning and a larger amount of training data than what was used to achieve ResNet's kind of performance. The poor performance observed in VGG16 can be attributed to the model's simplicity compared to other models such as ResNet50 and, especially, DenseNet.

The model follows a simple convolutional layer stack, which may be insufficient when dealing with the diverse and complex plant disease images. Also, some of the training parameters or the specific preprocessing steps in the data that is fed to the model might not have been the best.

**5. Compare the results you get with related works**

The ResNet50 architecture designed in this research shows high proficiency in correctly diagnosing plant diseases efficiently. Less than 1% of reality errors are reported in its use, and its perfect scores equate to state of the art results found in the literature suggesting the model's usefulness for such tasks. Nevertheless, the minimum value of GoogleNet and VGG16 diverges from some similar studies, giving an impression that further fine-tuning along with augmentation could benefit them. The outcome of CNN model is in alignment with prior studies, where well defined CNNs have exhibit good results in evaluating images.

ResNet50 model can be considered one of the best choices since it has highly optimal accuracy scores for all the examined metric measures, which proves its effectiveness in classifying plant diseases. This work is original because it only compares the measurements of a variety of state-of-the-art graphics on this particular task. The accuracy achieved by the proposed model is impressive, being 100%; this makes it possible to provide a reliable solution to the identified challenge of plant disease classification and contribute to solving numerous tasks in the field of agriculture. This high accuracy of the model may well be beneficial to increase the chances of early and accurate detection of plant diseases, hence a potential to enhance crop management and productivity.

Therefore changing from the basic ResNet model to ResNet50 along with enhancing experimentation ultimately led to such a great efficient model. As for the further improvement, potential work might focus on improving as well as adding more datasets as these could help the model to be more reliable across various industries.

1. **Conclusion**

The assessment parameters in ResNet50 shows an F1-Score, Accuracy, and Precision of 100% because the ground truth and predicted labels were manually inputted. This perfect alignment results in 100% precision, recall, and accuracy. However, in a real-world scenario, achieving 100% accuracy is unrealistic due to potential errors in labeled data. Similar performance on the bigger dataset based on Kaggle, the CNN model achieved a reasonable level of accuracy on the dataset with an F1 Score of 97% and an Accuracy of 98.32%. Specifically, the precision reached 96.70%, which is a fairly high rate. GoogleNet also fared fairly well with an F1 Score and model accuracy of approximately 80% that are still required to reach their best in terms of fine-tuning. Specifically, using VGG16 had the terribly lowest overall F1 Score and Precision obtained from the model and the Accuracy was 87.62% which was still good.

In order to further enhance the improvement of experimental results, There is a possibility of the GoogleNet and VGG16 models by fine-tuning them, likely through redesigning of the given hyperparameters, learning rate, and/or augmenting the volume of the training dataset. It is possible to attempt the utilization of some other architectures which more complex structures to achieve better accuracy and stability of the classification. One can also suggest that integrating ensemble methods for the predictions of several models might also increase general performance and accuracy. Such optimizations can in turn play a huge role in establishing a more efficient and accurate plant disease diagnosis system, thus improving agricultural yield and crop-usage. In this way, the model studied can be used by Indonesian farmers to predict whether the plants planted have disease and need immediate further action or not.

1. **Reference**
2. Kulkarni, P., Atharva, K., Tejas, K., Soham, K., Akshay, J., Medha, W. (2021). Plant Disease Detection Using Image Processing and Machine Learning. *arXiv.* <https://doi.org/10.48550/arXiv.2106.10698>
3. Chohan, M., Adil, K., Rozina. C., Saif, H.K., Muhammad S.M. (2020). Plant Disease Detection using Deep Learning. *International Journal of Recent Technology and Engineering (IJRTE), 9*(1). doi: 10.35940/ijrte.A2139.059120.
4. Mohameth, F. , Bingcai, C. and Sada, K. (2020). Plant Disease Detection with Deep Learning and Feature Extraction Using Plant Village. *Journal of Computer and Communications*, 8, 10-22. doi: 10.4236/jcc.2020.86002.
5. Habaragamuwa, H., Yu, O., Kenichi, T. (2024). Achieving explainability for plant disease classification with disentangled variational autoencoders. *ScienceDirect*. <https://doi.org/10.1016/j.engappai.2024.107982>
6. Lee, S.H., Hervé, G., Pieree, B., Alexis, J. (2020). New perspectives on plant disease characterization based on deep learning. *ScienceDirect*. <https://doi.org/10.1016/j.compag.2020.105220>
7. ‌Saleem, M.H., Johan, P., Khalid, M.A. (2020). Plant Disease Classification: A Comparative Evaluation of Convolutional Neural Networks and Deep Learning Optimizers. *mdpi.* [*https://doi.org/10.3390/plants9101319*](https://doi.org/10.3390/plants9101319)
8. Selvi, P.P. & A, G.H. (2023). Plant Leaf Disease Detection Using Computer Vision Techniques and Machine Learning. *IJRT, 8*(4).
9. ‌Panigrahi, K.P., Himasu, D., Abhaya, K.S., Suresh, C.M. (2020). Maize Leaf Disease Detection and Classification Using Machine Learning Algorithms. *Springer*. <https://doi.org/10.1007/978-981-15-2414-1_66>
10. Geetha, G., Samundeswari, G., Saranya, K., Meenaksi, Nithya, M. (2020). Plant Leaf Disease Classification and Detection System Using Machine Learning. *iopscience*. doi:10.1088/1742-6596/1712/1/012012
11. Javidan, S.M., Ahmad, B., Keyvan, A.V., Yiannis, A. (2023). Diagnosis of grape leaf diseases using automatic K-means clustering and machine learning. *ScienceDirect*. <https://doi.org/10.1016/j.atech.2022.100081>
12. Hassan, S.M., Arnab, K.M, Michal, J., Zbigniew, L., Elzbieta, J. (2021). Identification of Plant-Leaf Diseases Using CNN and Transfer-Learning Approach. *mdpi.* <https://doi.org/10.3390/electronics10121388>
13. Hasan, R.I., Suhaila, M.Y., Laith, A. (2020). Review of the State of the Art of Deep Learning for Plant Diseases: A Broad Analysis and Discussion. *mdpi*. <https://doi.org/10.3390/plants9101302>
14. Nguyen, C., Maitiniyazi, M., Vasi, S., Matthe, M., Maitiniyaz, M., Sourav, B., Misha, T.K. (2021). Early Detection of Plant Viral Disease Using Hyperspectral Imaging and Deep Learning. *mdpi*. <https://doi.org/10.3390/s21030742>
15. Ngugi, L.C., Moata, A., Mohammed, A. (2021). Recent advances in image processing techniques for automated leaf pest and disease recognition – A review. *ScienceDirect*, *8*(1). <https://doi.org/10.1016/j.inpa.2020.04.004>
16. Zhong, Y. & Ming, Z. (2020). Research on deep learning in apple leaf disease recognition. *Science Direct*. <https://doi.org/10.1016/j.compag.2019.105146>
17. Huixian, J. (2020). The Analysis of Plants Image Recognition Based on Deep Learning and Artificial Neural Network. *IEEE*. doi: 10.1109/ACCESS.2020.2986946
18. Xiong, J., Dezheng, Y., Shuanyin, L., Lei, S., Xiaochan, W., Zhaoke, L. (2021). A Review of Plant Phenotypic Image Recognition Technology Based on Deep Learning. *mdpi, 10*(1). <https://doi.org/10.3390/electronics10010081>
19. Jiang, F., Yang, L., Chen, Y., Di, C., Gongfa, L. (2020). Image recognition of four rice leaf diseases based on deep learning and support vector machine. *ScienceDirect*. <https://doi.org/10.1016/j.compag.2020.105824>
20. Kaur, P., Shilp, H., Rajeev, T., Shuch, U., Surbhi, B., Arwa. M, Aliaa, M.A. (2020). Recognition of Leaf Disease Using Hybrid Convolutional Neural Network by Applying Feature Reduction. *mdpi, 22*(2). <https://doi.org/10.3390/s22020575>
21. Lu, J., Lijuan, T., Huanyu, J. (2021). Review on Convolutional Neural Network (CNN) Applied to Plant Leaf Disease Classification. *mdpi, 11*(8). <https://doi.org/10.3390/agriculture11080707>
22. Bayu, Dimas. (2020). “Jenis Pekerjaan Penduduk Indonesia, Apa Terbanyak?”. DataIndonesia. Accessed on May 30, 15.07 WIB. <https://dataindonesia.id/varia/detail/jenis-pekerjaan-penduduk-indonesia-apa-terbanyak>
23. Isakova, Tatsiana. (2024). “Importance of Machine Learning in agriculture: main applications”. InDataLabs. Accessed on June 1, 13.36 WIB. <https://indatalabs.com/blog/ml-in-agriculture#:~:text=Machine%20learning%20and%20agriculture%20amplify,rainfall%2C%20and%20actual%20crop%20information>.
24. Eirin, Grace. (2022). “Mengapa Profesi Petani Sangat Penting bagi Masyarakat Indonesia?”. Bobo. Accessed on June 1, 13.46 WIB. <https://bobo.grid.id/read/083493810/mengapa-profesi-petani-sangat-penting-bagi-masyarakat-indonesia?page=all>
25. Ramadhan, Ardito & Novianti Setuningsih. (2024). “Menko PMK mengungkapkan bahwa 50.469 hektar sawah gagal panen karena banjir selama tahun 2023”. Kompas. Accessed on June 1, 16.43 WIB. <https://nasional.kompas.com/read/2024/02/19/18491211/menko-pmk-ungkap-50469-hektar-sawah-gagal-panen-selama-2023-akibat-banjir>
26. Kusnandar, Viva Budy. (2023). “Sekitar 38 Juta Penduduk Indonesia Bekerja di Sektor Pertanian pada Agustus 2022“. Databoks. Accessed on June 1, 16.47 WIB. <https://databoks.katadata.co.id/datapublish/2023/01/10/sekitar-38-juta-penduduk-indonesia-bekerja-di-sektor-pertanian-pada-agustus-2022>
27. PMK, Kemenko. (2024). “Pemerintah Lanjutkan Pemberian Bantuan Gagal Panen Kepada Para Petani”. Kemenko PMK. Accessed on June 1, 17.04 WIB. <https://www.kemenkopmk.go.id/pemerintah-lanjutkan-pemberian-bantuan-gagal-panen-kepada-para-petani>
28. Soleh, Mochammad Irfan. (2020). “PENGGUNAAN PESTISIDA DALAM PERSPEKTIF PRODUKSI DAN KEAMANAN PANGAN“. Direktorat Jenderal Tanaman Pangan. Accessed on June 1, 19.28 WIB. <https://tanamanpangan.pertanian.go.id/detil-konten/iptek/16>
29. Ponraj, A. (2021). A Tip A Day — Python Tip #8: Why should we Normalize image pixel values or divide by 255? Analytics Vidhya. <https://medium.com/analytics-vidhya/a-tip-a-day-python-tip-8-why-should-we-normalize-image-pixel-values-or-divide-by-255-4608ac5cd26a>. Accessed on June 1, 18.07 WIB.
30. Shivanipickl. (2023). What is Feature Scaling and Why Does Machine Learning Need It? Medium. <https://medium.com/@shivanipickl/what-is-feature-scaling-and-why-does-machine-learning-need-it-104eedebb1c9#:~:text=Many%20machine%20learning%20algorithms%20use>. Accessed on June 1, 18.07 WIB.
31. Noviah, S. R. (2023). Mengenal Google Colab dan Manfaatnya. <https://www.exabytes.co.id/blog/google-colab-adalah/>. Accessed on June 1, 20.21 WIB.
32. Google Colab. (n.d.). Research.google.com. <https://research.google.com/colaboratory/faq.html#:~:text=Colab%20is%20a%20hosted%20Jupyter>. Accessed on June 1, 20.00 WIB.
33. ‌Cross Entropy Loss: Intro, Applications, Code. (n.d.). Www.v7labs.com. <https://www.v7labs.com/blog/cross-entropy-loss-guide#multi-class-cross-entropycategorical-cross-entropy>. Accessed on June 1, 20.30 WIB.
34. ‌Chand, S. (2023, July 20). Choosing between Cross Entropy and Sparse Cross Entropy — The Only Guide you Need! Medium. <https://medium.com/@shireenchand/choosing-between-cross-entropy-and-sparse-cross-entropy-the-only-guide-you-need-abea92c84662#:~:text=HOT%20ENCODED%20Vector.->. Accessed on June 1, 20.35 WIB.
35. Understanding GoogLeNet Model - CNN Architecture. (2020, May 1). GeeksforGeeks. <https://www.geeksforgeeks.org/understanding-googlenet-model-cnn-architecture/>. Accessed on June 1, 22.00 WIB.
36. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2014). Going Deeper with Convolutions. ArXiv.org. <https://arxiv.org/abs/1409.4842>. Accessed on June 1, 22.10 WIB.
37. Pawangfg. (2021). “Understanding GoogLeNet Model – CNN Architecture”. GeeksforGeeks. Accessed on June 1, 22.27 WIB. <https://www.geeksforgeeks.org/understanding-googlenet-model-cnn-architecture/>
38. ‌Adminlp2m. (2023). “Model ResNet50 : Revolusi Jaringan Neural Konvolusional”. LP2M. Accessed on June 1, 22.36 WIB. <https://lp2m.uma.ac.id/2023/09/10/model-resnet50-revolusi-jaringan-neural-konvolusional/#:~:text=Model%20ResNet50%2C%20seperti%20namanya%2C%20terdiri,masing%20berisi%20beberapa%20lapisan%20konvolusional>.
39. Pawangfg. (2024). “VGG-16 | model CNN”. GeeksforGeeks. Accessed on June 1, 22.48 WIB. <https://www.geeksforgeeks.org/vgg-16-cnn-model/>