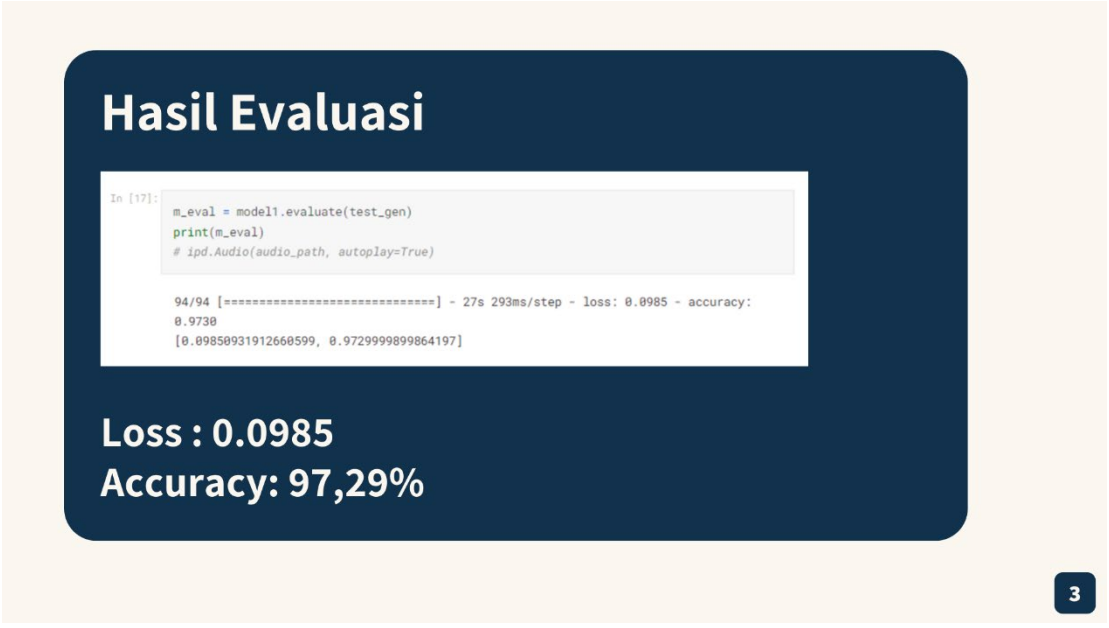
Vegetable Image Classification with CNN by Difa Farhani Hakim



Summary

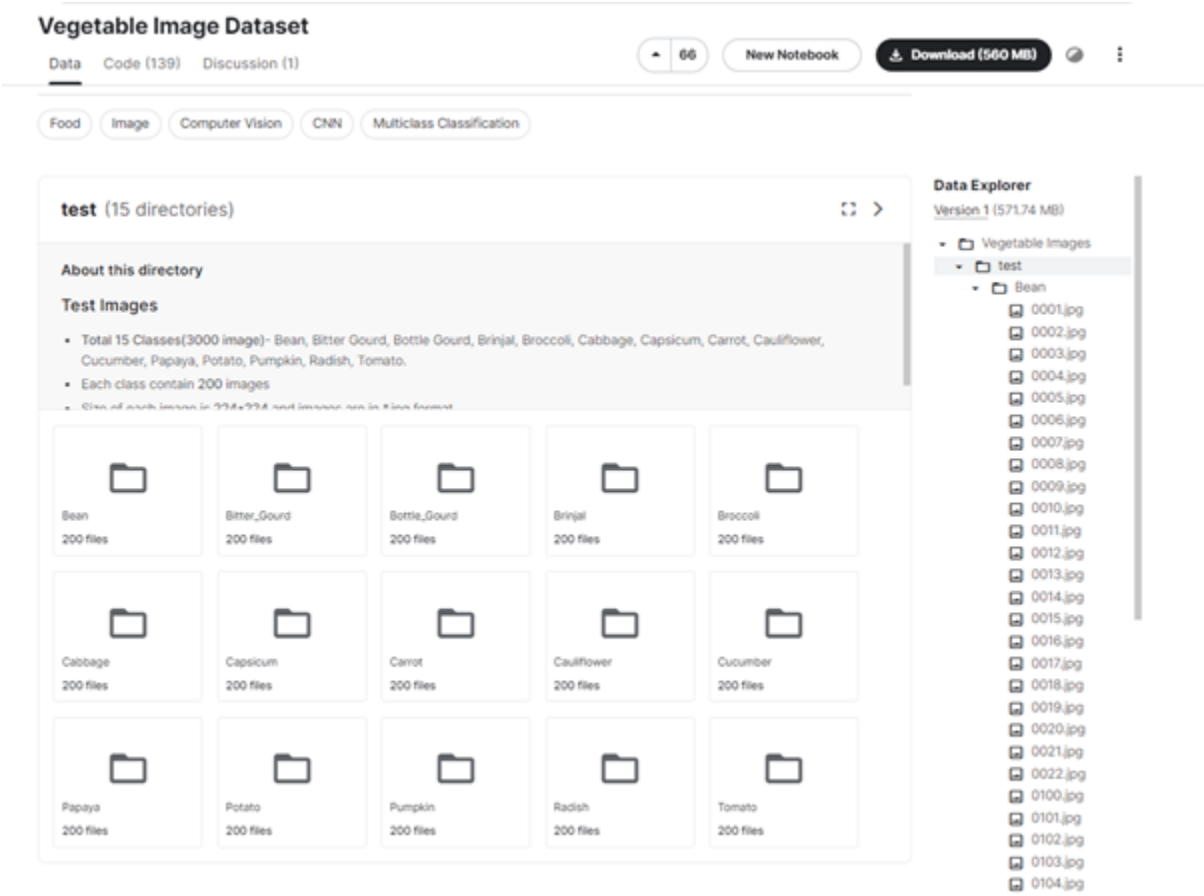
Model klasifikasi sayuran menggunakan CNN berdasarkan sebuah dataset gambar yang diperoleh dari kaggle berhasil dilakukan. Dataset tersebut berisi 15 jenis sayuran dengan total gambar 21.000 yang berukuran 224x224 dan dalam format gambar *.jpg. Dataset tersebut terbagi menjadi 15.000 gambar untuk data training, 3000 gambar untuk validasi, dan 3000 gambar untuk testing. Hasil testing model yang diperoleh memiliki keakuratan sebesar 97,29%.

Description

Convolutional Neural Network (CNN) adalah salah satu jenis neural network yang dapat digunakan untuk mendeteksi dan mengenali objek pada sebuah image mengikuti cara kerja sistem penglihatan manusia.

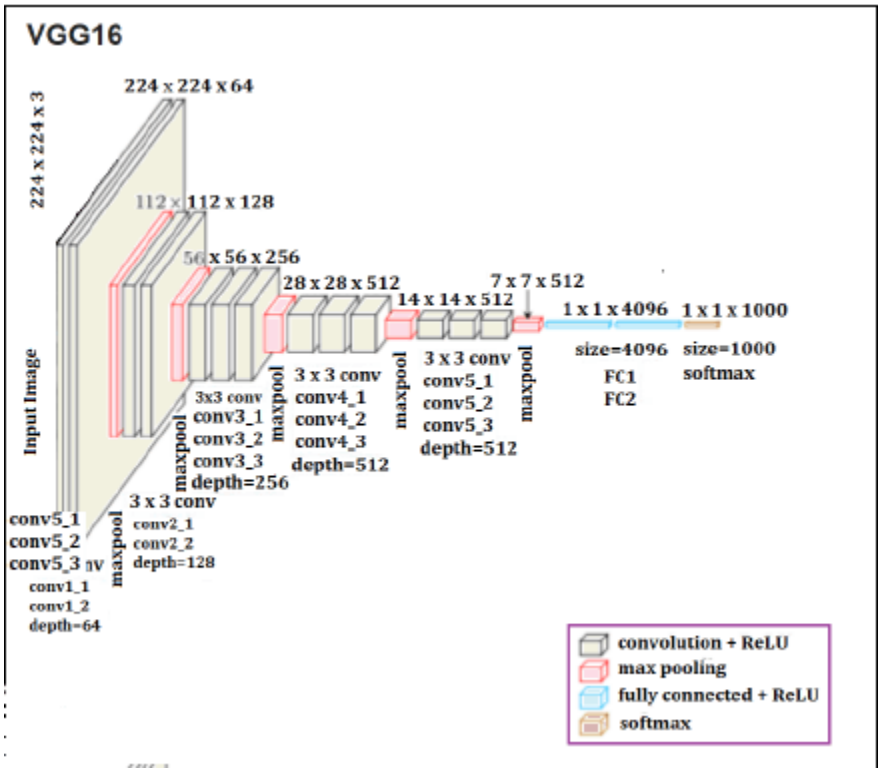
Data Set

Dataset yang digunakan berasal dari kaggle dengan username M Israk ahmed dengan judul “Vegetable Image Dataset”. Dataset tersebut memiliki 15 kelas sayuran, antara lain bean, bitter gourd, bottle gourd, brinjal, broccoli, cabbage, capsicum, carrot, cauliflower, cucumber, papaya, potato, pumpkin, radish and tomato. Total gambar yang ada di dalam dataset berjumlah 21.000 gambar dengan pembagian 1.400 gambar per kelas. Format gambar yang digunakan adalah *.jpg serta ukuran yang gambar adalah 224x224 px. Dalam proses training permodelan, dataset tersebut terbagi menjadi 3 folder yang berisi 15.000 gambar untuk training, 3.000 gambar untuk validasi, dan 3.000 gambar untuk testing.



Permodelan

Model yang dibuat terinspirasi dengan architecture model VGG-16 dan model yang ada pada buku Practical Machine Learning and Image Processing karya Himanshu Singh.



Import Library

In [2]:

```
# Common
import os
import keras
import numpy as np
import tensorflow as tf
import random

# Sound Load
import IPython.display as ipd

# Data
from keras.preprocessing.image import ImageDataGenerator

# Data Visualization
import plotly.express as px
import matplotlib.pyplot as plt

# Image Load
from PIL import Image

# Callbacaks
from keras.callbacks import EarlyStopping, ModelCheckpoint

# Model
from tensorflow import keras
from keras.models import Sequential, load_model
from keras.layers import Dense, Flatten, Conv2D, MaxPooling2D, Dropout
from tensorflow.keras import layers
```

Fungsi

Plot Acurracy dan Loss

In [3]:

```
def plot_accuracy_and_loss(train_model):
    hist = train_model.history
    acc = hist['accuracy']
    val_acc = hist['val_accuracy']
    loss = hist['loss']
    val_loss = hist['val_loss']
    epochs = range(len(acc))
    f, ax = plt.subplots(1,2, figsize=(14,6))
    ax[0].plot(epochs, acc, 'g', label='Training accuracy')
    ax[0].plot(epochs, val_acc, 'r', label='Validation accuracy')
    ax[0].set_title('Training and validation accuracy')
    ax[0].set(xlabel='Epoch', ylabel='Accuracy')
    ax[0].legend()
    ax[1].plot(epochs, loss, 'g', label='Training loss')
    ax[1].plot(epochs, val_loss, 'r', label='Validation loss')
    ax[1].set_title('Training and validation loss')
    ax[1].set(xlabel='Epoch', ylabel='Accuracy')
    ax[1].legend()
    plt.show()
```

Plot Gambar dan Tebakan

In [4]:

```
def show_images(GRID=[5,5], model=None, size=(20,20), Data=1):
    n_rows = GRID[0]
    n_cols = GRID[1]
    n_images = n_cols * n_rows

    i = 1
    plt.figure(figsize=size)
    for images, labels in Data:
        id = np.random.randint(32)
        image, label = images[id], class_names[int(labels[id])]

        plt.subplot(n_rows, n_cols, i)
        plt.imshow(image)

        if model is None:
            title = f"Class : {label}"
        else:
            pred = class_names[int(np.argmax(model.predict(image[np.newaxis, ...])))]
            title = f"Org : {label}, Pred : {pred}"

        plt.title(title)
        plt.axis('off')

        i+=1
        if i>=(n_images+1):
            break

    plt.tight_layout()
    plt.show()
```

Load Data

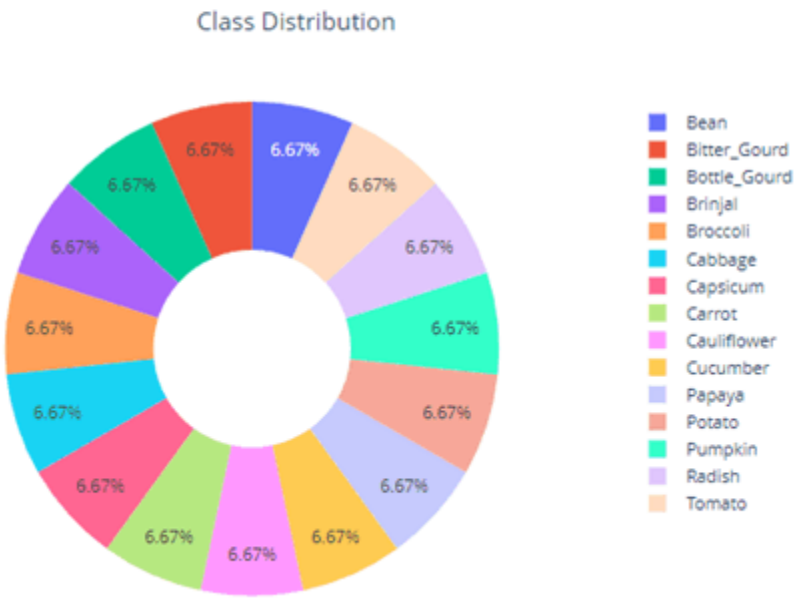
Detect class name

In [5]:
```
root_path = '../input/vegetable-image-dataset/Vegetable Images/train/'
class_names = sorted(os.listdir(root_path))
n_classes = len(class_names)

# Class Distribution
class_dis = [len(os.listdir(root_path + name)) for name in class_names]

# Show
print(f"Total Number of Classes : {n_classes} \nClass Names : {class_names}")
```

```
Total Number of Classes : 15
Class Names : ['Bean', 'Bitter_Gourd', 'Bottle_Gourd', 'Brinjal', 'Broccoli', 'Cabb
age', 'Capsicum', 'Carrot', 'Cauliflower', 'Cucumber', 'Papaya', 'Potato', 'Pumpki
n', 'Radish', 'Tomato']
```

Data Visualisasi

In [6]:
```
# Visualize
fig = px.pie(names=class_names, values=class_dis, title="Class Distribution", hole=0.
4)
fig.update_layout({'title':{'x':0.5}})
fig.show()
```



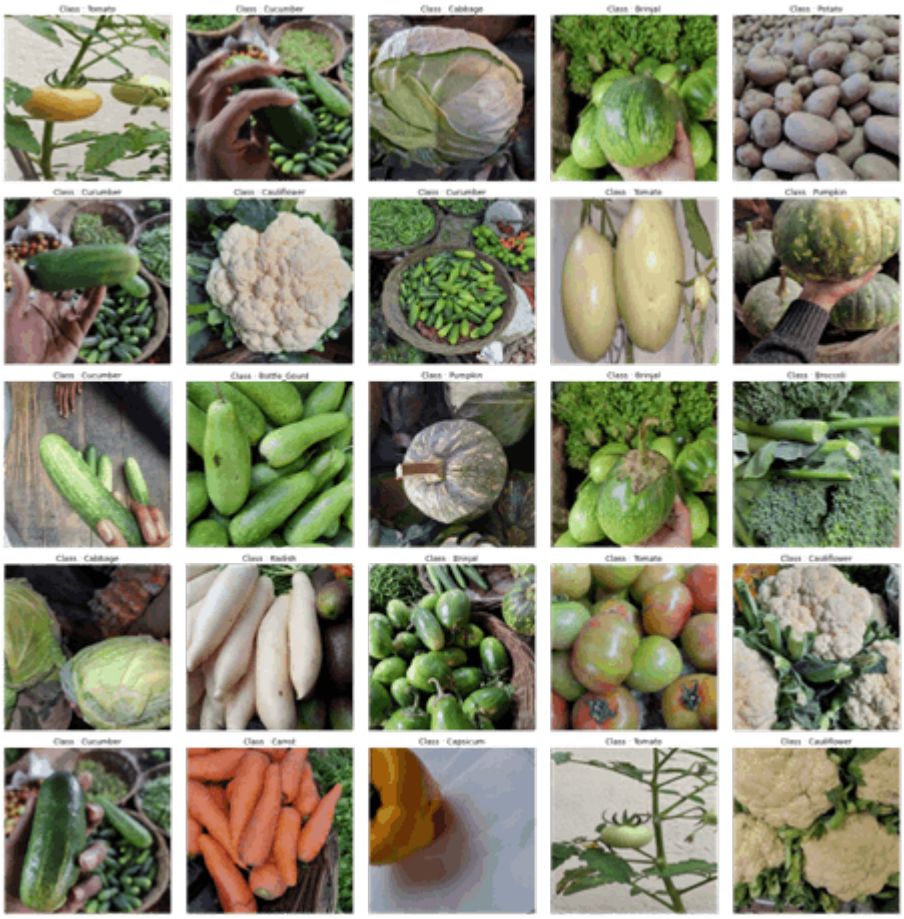
Load Data Model

In [8]:
```
input_shape = (224,224,3)
input_shape2 = (224,224)
BATCH = 32
```

In [9]:
```
# All images will be rescaled by 1./255
train_datagen = ImageDataGenerator(rescale=1/255)
test_datagen = ImageDataGenerator(rescale=1/255)
validation_datagen = ImageDataGenerator(rescale=1/255)

# Flow training images in batches of 370 using train_datagen generator
train_gen = train_datagen.flow_from_directory(
        root_path,  # This is the source directory for training images
        class_mode ='binary',
        target_size= input_shape2,  # All images will be resized to 200x200
        shuffle = True,
        batch_size=BATCH)

test_gen = train_datagen.flow_from_directory(
        root_path.replace('train','test'),  # This is the source directory for training
images
        class_mode ='binary',
        target_size= input_shape2,  # All images will be resized to 200x200
        shuffle = True,
        batch_size=BATCH)

# Flow validation images in batches of 20 using valid_datagen generator
validation_gen = validation_datagen.flow_from_directory(
        root_path.replace('train','validation'),  # This is the source directory for tr
aining images
        class_mode ='binary',
        target_size= input_shape2,  # All images will be resized to 200x200
        shuffle = True,
        batch_size=BATCH)
```

```
Found 15000 images belonging to 15 classes.
Found 3000 images belonging to 15 classes.
Found 3000 images belonging to 15 classes.
```

Menampilkan Beberapa Foto Testing



Pembuatan Model

Bentuk model yang telah dibuat terinspirasi dari VGG 16 dengan jumlah parameter sebesar 6,422,255 node

Model

In [11]:

```
model1 = Sequential([])

model1.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=input_shape))
model1.add(Conv2D(32, kernel_size=(3, 3), activation='relu'))
model1.add(MaxPooling2D(pool_size=(2, 2)))

model1.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
model1.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
model1.add(MaxPooling2D(pool_size=(2, 2)))

model1.add(Conv2D(96, kernel_size=(3, 3), activation='relu'))
model1.add(MaxPooling2D(pool_size=(2, 2)))

model1.add(Conv2D(96, kernel_size=(3, 3), activation='relu'))
model1.add(MaxPooling2D(pool_size=(2, 2)))

model1.add(Dropout(0.25))

model1.add(Flatten())

model1.add(Dense(512, activation='relu'))
model1.add(Dropout(0.5))
model1.add(Dense(512, activation='relu'))
model1.add(Dense(n_classes, activation='softmax'))
```

Summary dari model tersebut.

In [12]:
```
# Architecture
model1.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 222, 222, 32)      896
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 220, 220, 32)      9248
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 110, 110, 32)      0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 108, 108, 64)      18496
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 106, 106, 64)      36928
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 53, 53, 64)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 51, 51, 96)        55392
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 25, 25, 96)        0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 23, 23, 96)        83040
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 11, 11, 96)        0
_________________________________________________________________
dropout (Dropout)            (None, 11, 11, 96)        0
_________________________________________________________________
flatten (Flatten)            (None, 11616)             0
_________________________________________________________________
dense (Dense)                (None, 512)               5947904
_________________________________________________________________
dropout_1 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               262656
_________________________________________________________________
dense_2 (Dense)              (None, 15)                7695
=================================================================
Total params: 6,422,255
Trainable params: 6,422,255
Non-trainable params: 0
_________________________________________________________________
```

Compiling dan Callback Model

Optimizer yang digunakan adalah adam dengan leraning rate = 0.0001, sedangkan loss yang digunakan adalah sparse categorical crossentropy yang cocok untuk data multiclass.

In [13]:
```
opt = tf.keras.optimizers.Adam(learning_rate=0.0001)
model1.compile(optimizer = opt,
               loss = 'sparse_categorical_crossentropy',
               metrics=['accuracy'])
```

In [14]:
```
cbs = [EarlyStopping(patience=5, restore_best_weights=True),
       ModelCheckpoint('model.best.h5', save_best_only=True)]
```

In [15]:
```
step = 15000//BATCH
valid_step = 3000//BATCH
```

Data Fitting

Epochs yang digunakan 100, namun terdapat callback untuk menghentikan data fitting sehingga tidak overfitting. Hasil fitting yang diperoleh berjalanan selama 24 menit dan 29 detik serta diperoleh 18 Epoch dengan nilai loss sebesar 0.0509, accuracy sebesar 0.9836, val_loss sebesar 0.1202, dan val_accuracy sebesar 0.9735.

In [16]:

```
%%time
hist_1 = model1.fit(
      train_gen,
      validation_data = validation_gen,
      validation_steps = valid_step,
      callbacks = cbs,
      epochs = 100,
      steps_per_epoch = step,
      batch_size = BATCH,
      verbose = 1)

# ipd.Audio(audio_path, autoplay=True)
```

```
2022-11-22 03:21:45.398952: I tensorflow/compiler/mlir/mlir_graph_optimization_pas
s.cc:185] None of the MLIR Optimization Passes are enabled (registered 2)

Epoch 1/100

2022-11-22 03:21:47.107966: I tensorflow/stream_executor/cuda/cuda_dnn.cc:369] Load
ed cuDNN version 8005

468/468 [==============================] - 124s 247ms/step - loss: 1.7451 - accurac
y: 0.3994 - val_loss: 1.1134 - val_accuracy: 0.6284
Epoch 2/100
468/468 [==============================] - 49s 105ms/step - loss: 0.8931 - accurac
y: 0.7043 - val_loss: 0.5907 - val_accuracy: 0.8105
Epoch 3/100
468/468 [==============================] - 47s 101ms/step - loss: 0.5639 - accurac
y: 0.8123 - val_loss: 0.4655 - val_accuracy: 0.8629
```

```
Epoch 15/100
468/468 [==============================] - 49s 105ms/step - loss: 0.0548 - accurac
y: 0.9820 - val_loss: 0.1787 - val_accuracy: 0.9570
Epoch 16/100
468/468 [==============================] - 49s 104ms/step - loss: 0.0568 - accurac
y: 0.9818 - val_loss: 0.2721 - val_accuracy: 0.9435
Epoch 17/100
468/468 [==============================] - 47s 101ms/step - loss: 0.0486 - accurac
y: 0.9837 - val_loss: 0.1552 - val_accuracy: 0.9640
Epoch 18/100
468/468 [==============================] - 49s 104ms/step - loss: 0.0509 - accurac
y: 0.9836 - val_loss: 0.1202 - val_accuracy: 0.9735
CPU times: user 17min 24s, sys: 43.7 s, total: 18min 7s
Wall time: 24min 29s
```

Evaluasi dari Data Testing

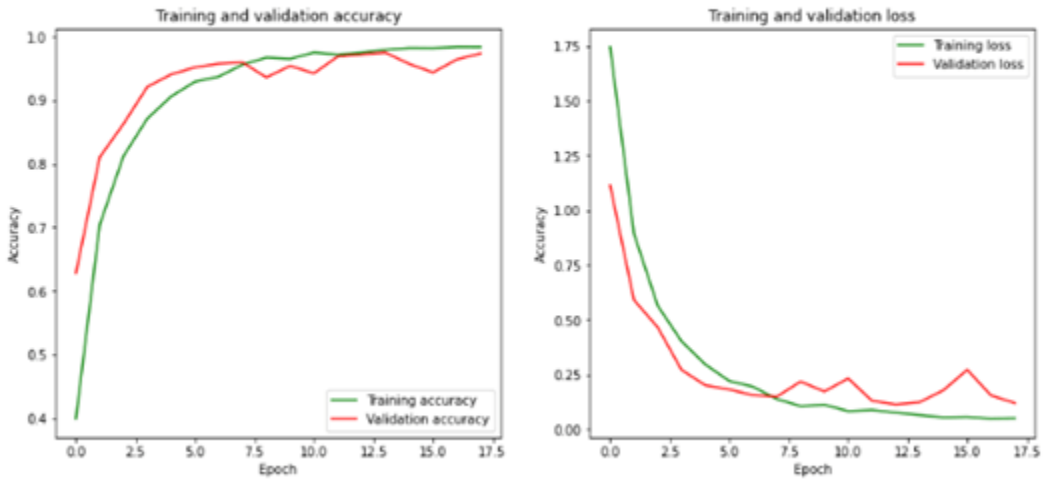
In [17]:
```
m_eval = model1.evaluate(test_gen)
print(m_eval)
# ipd.Audio(audio_path, autoplay=True)
```

```
94/94 [==============================] - 27s 293ms/step - loss: 0.0985 - accuracy:
0.9730
[0.09850931912660599, 0.9729999899864197]
```

Dari hasil evaluasi data diperoleh accuracy sebesar 0.9729 atau 97,29%

Plot Accuracy dan Loss

In [18]:
```
plot_accuracy_and_loss(hist_1)
```



Dari hasil plot, bentuk grafik antara training dan validation memiliki jarak yang cukup kecil. Hal tersebut menunjukkan bahwa model yang diperoleh tidak overfit.

Menebak Gambar Testing dari Model

Reference

Model Inpiration: VGG-16, Practical Machine Learning and Image Processing karya Himanshu Singh terbitan tahun 2019.

Klasifikasi dengan ResNet50V2: https://www.kaggle.com/code/utkarshsaxenadn/vegetable-classification-resnet50v2-acc-99

Dataset: https://www.kaggle.com/datasets/misrakahmed/vegetable-image-dataset

CNN: https://medium.com/@16611110/apa-itu-convolutional-neural-network-836f70b193a4

Github: https://www.kaggle.com/code/difafarhanihakim/vegetable-cnn?scriptVersionId=111704685