BAB IV

HASIL PENELITIAN DAN PEMBAHASAN

4.1 Pengambilan Dataset

Dataset yang digunakan dalam penelitian ini berasal dari kaggle, pengunduhan dataset dilakukan melalui *google colab* drive.

```
from google.colab import drive
drive.mount('/content/drive')
```

Gambar 4.1 Upload File Pengambilan Dataset

Gambar 4.1 menunjukkan proses pengunggahan file dataset dari kaggle ke *google drive* menggunakan *library files* di *google colab*. Baris pertama kode, `*from google.colab import drive*`, mengimport library files dari *google colab*, yang menyediakan fungsi-fungsi untuk berinteraksi dengan file di *colab*.

4.2 Pelatihan Model VGG-19

```
[ ] from tensorflow import keras
    from tensorflow.keras.layers import Dense,Flatten
    from tensorflow.keras.models import Model, Sequential,load_model
    from tensorflow.keras.preprocessing import image
    from tensorflow.keras.preprocessing.image import ImageDataGenerator
    from tensorflow.keras.applications import VGG19
    import pandas as pd
    import numpy as np
    import matplotlib.pyplot as plt
    from glob import glob
```

Gambar 4.2 Import VGG-19

Gambar 4.2 merupakan *import* dari beberapa modul, fungsi yang diperlukan untuk membangun dan melatih model *neural network* menggunakan *TensorFlow*. Modul-modul yang di import antara lain:

1. `keras`: untuk membangun dan melatih model *neural network*.
2. `*Dense*` dan `*Flatten*`: untuk menambahkan *layer-layer neural network.*
3. `Model`, `*Sequential*`, dan `load_model`: untuk mendefinisikan dan memuat model *neural network.*

4. `*image*` dan `*ImageDataGenerator*`: untuk memproses dan melakukan augmentasi pada gambar.
5. `VGG19`: untuk menggunakan arsitektur model VGG-19.
6. `pandas`: untuk manipulasi data.
7. `*numpy*`: untuk operasi matematika.
8. `*matplotlib.pyplot*`: untuk visualisasi data.
9. `glob`: untuk mencocokkan *pathnames* dengan pola tertentu.

Ini adalah *import* yang umum digunakan dalam pembuatan dan pelatihan model *neural network* menggunakan *TensorFlow*.

```
[ ] train_datagen=ImageDataGenerator(rescale=1./255,
                                     horizontal_flip=True,shear_range=.2,rotation_range=.2)
    test_datagen=ImageDataGenerator(rescale=1./255)
```

Gambar 4.3 Image Data Generator

Kode di atas mendefinisikan dua objek `*ImageDataGenerator*`: `*train_datagen*` dan `*test_datagen*`. Objek `*train_datagen*` digunakan untuk melakukan augmentasi data pada data latih, termasuk *rescaling* gambar ke rentang 0-1, penerapan horizontal flip (memutar gambar secara horizontal), serta penerapan *shear* dan rotasi gambar. Objek `*test_datagen*` hanya melakukan *rescaling* gambar ke rentang 0-1 tanpa augmentasi lainnya.

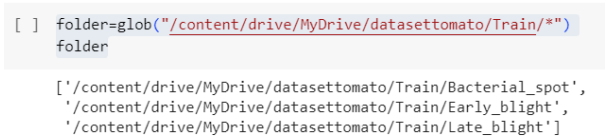
```
IMAGE_SIZE=256
BATCH_SIZE=32
```

Gambar 4.4 Variabel dalam Gambar

Gambar 4.4 merupakan variabel `*IMAGE_SIZE*` ditetapkan nilainya sebesar 256, yang mengindikasikan bahwa seluruh gambar akan diubah ukurannya menjadi 256x256 piksel. Sedangkan `*BATCH_SIZE*` ditetapkan sebesar 32, yang menandakan bahwa dalam setiap iterasi pelatihan, model akan melihat 32 gambar sekaligus.

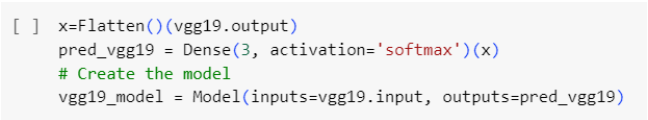
Gambar 4.5 Menentukan Folder *Training*, *Test*, dan *Validation*

Gambar 4.5 menggunakan objek `*train_datagen*` dan `*test_datagen*` yang telah didefinisikan sebelumnya untuk memuat data latih, data uji, dan data validasi dari direktori yang sesuai. Ditemukan 1904 gambar yang termasuk ke dalam 3 kelas penyakit daun tomat pada data *training*. Selain itu, 270 gambar pada data *testing* dan 540 gambar pada data *validation*, yang juga memiliki 3 kelas yang sama.



Gambar 4.6 Membuka Folder Data Training

Gambar 4.6 merupakan fungsi `glob` untuk mencocokkan semua *pathnames* yang sesuai dengan pola yang diberikan, yaitu semua folder di dalam direktori "/content/drive/MyDrive/datasettomato/Train/". Hasilnya adalah daftar *pathnames* dari semua folder di dalam direktori tersebut, yang disimpan dalam variabel `folder`. Variabel folder berisi daftar pathnames dari tiga folder di dalam direktori "/content/drive/MyDrive/datasettomato/Train/". Ketiga folder tersebut berisi *Bacterial spot*, *Early Blight* dan *Late Blight*.



Gambar 4.7 *Activation Softmax*

Gambar 4.7 mengambil output dari model VGG-19 dan melakukan operasi *flatten* (menjadikan output menjadi satu dimensi). Kemudian, ditambahkan *layer Dense* dengan 3 *neuron* (sesuai dengan jumlah kelas yang akan diprediksi) dan menggunakan fungsi aktivasi *softmax* untuk menghasilkan probabilitas kelas-kelas.

Selanjutnya, model baru dibuat menggunakan objek `Model`, dengan input dari model VGG-19 dan output dari *layer* prediksi yang baru ditambahkan. Dengan demikian, model baru ini akan memiliki struktur yang sama dengan VGG-19, tetapi dengan output yang dimodifikasi sesuai dengan tujuan klasifikasi kita.

```
[ ]  vgg19_model.compile(optimizer="Adam",loss="categorical_crossentropy",metrics=["accuracy"])
```

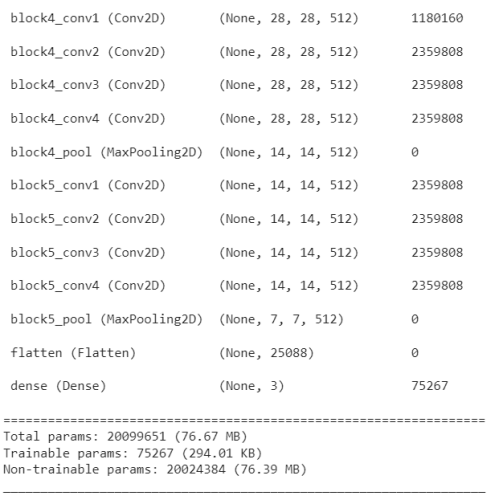
Gambar 4.8 *Optimizer* Adam

Gambar 4.8 mengompilasi model `vgg19_model` yang telah didefinisikan sebelumnya. Proses kompilasi ini melibatkan konfigurasi berbagai parameter yang diperlukan untuk melatih model:

1. `*optimizer*="Adam"`: Digunakan untuk menentukan algoritma optimisasi yang akan digunakan selama proses pelatihan. Di sini, "Adam" digunakan, yang merupakan algoritma optimisasi yang efisien dan sering digunakan.
2. `*loss*="*categorical_crossentropy*"`: Menentukan fungsi loss yang akan dioptimalkan selama pelatihan model. "*Categorical crossentropy*" digunakan karena kita memiliki tugas klasifikasi multi-kelas.
3. `*metrics*=["*accuracy*"]`: Mendefinisikan metrik yang akan digunakan untuk mengevaluasi kinerja model selama pelatihan. Dalam hal ini, kita hanya tertarik pada akurasi sebagai metrik evaluasi.

4.3 Model VGG-19

```
[ ]  vgg19_model.summary()

Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 224, 224, 3)]     0

 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792

 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928

 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0

 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856

 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584

 block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0

 block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168

 block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080

 block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080

 block3_conv4 (Conv2D)       (None, 56, 56, 256)       590080

 block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0

 block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160
```



```
block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160
block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808
block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808
block4_conv4 (Conv2D)       (None, 28, 28, 512)       2359808
block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0
block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808
block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808
block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808
block5_conv4 (Conv2D)       (None, 14, 14, 512)       2359808
block5_pool (MaxPooling2D)  (None, 7, 7, 512)         0
flatten (Flatten)           (None, 25088)             0
dense (Dense)               (None, 3)                 75267
=================================================================
Total params: 20099651 (76.67 MB)
Trainable params: 75267 (294.01 KB)
Non-trainable params: 20024384 (76.39 MB)
_________________________________________________________________
```

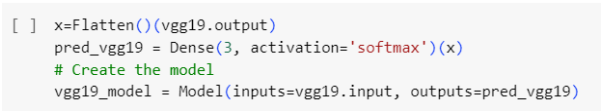
Gambar 4.9 Model VGG-19

Gambar 4.9 adalah representasi model yang telah didefinisikan, dengan struktur *layer-layer* yang dibangun di atas arsitektur VGG-19. Jumlah parameter untuk setiap layer juga disediakan:

1. `*InputLayer*`: Mengambil input gambar dengan ukuran 224x224 piksel dan 3 *channel* warna (RGB).
2. `Conv2D` dan `*MaxPooling*2D`: Merupakan bagian dari lapisan konvolusi dan pengelompokan (*pooling*) dari VGG-19.
3. `*Flatten*`: Mengubah output dari lapisan konvolusi menjadi satu dimensi agar dapat dihubungkan ke lapisan *Dense*.
4. `*Dense*`: Lapisan *Dense* terakhir dengan 3 neuron (sesuai dengan jumlah kelas) dan menggunakan fungsi aktivasi *softmax*.

Total parameter model adalah 20,099,651, di mana sebagian besar parameter (20,024,384) tidak dapat dilatih (*non-trainable*), dan hanya 75,267 parameter yang dapat dilatih (*trainable*).

4.4 Modifikasi *Epoch* 18 Aktivasi *Softmax* dan *Optimizer* Adam



```
[ ]  x=Flatten()(vgg19.output)
     pred_vgg19 = Dense(3, activation='softmax')(x)
     # Create the model
     vgg19_model = Model(inputs=vgg19.input, outputs=pred_vgg19)
```

```
[ ]  vgg19_model.compile(optimizer="Adam",loss="categorical_crossentropy",metrics=["accuracy"])
```

```
# Melatih model
history = vgg19_model.fit(
    train_generator,
    validation_data=test_generator,
    epochs=18
)
```

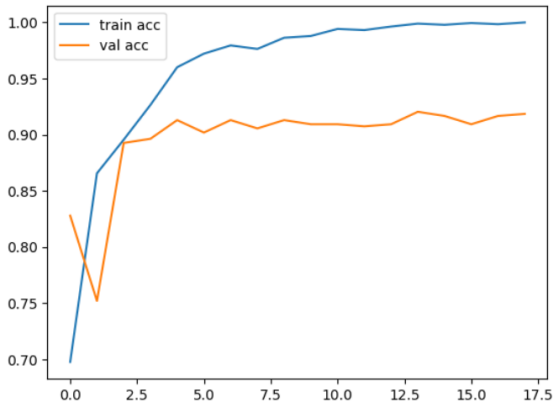
Gambar 4.10 Modifikasi *Epoch* 18 Aktivasi *Softmax* dan *Optimizer* Adam

Pada Gambar 4.10, saya memutuskan untuk mengubah *epoch* 17 yang digunakan dalam model sebelumnya ke *epoch* 18. Dengan aktivasi yang sama dengan model sebelumnya yaitu *softmax* dan *optimizer* adam.

```
Epoch 2/18
60/60 [==============================] - 34s 565ms/step - loss: 0.3343 - accuracy: 0.8655 - val_loss: 0.5454 - val_accuracy: 0.7519
Epoch 3/18
60/60 [==============================] - 35s 584ms/step - loss: 0.2716 - accuracy: 0.8955 - val_loss: 0.2907 - val_accuracy: 0.8926
Epoch 4/18
60/60 [==============================] - 33s 543ms/step - loss: 0.2065 - accuracy: 0.9265 - val_loss: 0.2725 - val_accuracy: 0.8963
Epoch 5/18
60/60 [==============================] - 35s 581ms/step - loss: 0.1446 - accuracy: 0.9601 - val_loss: 0.2458 - val_accuracy: 0.9130
Epoch 6/18
60/60 [==============================] - 34s 560ms/step - loss: 0.1184 - accuracy: 0.9722 - val_loss: 0.2369 - val_accuracy: 0.9019
Epoch 7/18
60/60 [==============================] - 35s 591ms/step - loss: 0.1005 - accuracy: 0.9795 - val_loss: 0.2242 - val_accuracy: 0.9130
Epoch 8/18
60/60 [==============================] - 33s 550ms/step - loss: 0.0932 - accuracy: 0.9764 - val_loss: 0.2511 - val_accuracy: 0.9056
Epoch 9/18
60/60 [==============================] - 33s 552ms/step - loss: 0.0755 - accuracy: 0.9863 - val_loss: 0.2416 - val_accuracy: 0.9130
Epoch 10/18
60/60 [==============================] - 34s 564ms/step - loss: 0.0581 - accuracy: 0.9879 - val_loss: 0.2215 - val_accuracy: 0.9093
Epoch 11/18
60/60 [==============================] - 34s 564ms/step - loss: 0.0487 - accuracy: 0.9942 - val_loss: 0.2197 - val_accuracy: 0.9093
Epoch 12/18
60/60 [==============================] - 35s 570ms/step - loss: 0.0458 - accuracy: 0.9932 - val_loss: 0.2449 - val_accuracy: 0.9074
Epoch 13/18
60/60 [==============================] - 33s 553ms/step - loss: 0.0442 - accuracy: 0.9963 - val_loss: 0.2481 - val_accuracy: 0.9093
Epoch 14/18
60/60 [==============================] - 33s 562ms/step - loss: 0.0340 - accuracy: 0.9989 - val_loss: 0.2505 - val_accuracy: 0.9204
Epoch 15/18
60/60 [==============================] - 33s 553ms/step - loss: 0.0320 - accuracy: 0.9979 - val_loss: 0.2231 - val_accuracy: 0.9167
Epoch 16/18
60/60 [==============================] - 34s 563ms/step - loss: 0.0260 - accuracy: 0.9995 - val_loss: 0.2172 - val_accuracy: 0.9093
Epoch 17/18
60/60 [==============================] - 33s 545ms/step - loss: 0.0282 - accuracy: 0.9984 - val_loss: 0.2157 - val_accuracy: 0.9167
Epoch 18/18
60/60 [==============================] - 35s 586ms/step - loss: 0.0215 - accuracy: 1.0000 - val_loss: 0.2119 - val_accuracy: 0.9185
```

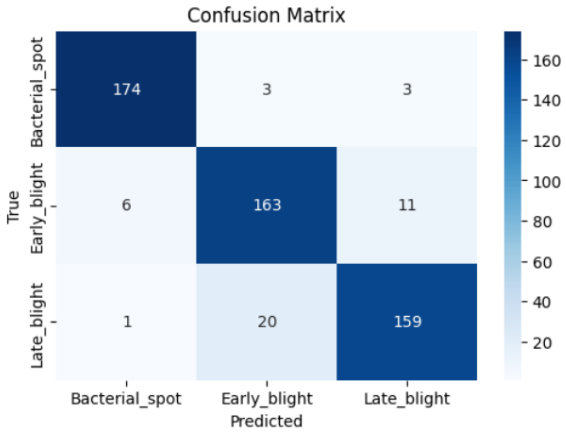
Gambar 4.11 Hasil Modifikasi *Epoch* 18 Aktivasi *Softmax* dan *Optimizer* Adam

Hasil pelatihan model pada Gambar 4.11, pada awalnya terjadi penurunan yang stabil pada *loss* dan peningkatan akurasi pada kedua dataset, *testing* dan *validation* yang menunjukkan bahwa model mempelajari dengan baik dari data *testing*. Pada *epoch* 5, terjadi akurasi yang sangat baik pada data validation 0.9130%, menandakan bahwa model mampu mengklasifikasikan data yang belum pernah dilihat sebelumnya dengan sangat baik. Selain itu, karena akurasi pada data *validation* meningkat seiring waktu, menunjukkan bahwa model tidak hanya mempelajari data pelatihan dengan baik, tetapi juga mampu melakukan generalisasi dengan baik pada data baru. Namun, perlu diperhatikan bahwa pelatihan model dengan jumlah *epoch* 18 membutuhkan waktu yang cukup lama. Dengan demikian, hasil ini menunjukkan bahwa model telah berhasil dilatih dengan baik dan mampu mengklasifikasikan data dengan tingkat akurasi yang tinggi.



Gambar 4.12 Grafik Hasil Modifikasi *Epoch* 18 Aktivasi *Softmax* dan *Optimizer* Adam

Pada grafik diatas, bahwa akurasi pada data *training* adalah 0.10000, kemudian untuk data validasinya adalah 0.9185. Akurasi model dari data *train* dan data *testing* jika dilihat dari *epoch* 1 sampai 18 dominan naik, namun juga ada penurunan pada epoch tertentu, kenaikan dan penurunan akurasi model terjadi karena data yang diprediksi benar setiap iterasi selalu berbeda atau naik turun.



Gambar 4.13 *Confusion Matrix*

Berikut contoh perhitungan untuk mendapatkan nilai akurasi untuk hasil evaluasi klasifikasi:

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

$$= \frac{174 + 163 + 159}{174 + 163 + 159 + 3 + 3 + 6 + 11 + 1 + 20}$$

$$= \frac{496}{540} = 0{,}9185 \times 100\% = 91{,}85\%$$

4.5 Modifikasi *Epoch* 17 Aktivasi *Softmax* dan *Optimizer* SGD

```
x=Flatten()(vgg19.output)
pred_vgg19 = Dense(3, activation='softmax')(x)
# Create the model
vgg19_model = Model(inputs=vgg19.input, outputs=pred_vgg19)
```

```
[12] vgg19_model.compile(optimizer="SGD",loss="categorical_crossentropy",metrics=["accuracy"])
```

```
[15] # Melatih model
     history = vgg19_model.fit(
         train_generator,
         validation_data=test_generator,
         epochs=17
     )
```

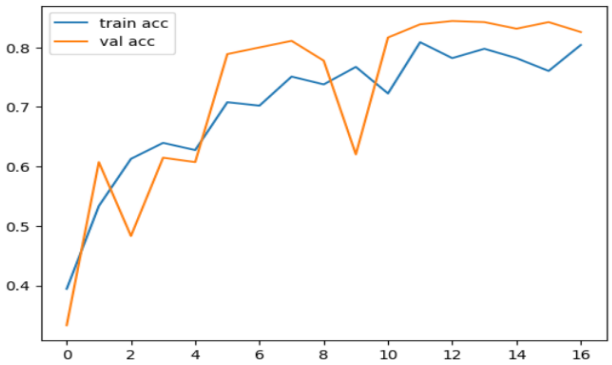
Gambar 4.14 Modifikasi *Epoch* 17 Aktivasi *Softmax* dan *Optimizer* SGD

Gambar 4.14 mengompilasi model VGG-19 dengan *optimizer* SGD. Kemudian melatih model tersebut dengan data *training* dan data *validation* dengan *epoch* 17.

```
Epoch 1/17
60/60 [==============================] - 459s 8s/step - loss: 18.9222 - accuracy: 0.3944 - val_loss: 26.9640 - val_accuracy: 0.3333
Epoch 2/17
60/60 [==============================] - 37s 613ms/step - loss: 10.1790 - accuracy: 0.5336 - val_loss: 13.8240 - val_accuracy: 0.6074
Epoch 3/17
60/60 [==============================] - 34s 566ms/step - loss: 6.1261 - accuracy: 0.6129 - val_loss: 14.7362 - val_accuracy: 0.4833
Epoch 4/17
60/60 [==============================] - 35s 585ms/step - loss: 5.9260 - accuracy: 0.6397 - val_loss: 5.1694 - val_accuracy: 0.6148
Epoch 5/17
60/60 [==============================] - 34s 568ms/step - loss: 5.3567 - accuracy: 0.6276 - val_loss: 10.5401 - val_accuracy: 0.6074
Epoch 6/17
60/60 [==============================] - 34s 567ms/step - loss: 4.2218 - accuracy: 0.7080 - val_loss: 1.3839 - val_accuracy: 0.7889
Epoch 7/17
60/60 [==============================] - 34s 565ms/step - loss: 4.2011 - accuracy: 0.7022 - val_loss: 1.4278 - val_accuracy: 0.8000
Epoch 8/17
60/60 [==============================] - 35s 576ms/step - loss: 2.8497 - accuracy: 0.7511 - val_loss: 1.2592 - val_accuracy: 0.8111
Epoch 9/17
60/60 [==============================] - 34s 564ms/step - loss: 2.9731 - accuracy: 0.7379 - val_loss: 2.5113 - val_accuracy: 0.7778
Epoch 10/17
60/60 [==============================] - 35s 582ms/step - loss: 2.4114 - accuracy: 0.7673 - val_loss: 4.8633 - val_accuracy: 0.6204
Epoch 11/17
60/60 [==============================] - 37s 617ms/step - loss: 3.3697 - accuracy: 0.7227 - val_loss: 1.5188 - val_accuracy: 0.8167
Epoch 12/17
60/60 [==============================] - 34s 573ms/step - loss: 1.9967 - accuracy: 0.8088 - val_loss: 1.1507 - val_accuracy: 0.8389
Epoch 13/17
60/60 [==============================] - 35s 587ms/step - loss: 2.5645 - accuracy: 0.7820 - val_loss: 1.2697 - val_accuracy: 0.8444
Epoch 14/17
60/60 [==============================] - 37s 613ms/step - loss: 2.0995 - accuracy: 0.7978 - val_loss: 1.1421 - val_accuracy: 0.8426
Epoch 15/17
60/60 [==============================] - 34s 566ms/step - loss: 2.1759 - accuracy: 0.7820 - val_loss: 1.1514 - val_accuracy: 0.8315
Epoch 16/17
60/60 [==============================] - 37s 617ms/step - loss: 3.2163 - accuracy: 0.7605 - val_loss: 1.0896 - val_accuracy: 0.8426
Epoch 17/17
60/60 [==============================] - 34s 560ms/step - loss: 2.2448 - accuracy: 0.8041 - val_loss: 1.3764 - val_accuracy: 0.8259
```

Gambar 4.15 Hasil Modifikasi *Epoch* 17 Aktivasi *Softmax* dan *Optimizer* SGD

Pada proses pelatihan model VGG-19 pada Gambar 4.15 setiap *epoch* merepresentasikan satu kali iterasi melalui seluruh data latih. Dalam setiap *epoch*, model VGG-19 menggunakan algoritma optimisasi yang disebut dengan SGD untuk menyesuaikan bobotnya berdasarkan nilai *loss* yang dihasilkan. Pada awal pelatihan, terlihat penurunan loss dan peningkatan akurasi pada setiap epoch. Hal ini menunjukkan bahwa model mempelajari dan semakin baik dalam memprediksi kelas gambar-gambar yang diberikan.



Gambar 4.16 Grafik Hasil Modifikasi *Epoch* 17 Aktivasi *Softmax* dan *Optimizer* SGD

4.6 Modifikasi *Epoch* 17 Aktivasi *Softmax* dan *Optimizer* Adagrad

```
[11] x=Flatten()(vgg19.output)
     pred_vgg19 = Dense(3, activation='softmax')(x)
     # Create the model
     vgg19_model = Model(inputs=vgg19.input, outputs=pred_vgg19)
```

```
[18] vgg19_model.compile(optimizer="Adagrad",loss="categorical_crossentropy",metrics=["accuracy"])
```

```
[19] # Melatih model
     history = vgg19_model.fit(
         train_generator,
         validation_data=test_generator,
         epochs=17
     )
```

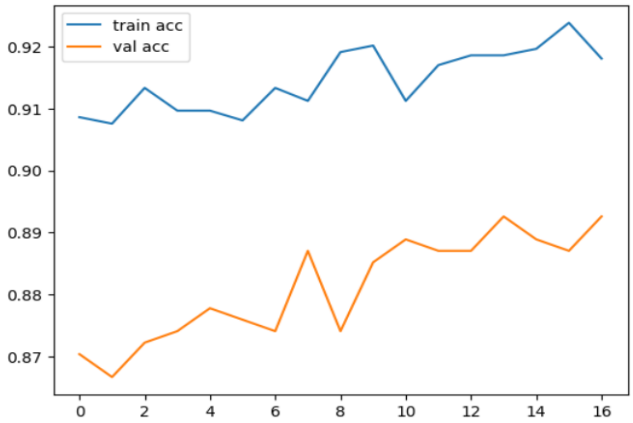
Gambar 4.17 Modifikasi *Epoch* 17 Aktivasi *Softmax* dan *Optimizer* Adagrad

Gambar 4.17 memodifikasikan *optimizer* menggunakan Adagrad untuk mendapatkan perbandingan hasil yang lain.

```
Epoch 1/17
60/60 [==============================] - 35s 559ms/step - loss: 0.5652 - accuracy: 0.9086 - val_loss: 0.8801 - val_accuracy: 0.8704
Epoch 2/17
60/60 [==============================] - 34s 564ms/step - loss: 0.5560 - accuracy: 0.9076 - val_loss: 0.8634 - val_accuracy: 0.8667
Epoch 3/17
60/60 [==============================] - 34s 560ms/step - loss: 0.5053 - accuracy: 0.9133 - val_loss: 0.8384 - val_accuracy: 0.8722
Epoch 4/17
60/60 [==============================] - 33s 548ms/step - loss: 0.5039 - accuracy: 0.9097 - val_loss: 0.8188 - val_accuracy: 0.8741
Epoch 5/17
60/60 [==============================] - 36s 599ms/step - loss: 0.4907 - accuracy: 0.9097 - val_loss: 0.8050 - val_accuracy: 0.8778
Epoch 6/17
60/60 [==============================] - 34s 564ms/step - loss: 0.4388 - accuracy: 0.9081 - val_loss: 0.7936 - val_accuracy: 0.8759
Epoch 7/17
60/60 [==============================] - 35s 577ms/step - loss: 0.4364 - accuracy: 0.9133 - val_loss: 0.7777 - val_accuracy: 0.8741
Epoch 8/17
60/60 [==============================] - 34s 559ms/step - loss: 0.4324 - accuracy: 0.9112 - val_loss: 0.7609 - val_accuracy: 0.8870
Epoch 9/17
60/60 [==============================] - 35s 580ms/step - loss: 0.4336 - accuracy: 0.9191 - val_loss: 0.7698 - val_accuracy: 0.8741
Epoch 10/17
60/60 [==============================] - 35s 578ms/step - loss: 0.4238 - accuracy: 0.9202 - val_loss: 0.7409 - val_accuracy: 0.8852
Epoch 11/17
60/60 [==============================] - 34s 569ms/step - loss: 0.4596 - accuracy: 0.9112 - val_loss: 0.7277 - val_accuracy: 0.8889
Epoch 12/17
60/60 [==============================] - 34s 560ms/step - loss: 0.4090 - accuracy: 0.9170 - val_loss: 0.7239 - val_accuracy: 0.8870
Epoch 13/17
60/60 [==============================] - 33s 555ms/step - loss: 0.3802 - accuracy: 0.9186 - val_loss: 0.7147 - val_accuracy: 0.8870
Epoch 14/17
60/60 [==============================] - 35s 575ms/step - loss: 0.3783 - accuracy: 0.9186 - val_loss: 0.7068 - val_accuracy: 0.8926
Epoch 15/17
60/60 [==============================] - 33s 555ms/step - loss: 0.3819 - accuracy: 0.9196 - val_loss: 0.6996 - val_accuracy: 0.8889
Epoch 16/17
60/60 [==============================] - 35s 578ms/step - loss: 0.3689 - accuracy: 0.9238 - val_loss: 0.6921 - val_accuracy: 0.8870
Epoch 17/17
60/60 [==============================] - 34s 560ms/step - loss: 0.3644 - accuracy: 0.9181 - val_loss: 0.6862 - val_accuracy: 0.8926
```

Gambar 4.18 Hasil Modifikasi *Epoch* 17 Aktivasi *Softmax* dan *Optimizer* Adagrad

Gambar 4.18 pelatihan pada *epoch* 17 dengan akurasi 0.9181% dengan akurasi validasi 0.8926%.



Gambar 4.19 Grafik Hasil Modifikasi *Epoch* 17 Aktivasi *Softmax* dan *Optimizer* Adagrad

4.7 Modifikasi *Epoch* 17 Aktivasi *Softmax* dan *Optimizer* RMSProp

```
[11] x=Flatten()(vgg19.output)
     pred_vgg19 = Dense(3, activation='softmax')(x)
     # Create the model
     vgg19_model = Model(inputs=vgg19.input, outputs=pred_vgg19)
```

```
vgg19_model.compile(optimizer="RMSProp",loss="categorical_crossentropy",metrics=["accuracy"])
```

```
# Melatih model
history = vgg19_model.fit(
    train_generator,
    validation_data=test_generator,
    epochs=17
)
```

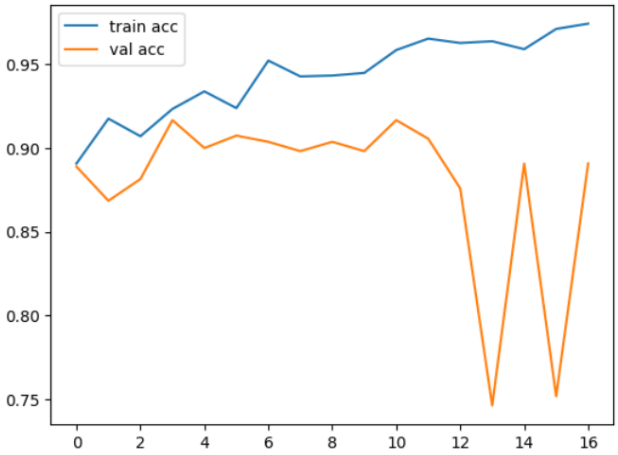
Gambar 4.20 Modifikasi *Epoch* 17 Aktivasi *Softmax* dan *Optimizer* RMSProp

Gambar 4.20 memodifikasikan *optimizer* menggunakan RMSProp untuk mendapatkan perbandingan hasil yang lain.

```
Epoch 1/17
60/60 [==============================] - 35s 565ms/step - loss: 0.5562 - accuracy: 0.8908 - val_loss: 0.6153 - val_accuracy: 0.8889
Epoch 2/17
60/60 [==============================] - 34s 563ms/step - loss: 0.3687 - accuracy: 0.9175 - val_loss: 0.8076 - val_accuracy: 0.8685
Epoch 3/17
60/60 [==============================] - 34s 572ms/step - loss: 0.3479 - accuracy: 0.9070 - val_loss: 0.5355 - val_accuracy: 0.8815
Epoch 4/17
60/60 [==============================] - 33s 549ms/step - loss: 0.2836 - accuracy: 0.9233 - val_loss: 0.4399 - val_accuracy: 0.9167
Epoch 5/17
60/60 [==============================] - 34s 573ms/step - loss: 0.2551 - accuracy: 0.9338 - val_loss: 0.5036 - val_accuracy: 0.9000
Epoch 6/17
60/60 [==============================] - 34s 560ms/step - loss: 0.2627 - accuracy: 0.9238 - val_loss: 0.3695 - val_accuracy: 0.9074
Epoch 7/17
60/60 [==============================] - 35s 582ms/step - loss: 0.1922 - accuracy: 0.9522 - val_loss: 0.4122 - val_accuracy: 0.9037
Epoch 8/17
60/60 [==============================] - 36s 598ms/step - loss: 0.1780 - accuracy: 0.9428 - val_loss: 0.4242 - val_accuracy: 0.8981
Epoch 9/17
60/60 [==============================] - 35s 583ms/step - loss: 0.1986 - accuracy: 0.9433 - val_loss: 0.3465 - val_accuracy: 0.9037
Epoch 10/17
60/60 [==============================] - 36s 592ms/step - loss: 0.1894 - accuracy: 0.9449 - val_loss: 0.3683 - val_accuracy: 0.8981
Epoch 11/17
60/60 [==============================] - 36s 604ms/step - loss: 0.1836 - accuracy: 0.9585 - val_loss: 0.3531 - val_accuracy: 0.9167
Epoch 12/17
60/60 [==============================] - 34s 565ms/step - loss: 0.1118 - accuracy: 0.9653 - val_loss: 0.4029 - val_accuracy: 0.9056
Epoch 13/17
60/60 [==============================] - 35s 581ms/step - loss: 0.1177 - accuracy: 0.9627 - val_loss: 0.5123 - val_accuracy: 0.8759
Epoch 14/17
60/60 [==============================] - 36s 599ms/step - loss: 0.1076 - accuracy: 0.9638 - val_loss: 1.6562 - val_accuracy: 0.7463
Epoch 15/17
60/60 [==============================] - 34s 563ms/step - loss: 0.1500 - accuracy: 0.9590 - val_loss: 0.4477 - val_accuracy: 0.8907
Epoch 16/17
60/60 [==============================] - 35s 592ms/step - loss: 0.1125 - accuracy: 0.9711 - val_loss: 1.6493 - val_accuracy: 0.7519
Epoch 17/17
60/60 [==============================] - 36s 607ms/step - loss: 0.0957 - accuracy: 0.9743 - val_loss: 0.5414 - val_accuracy: 0.8907
```

Gambar 4.21 Hasil Modifikasi *Epoch* 17 Aktivasi *Softmax* dan *Optimizer* RMSProp

Hasil pada Gambar 4.21 memerlukan waktu yang cukup lama dari sebelumnya, akurasi yang didapatkan rendah.



Gambar 4.22 Grafik Hasil Modifikasi *Epoch* 17 Aktivasi *Softmax* dan *Optimizer* RMSProp

4.8 Modifikasi *Epoch* 17 Aktivasi ReLu dan *Optimizer* Adam

```
[12] x=Flatten()(vgg19.output)
     pred_vgg19 = Dense(3, activation='relu')(x)
     # Create the model
     vgg19_model = Model(inputs=vgg19.input, outputs=pred_vgg19)

[13] vgg19_model.compile(optimizer="Adam",loss="categorical_crossentropy",metrics=["accuracy"])

     # Melatih model
     history = vgg19_model.fit(
         train_generator,
         validation_data=test_generator,
         epochs=17
     )
```

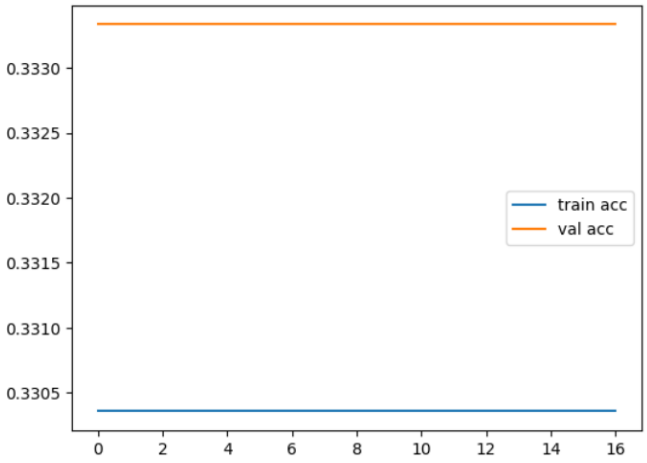
Gambar 4.23 Modifikasi *Epoch* 17 Aktivasi ReLu dan *Optimizer* Adam

Pada percobaan terakhir ini saya kembali memodifikasikan *epoch* 17 dengan aktivasi relu dan *optimizer* Adam.

```
Epoch 1/17
60/60 [==============================] - 37s 584ms/step - loss: nan - accuracy: 0.3304 - val_loss: nan - val_accuracy: 0.3333
Epoch 2/17
60/60 [==============================] - 35s 580ms/step - loss: nan - accuracy: 0.3304 - val_loss: nan - val_accuracy: 0.3333
Epoch 3/17
60/60 [==============================] - 34s 568ms/step - loss: nan - accuracy: 0.3304 - val_loss: nan - val_accuracy: 0.3333
Epoch 4/17
60/60 [==============================] - 35s 590ms/step - loss: nan - accuracy: 0.3304 - val_loss: nan - val_accuracy: 0.3333
Epoch 5/17
60/60 [==============================] - 34s 567ms/step - loss: nan - accuracy: 0.3304 - val_loss: nan - val_accuracy: 0.3333
Epoch 6/17
60/60 [==============================] - 35s 588ms/step - loss: nan - accuracy: 0.3304 - val_loss: nan - val_accuracy: 0.3333
Epoch 7/17
60/60 [==============================] - 36s 604ms/step - loss: nan - accuracy: 0.3304 - val_loss: nan - val_accuracy: 0.3333
Epoch 8/17
60/60 [==============================] - 34s 569ms/step - loss: nan - accuracy: 0.3304 - val_loss: nan - val_accuracy: 0.3333
Epoch 9/17
60/60 [==============================] - 35s 594ms/step - loss: nan - accuracy: 0.3304 - val_loss: nan - val_accuracy: 0.3333
Epoch 10/17
60/60 [==============================] - 36s 608ms/step - loss: nan - accuracy: 0.3304 - val_loss: nan - val_accuracy: 0.3333
Epoch 11/17
60/60 [==============================] - 34s 565ms/step - loss: nan - accuracy: 0.3304 - val_loss: nan - val_accuracy: 0.3333
Epoch 12/17
60/60 [==============================] - 35s 583ms/step - loss: nan - accuracy: 0.3304 - val_loss: nan - val_accuracy: 0.3333
Epoch 13/17
60/60 [==============================] - 35s 577ms/step - loss: nan - accuracy: 0.3304 - val_loss: nan - val_accuracy: 0.3333
Epoch 14/17
60/60 [==============================] - 36s 597ms/step - loss: nan - accuracy: 0.3304 - val_loss: nan - val_accuracy: 0.3333
Epoch 15/17
60/60 [==============================] - 34s 570ms/step - loss: nan - accuracy: 0.3304 - val_loss: nan - val_accuracy: 0.3333
Epoch 16/17
60/60 [==============================] - 35s 582ms/step - loss: nan - accuracy: 0.3304 - val_loss: nan - val_accuracy: 0.3333
Epoch 17/17
60/60 [==============================] - 36s 601ms/step - loss: nan - accuracy: 0.3304 - val_loss: nan - val_accuracy: 0.3333
```

Gambar 4.24 Hasil Modifikasi *Epoch* 17 Aktivasi ReLu dan *Optimizer* Adam

Hasil yang didapatkan pada *epoch* 17 memperoleh akurasi 0.3304% dengan akurasi validasi 0.3333%.



Gambar 4.25 Grafik Hasil Modifikasi *Epoch* 17 Aktivasi ReLu dan *Optimizer* Adam

4.9 Evaluasi Model VGG-19

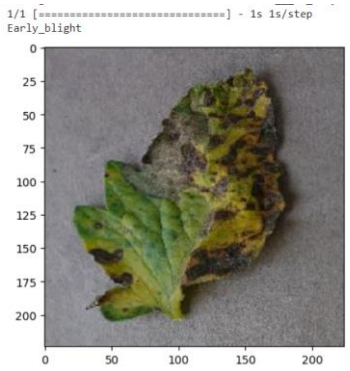
```
17/17 [==============================] - 3s 197ms/step
Accuracy: 91.85%
Confusion Matrix:
[[174   3   3]
 [  6 163  11]
 [  1  20 159]]
Classification Report:
                precision    recall  f1-score   support

Bacterial_spot       0.96      0.97      0.96       180
  Early_blight       0.88      0.91      0.89       180
   Late_blight       0.92      0.88      0.90       180

      accuracy                           0.92       540
     macro avg       0.92      0.92      0.92       540
  weighted avg       0.92      0.92      0.92       540
```

Gambar 4.26 Evaluasi Model VGG-19

Gambar 4.26 merupakan hasil evaluasi dari percobaan dengan akurasi terbaik, dengan memodifikasi 18 *epoch* dengan menggunakan aktivasi *Softmax* dan *Optimizer* Adam. Terdapat 540 jumlah total gambar yang ditemukan dalam direktori validasi, yang termasuk ke dalam 3 kelas. Pada matriks ini, setiap baris mewakili kelas yang sebenarnya dari data, sedangkan setiap kolom mewakili kelas yang diprediksi oleh model. Angka di dalam matriks menunjukkan jumlah sampel yang ditempatkan dalam setiap kelompok kelas "*Bacterial_spot*" terdapat 180 gambar yang diprediksi dengan benar sebagai "*Bacterial_spot*". Kelas "*Early_blight*" terdapat 180 gambar yang diprediksi dengan benar sebagai "*Early_Blight*" tanpa ada prediksi yang salah. Kelas "*Late_blight*" terdapat 180 gambar yang diprediksi dengan benar sebagai "*Late_blight*" tanpa ada prediksi yang salah.

4.10 Prediksi Kelas Gambar dengan Menggunakan VGG-19

```
from google.colab import files
uploaded = files.upload()

for fn in uploaded.keys():
  # predict images
  path = '/content/' + fn
  img = image.load_img(path, target_size=(224,224))
  x = image.img_to_array(img)
  x = np.expand_dims(x, axis =0)

  preds = modelvgg19.predict(x)
  preds=np.argmax(preds, axis=1)
  if preds==0:
        preds="Bacterial_spot"
  elif preds==1:
        preds="Early_blight"
  else:
        preds="Late_blight"
  print(preds)
  plt.imshow(img)
```



Gambar 4.27 Deteksi dengan Gambar

Gambar 4.27 merupakan implementasi untuk memprediksi kelas suatu gambar menggunakan model VGG-19 yang telah dilatih sebelumnya. Pertama-tama, gambar yang akan diprediksi kelasnya dimuat dan diproses sesuai dengan ukuran yang diharapkan oleh model. Kemudian, prediksi dilakukan dengan menggunakan model VGG-19 yang telah dilatih sebelumnya. Hasil prediksi berupa label kelas yang diprediksi dan probabilitasnya. Label kelas dan probabilitas ini kemudian ditampilkan di atas gambar sebagai judul. Hasil akhirnya adalah gambar yang ditampilkan beserta dengan prediksi kelasnya dan probabilitasnya.

Table 4.1 Table Perbandingan

Aktivasi	Optimizer	Epoch	Accuracy
Softmax	Adam	18	91,85%
Softmax	SGD	17	82,59%
Softmax	Adagrad	17	89,26%
Softmax	RMSProp	17	89,07%
ReLu	Adam	17	33,33%
ReLu	SGD	17	33,33%
ReLu	Adagrad	17	33,33%

ReLu	RMSProp	17	33,33%

Tabel 4.1 menunjukkan peningkatan performa yang signifikan pada model yang dikembangkan, dengan peningkatan akurasi pada setiap aspek yang dievaluasi. Dengan menggunakan penggunaan aktivasi Softmax, penggunaan optimizer Adam dan bertambahnya epoch menjadi 18 model baru berhasil mencapai akurasi tertinggi sebesar 91,85%.

4.11 Perangkat Lunak

Spesifikasi perangkat lunak yang digunakan untuk mengembangkan model ini adalah sebagai berikut :

1. Sistem operasi windows 10
2. Google Colab
3. Bahasa pemograman pyhton
4. Google Chrome
5. Koneksi Internet

4.12 Spesifikasi Perangkat Keras

Spesifikasi perangkat keras yang digunakan untuk mengembangkan model ini adalah sebagai berikut :

1. Intel Core i3-1115GB
2. RAM DDR4 4GB
3. SSD 512GB