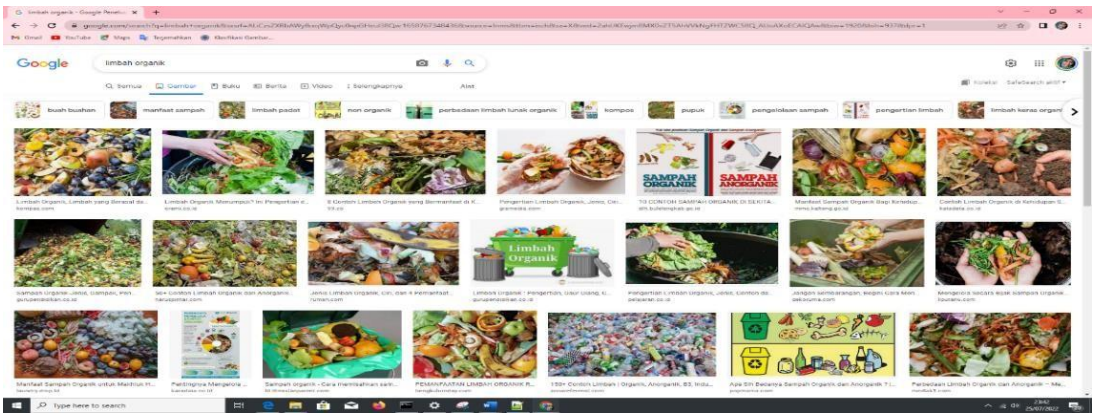
BAB IV
HASIL DAN PEMBAHASAN

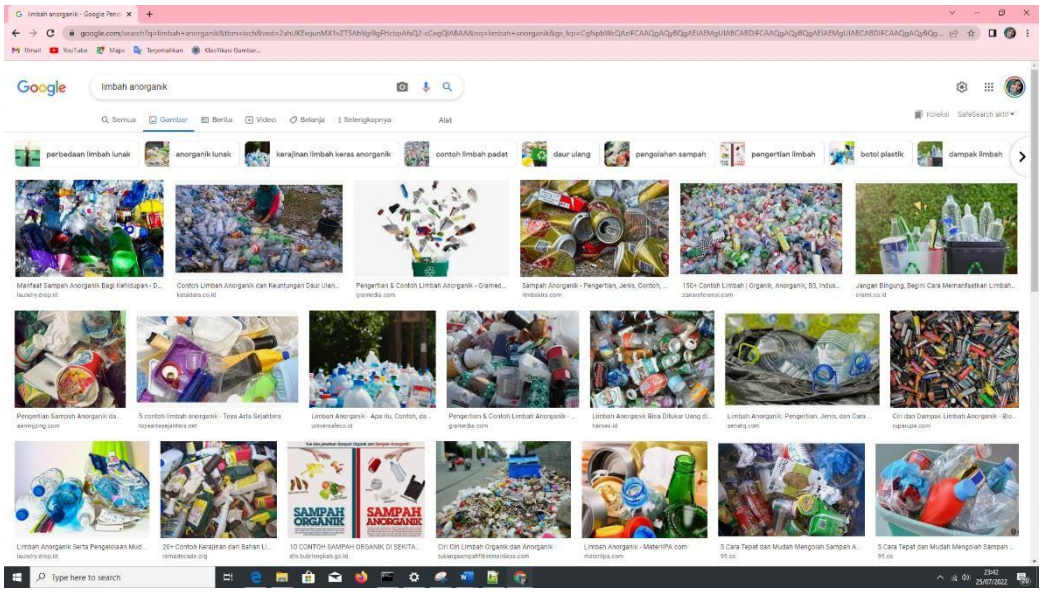
Pada ini, peneliti melakukan klasifikasi gambar limbah organic dan non-organik dengan menggunakan Convolutional Neural Network (CNN). Proses utama dalam pembuatan model klasifikasi metode CNN adalah perancangan klasifikasi. Prinsip dasar dari proses ini adalah dengan melakukan pelatihan pada CNN untuk menghasilkan model yang paling bagus dalam hal akurasi yang baik. Dalam pelatihan CNN terdapat data in dan data test atau validation, data in digunakan untuk melakukan training pada model dan data test digunakan untuk validasi dari data ining. Akurasi model tersebut dapat ditentukan dengan melakukan validasi dengan menggunakan data uji. Pelatihan model menggunakan library keras pada nd prompt dengan back-end Tensorflow. Google mengembangkan Keras sebagai pembungkus Tensorflow untuk menerapkan dan memberikan bagian dari folder contrib dalam kode Tensorflow. Keras dirancang untuk menghilangkan kode boilerplate. Beberapa ris kode keras akan menghasilkan lebih banyak daripada kode Tensorflow asli, sehingga pat dengan mudah merancang CNN dan RNN serta dapat menjalankannya di GPU atau CPU.

4.1 Pembuatan Dataset

Peneliti mengumpulkan dan membuat dataset berupa gambar yang di dapat dari google image. Deep learning terutama CNN biasanya n banyak data mbar. Pengambilan gambar dilakukan dengan Teknik crawling dari google dan disimpan pada google drive.



Gambar 4.1 Google Image Search limbah organic



Gambar 4.2 Google Image Search limbah organic

Setelah gambar disimpan pada google drive, selanjutnya penulis menghubungkan google colab notebook dengan google drive.

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

Gambar 4.3 Import Drive

setelah membuat dataset dan menghubungkannya pada google drive, peneliti akan melakukan import library yang diantaranya adalah numpy, matplotlib, tensorflow, keras dan library yang dibutuhkan untuk klasifikasi limbah non- organik.

```
[2] import numpy as np
    import matplotlib.pyplot as plt
    import seaborn as sns
    %matplotlib inline

    import os
    os.environ["TF_CPP_MIN_LOG_LEVEL"]="2"
    import warnings
    warnings.filterwarnings("ignore")

    from sklearn.metrics import confusion_matrix,classification_report

    import tensorflow as tf
    from tensorflow import keras
    from tensorflow.keras.models import Model
    from tensorflow.keras.layers import Input, Dense, GlobalAveragePooling2D, Dropout
    from tensorflow.keras.optimizers import Adam
    from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau
    from tensorflow.keras.losses import BinaryCrossentropy
    from tensorflow.keras.preprocessing.image import ImageDataGenerator
    from tensorflow.keras.applications import VGG16, ResNet50

[ ] !pip install tensorflow==2.7.0
```

Gambar 4.4 Import Library

4.2 Preprocessing Citra

Setelah mengumpulkan dataset, selanjutnya dilakukan preprocessing citra dengan beberapa tahapan seperti berikut:

1. Resize

   Ukuran citra isanya in oleh simbol h = height dan w = widtht dalam dua dimensi (h x w). Pada n-peneltian sebelumnya h banyak yang menggunakan ukuran 28x28 atau 32x32, namun padapenelitian ini penulis ingin membuat program dapat bekerjan citraukuran 256x256.

```
[4] BATCH_SIZE = 32
    IMG_WIDTH = 256
    IMG_HEIGHT = 256
    IMG_CHANNELS = 3
    IMG_SHAPE = (IMG_WIDTH, IMG_HEIGHT, IMG_CHANNELS)

    print(f"Batch Size: {BATCH_SIZE}")
    print(f"Image Shape: {IMG_SHAPE}")

    Batch Size: 32
    Image Shape: (256, 256, 3)
```

Gambar 4.5 Resize

2. Pelabelan Gambar

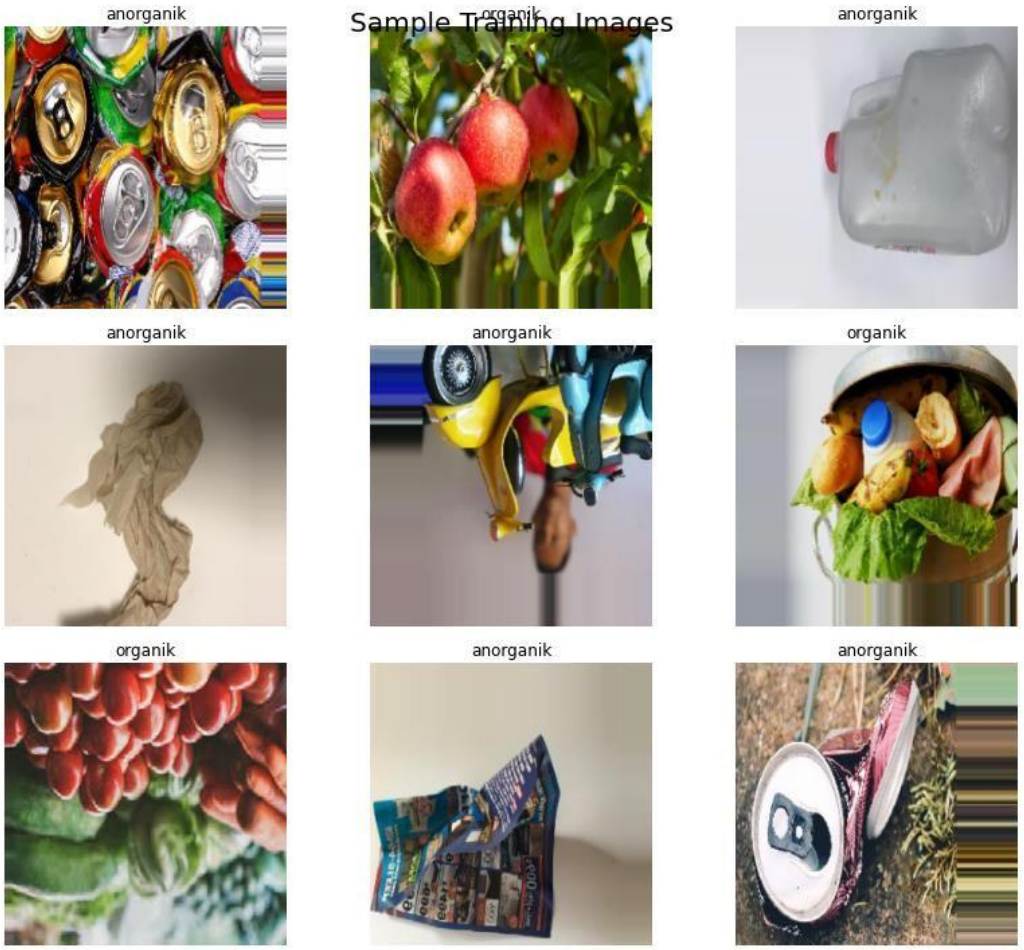
Pelabelan gambar adalah tahap awal dimana dataset input diberikan label atau pengenal (tanda) dengan tujuan untuk menyimpan informasi gambar.

```
[8] labels = {value: key for key, value in train_generator.class_indices.items()}

    print("Label Mappings for classes present in the training and validation datasets\n")
    for key, value in labels.items():
        print(f"{key} : {value}")
```

```
Label Mappings for classes present in the training and validation datasets

0 : anorganik
1 : organik
```

Gambar 4.6 Pelabelan Gambar

Setelah melakukan rezise dan memberi label pada gambar, peneliti menampilkan contoh gambar yang akan diklasifikasikan sebagai berikut :

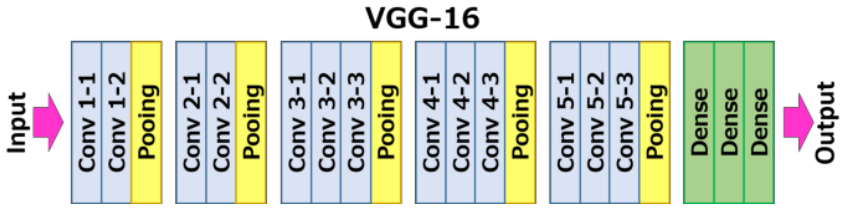


Gambar 4.7 Contoh Gambar

4.3 Setting Arsitektur CNN

VGG 16

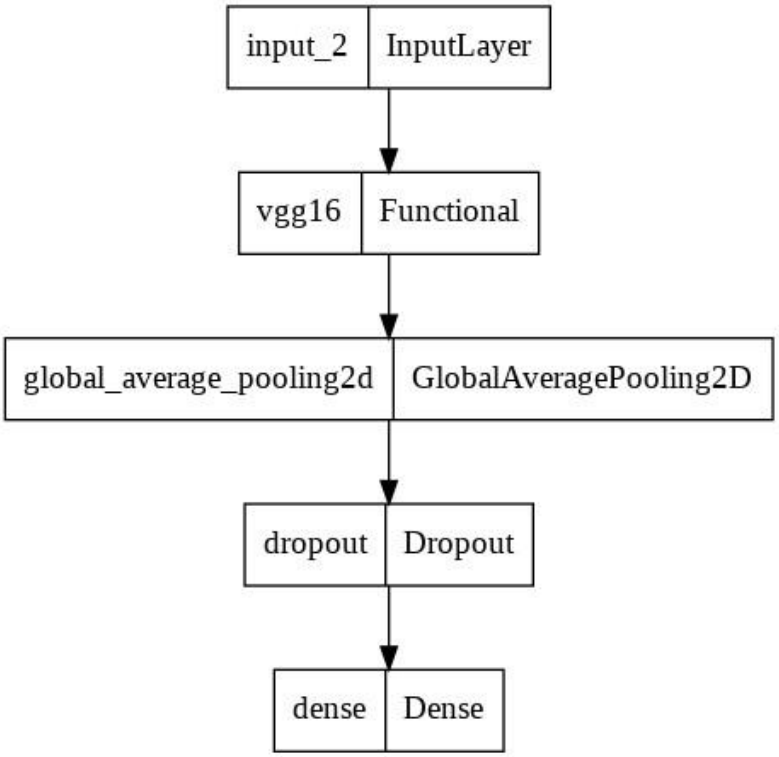
VGG16 adalah Arsitektur Jaringan Saraf Konvolusi (CNN) sederhana n banyak digunakan yang digunakan untuk ImageNet, proyek basis data l besar yang digunakan dalam penelitian perangkat lunak pengenalan objek visual. Arsitektur VGG16 dikembangkan dan diperkenalkan oleh Karen Simonyan dan Andrew Zisserman dari Universitas Oxford, pada hun 2014, melalui artikel mereka "Jaringan Konvolusi Sangat Dalam untuk Pengenalan Gambar Skala Besar." 'VGG' adalah singkatan dari Visual Geometry Group, yang merupakan sekelompok peneliti di Universitas Oxford yang mengembangkan arsitektur ini, dan '16' menyiratkan bahwa rsitektur ini memiliki 16 lapisan.



Gambar 4.8 Arsitektur VGG16

VGG16 memiliki input layer yang berukuran 256x256. VGG16 terdapat 16 layer convolution dengan kernel sebesar 3x3, stride sebesar 1, dan ReLu untuk setiap hidden layer. Convolution layer akan diikuti dengan 5 Max Pooling layer. Max Pooling layer yang digunakan memiliki kernel sebesar 2x2 dan stride sebesar 2. Kemudian akan diikuti dengan 3 Fully Connected layer yang 2 diantaranya memiliki 4096 channels dan layer terakhir menggunakan sigmoid activation untuk proses klasifikasi dengan jumlah channels yang sesuai dengan jumlah kelas. Modifikasi VGG16 menerapkan konsep transfer learning. Model modifikasi VGG16 mengambil pre-trained weights dari model yang telah di-train sebelumnya. Kemudian model dari VGG16 ini akan dimodifikasi pada bagian fully connected layer. Modifikasi ini meliputi menambahkan Average Pooling La yer dan dropout.

Begini penampakan modelnya. Peneliti telah menambahkan lapisan Global Average Pooling setelah Model VGG untuk meratakan output dan kemudian Lapisan Dropout dan Padat untuk memprediksi kelas.



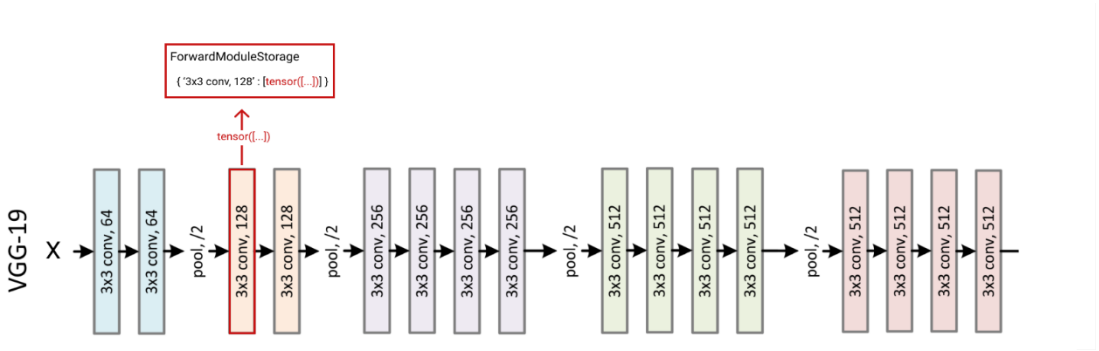
Gambar 4.9 Model Arsitektur VGG16

```
[11] Model: "vgg16"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 256, 256, 3)]     0

 block1_conv1 (Conv2D)       (None, 256, 256, 64)      1792

 block1_conv2 (Conv2D)       (None, 256, 256, 64)      36928

 block1_pool (MaxPooling2D)  (None, 128, 128, 64)      0

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

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

 block2_pool (MaxPooling2D)  (None, 64, 64, 128)       0

 block3_conv1 (Conv2D)       (None, 64, 64, 256)       295168

 block3_conv2 (Conv2D)       (None, 64, 64, 256)       590080

 block3_conv3 (Conv2D)       (None, 64, 64, 256)       590080

 block3_pool (MaxPooling2D)  (None, 32, 32, 256)       0

 block4_conv1 (Conv2D)       (None, 32, 32, 512)       1180160

 block4_conv2 (Conv2D)       (None, 32, 32, 512)       2359808

 block4_conv3 (Conv2D)       (None, 32, 32, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 16, 16, 512)       0

 block5_conv1 (Conv2D)       (None, 16, 16, 512)       2359808

 block5_conv2 (Conv2D)       (None, 16, 16, 512)       2359808

 block5_conv3 (Conv2D)       (None, 16, 16, 512)       2359808

 block5_pool (MaxPooling2D)  (None, 8, 8, 512)         0

=================================================================
Total params: 14,714,688
Trainable params: 14,714,688
Non-trainable params: 0
```

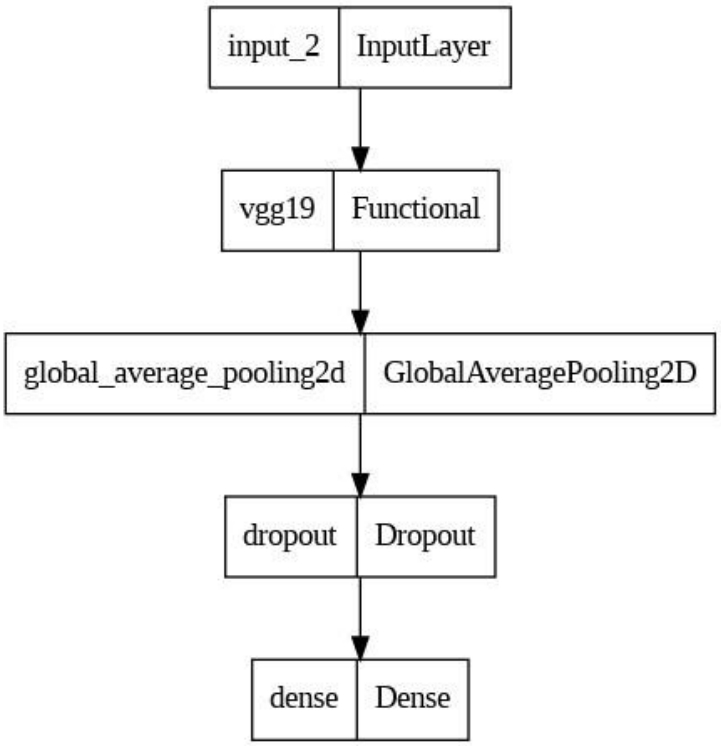
Gambar 4.10 Model CNN VGG16

VGG19

Konsep model VGG19 (jugaVGGNet-19) samaın VGG16 kecuali mendukung 19 lapisan. "16" dan "19" adalah singkatan dari jumlah lapisan bobot dalam model (lapisan konvolusional). Ini berarti VGG19 memiliki tiga lapisan konvolusi lebih banyak daripada VGG16.



Gambar 4.11 Arsitektur VGG19



Gambar 4.12 Model Arsitektur VGG19

```
Model: "vgg19"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 256, 256, 3)]     0

 block1_conv1 (Conv2D)       (None, 256, 256, 64)      1792

 block1_conv2 (Conv2D)       (None, 256, 256, 64)      36928

 block1_pool (MaxPooling2D)  (None, 128, 128, 64)      0

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

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

 block2_pool (MaxPooling2D)  (None, 64, 64, 128)       0

 block3_conv1 (Conv2D)       (None, 64, 64, 256)       295168

 block3_conv2 (Conv2D)       (None, 64, 64, 256)       590080

 block3_conv3 (Conv2D)       (None, 64, 64, 256)       590080

 block3_conv4 (Conv2D)       (None, 64, 64, 256)       590080

 block3_pool (MaxPooling2D)  (None, 32, 32, 256)       0

 block4_conv1 (Conv2D)       (None, 32, 32, 512)       1180160

 block4_conv2 (Conv2D)       (None, 32, 32, 512)       2359808

 block4_conv3 (Conv2D)       (None, 32, 32, 512)       2359808

 block4_conv4 (Conv2D)       (None, 32, 32, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 16, 16, 512)       0

 block5_conv1 (Conv2D)       (None, 16, 16, 512)       2359808

 block5_conv2 (Conv2D)       (None, 16, 16, 512)       2359808

 block5_conv3 (Conv2D)       (None, 16, 16, 512)       2359808

 block5_conv4 (Conv2D)       (None, 16, 16, 512)       2359808

 block5_pool (MaxPooling2D)  (None, 8, 8, 512)         0

=================================================================
Total params: 20,024,384
Trainable params: 20,024,384
Non-trainable params: 0
_________________________________________________________________
```

Gambar 4.13 Model CNN VGG19

4.4 Rancangan Pengujian

Pengujian ini dilakukan ntuk melakukan evaluasi terhadap model yang dihasilkan oleh CNN. Pengujian ini dilakukan dua tahap yaitu tahapan training dan testing. Tahap training adalah tahap dimana model CNN diuji dengan data latih yang sudah disediakan. Jumlah data tih yang disediakan sebanyak 160 data mbar, dengan jumlah gambar perkelas sebanyak 80 gambar. Data training di bagi kembali menjadi dua yaitu training dan validasi, yaitu sebanyak 160 training dan 40 validasi. Tahap Testing adalah tahap pengujian model yang sudah dilakukan tahap pelatihan. Jumlah data tih dalam penelitian ini sebanyak 160 data mbar, dengan jumlah gambar perkelas sebanyak 80 gambar. Pada hap ini model di uji dengan gambar yang berbeda n tujuan menguji apakah model sudah menghasilkan performa ng baik dalam mengklasifikasikan sebuah gambar.

Perangkat Pengujian

Pengujian dilakukan pada laptop dengan spesifikasi sebagai berikut :

1. Intel core i7-6700HQ
2. 8 GB RAM
3. Sistem operasi Windows
4. Google collab notebook

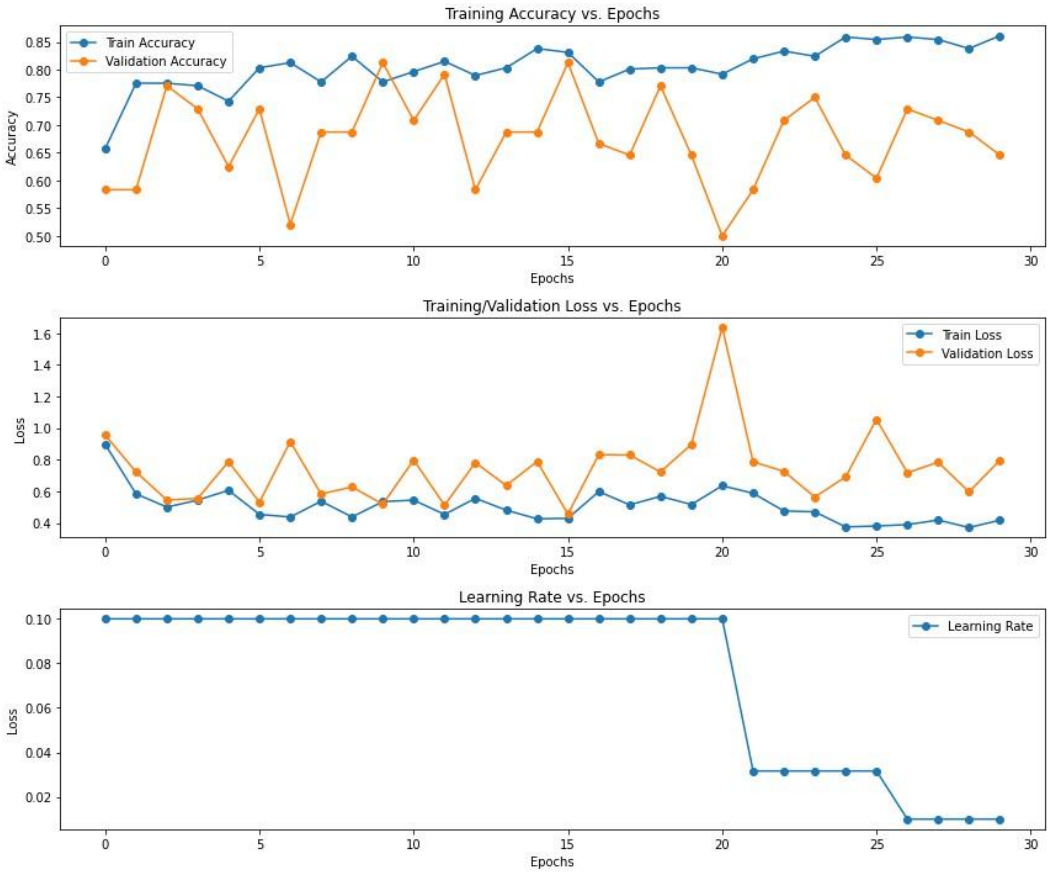
4.5 Model Hasil Training

VGG 16

Setelah melalu beberapa proses dalam algoritma l Neural Network (CNN) didapatkan hasil training dan validation. Proses ini menggunakan jumlah 30 epoch.

```
Epoch 9/30
14/14 [==============================] - 15s 1s/step - loss: 0.4377 - accuracy: 0.8241 - val_loss: 0.6277 - val_accuracy: 0.6875 - lr: 0.1000
Epoch 10/30
14/14 [==============================] - 13s 928ms/step - loss: 0.5347 - accuracy: 0.7778 - val_loss: 0.5182 - val_accuracy: 0.8125 - lr: 0.1000
Epoch 11/30
14/14 [==============================] - 13s 972ms/step - loss: 0.5441 - accuracy: 0.7963 - val_loss: 0.7978 - val_accuracy: 0.7083 - lr: 0.1000
Epoch 12/30
14/14 [==============================] - 14s 976ms/step - loss: 0.4537 - accuracy: 0.8148 - val_loss: 0.5119 - val_accuracy: 0.7917 - lr: 0.1000
Epoch 13/30
14/14 [==============================] - 13s 909ms/step - loss: 0.5556 - accuracy: 0.7894 - val_loss: 0.7816 - val_accuracy: 0.5833 - lr: 0.1000
Epoch 14/30
14/14 [==============================] - 14s 977ms/step - loss: 0.4806 - accuracy: 0.8032 - val_loss: 0.6368 - val_accuracy: 0.6875 - lr: 0.1000
Epoch 15/30
14/14 [==============================] - 14s 928ms/step - loss: 0.4257 - accuracy: 0.8380 - val_loss: 0.7888 - val_accuracy: 0.6875 - lr: 0.1000
Epoch 16/30
14/14 [==============================] - 13s 921ms/step - loss: 0.4292 - accuracy: 0.8310 - val_loss: 0.4557 - val_accuracy: 0.8125 - lr: 0.1000
Epoch 17/30
14/14 [==============================] - 13s 970ms/step - loss: 0.5985 - accuracy: 0.7778 - val_loss: 0.8321 - val_accuracy: 0.6667 - lr: 0.1000
Epoch 18/30
14/14 [==============================] - 14s 989ms/step - loss: 0.5143 - accuracy: 0.8009 - val_loss: 0.8296 - val_accuracy: 0.6458 - lr: 0.1000
Epoch 19/30
14/14 [==============================] - 13s 968ms/step - loss: 0.5683 - accuracy: 0.8032 - val_loss: 0.7225 - val_accuracy: 0.7708 - lr: 0.1000
Epoch 20/30
14/14 [==============================] - 12s 860ms/step - loss: 0.5164 - accuracy: 0.8032 - val_loss: 0.8967 - val_accuracy: 0.6458 - lr: 0.1000
Epoch 21/30
14/14 [==============================] - 13s 910ms/step - loss: 0.6351 - accuracy: 0.7917 - val_loss: 1.6376 - val_accuracy: 0.5000 - lr: 0.1000
Epoch 22/30
14/14 [==============================] - 14s 986ms/step - loss: 0.5876 - accuracy: 0.8194 - val_loss: 0.7865 - val_accuracy: 0.5833 - lr: 0.0316
Epoch 23/30
14/14 [==============================] - 15s 1s/step - loss: 0.4757 - accuracy: 0.8333 - val_loss: 0.7264 - val_accuracy: 0.7083 - lr: 0.0316
Epoch 24/30
14/14 [==============================] - 12s 868ms/step - loss: 0.4701 - accuracy: 0.8241 - val_loss: 0.5639 - val_accuracy: 0.7500 - lr: 0.0316
Epoch 25/30
14/14 [==============================] - 14s 985ms/step - loss: 0.3748 - accuracy: 0.8588 - val_loss: 0.6910 - val_accuracy: 0.6458 - lr: 0.0316
Epoch 26/30
14/14 [==============================] - 14s 964ms/step - loss: 0.3802 - accuracy: 0.8542 - val_loss: 1.0555 - val_accuracy: 0.6042 - lr: 0.0316
Epoch 27/30
14/14 [==============================] - 12s 844ms/step - loss: 0.3892 - accuracy: 0.8588 - val_loss: 0.7173 - val_accuracy: 0.7292 - lr: 0.0100
Epoch 28/30
14/14 [==============================] - 14s 951ms/step - loss: 0.4181 - accuracy: 0.8542 - val_loss: 0.7840 - val_accuracy: 0.7083 - lr: 0.0100
Epoch 29/30
14/14 [==============================] - 14s 991ms/step - loss: 0.3708 - accuracy: 0.8380 - val_loss: 0.5964 - val_accuracy: 0.6875 - lr: 0.0100
Epoch 30/30
14/14 [==============================] - 13s 926ms/step - loss: 0.4175 - accuracy: 0.8611 - val_loss: 0.7960 - val_accuracy: 0.6458 - lr: 0.0100
```

Gambar 4.14 Proses Training VGG16



Gambar 4.15 Grafik Hasil Training VGG16

Berdasarkan gambar accuracy dari training model mencapai 0.8472 dengan nilai loss sebesar 0.3399. Proses training disini menggunakan learning rate 0.1 dengan input gambar sebesar 256 x 256 piksel. Waktu pelatihan yang dibutuhkan untuk 30 epoch dalam menjalankan training model ini yaitu 2 menit. Semakin Banyak epoch maka makin lama waktu yang dibutuhkan untuk training model.

```
4/4 [==============================] - 2s 298ms/step - loss: 0.5529 - accuracy: 0.7667
Test Loss:     0.552945077419281
Test Accuracy: 0.7666666507720947
```

Gambar 4.16 Proses Testing VGG16

Berdasarkan gambar accuracy dari proses testing mencapai 0.7666 dengan nilai loss sebesar 0.5529.

Gambar 4.17 Hasil Prediksi

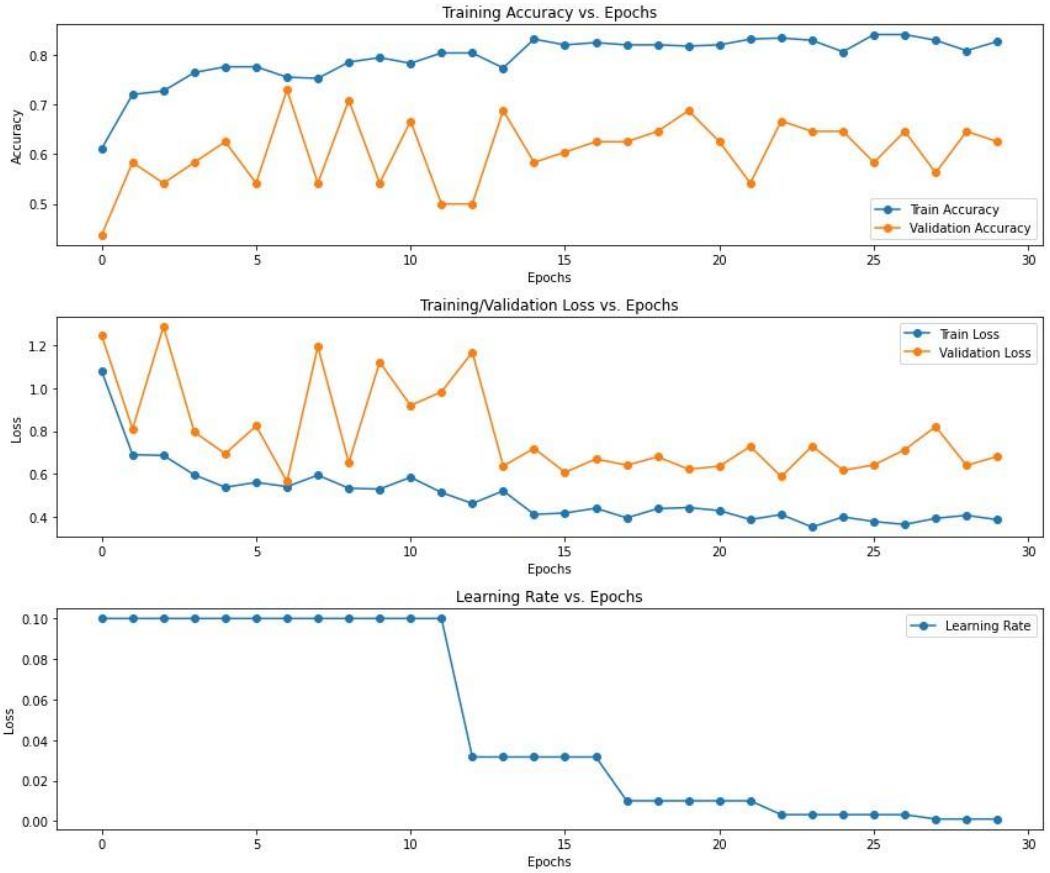
diatas merupakan test prediksi pada ta mbar yang ada, dengan melihat benar atau tidaknya label yang ada pada gambar tersebut maka peneliti dapat menentukan berapa persen akurasi yang ada pada program yang dibuat.

VGG 19

Setelah melalu beberapa proses dalam algoritma Convolutional Neural Network (CNN) didapatkan hasil training dan validation. Proses ini menggunakan jumlah 30 epoch.

```
Epoch 9/30
14/14 [==============================] - 14s 960ms/step - loss: 0.5344 - accuracy: 0.7847 - val_loss: 0.6522 - val_accuracy: 0.7083 - lr: 0.1000
Epoch 10/30
14/14 [==============================] - 13s 966ms/step - loss: 0.5306 - accuracy: 0.7940 - val_loss: 1.1205 - val_accuracy: 0.5417 - lr: 0.1000
Epoch 11/30
14/14 [==============================] - 12s 856ms/step - loss: 0.5855 - accuracy: 0.7824 - val_loss: 0.9189 - val_accuracy: 0.6667 - lr: 0.1000
Epoch 12/30
14/14 [==============================] - 14s 966ms/step - loss: 0.5149 - accuracy: 0.8032 - val_loss: 0.9820 - val_accuracy: 0.5000 - lr: 0.1000
Epoch 13/30
14/14 [==============================] - 14s 1s/step - loss: 0.4636 - accuracy: 0.8032 - val_loss: 1.1663 - val_accuracy: 0.5000 - lr: 0.0316
Epoch 14/30
14/14 [==============================] - 13s 954ms/step - loss: 0.5222 - accuracy: 0.7731 - val_loss: 0.6379 - val_accuracy: 0.6875 - lr: 0.0316
Epoch 15/30
14/14 [==============================] - 13s 893ms/step - loss: 0.4133 - accuracy: 0.8310 - val_loss: 0.7197 - val_accuracy: 0.5833 - lr: 0.0316
Epoch 16/30
14/14 [==============================] - 14s 988ms/step - loss: 0.4192 - accuracy: 0.8194 - val_loss: 0.6083 - val_accuracy: 0.6042 - lr: 0.0316
Epoch 17/30
14/14 [==============================] - 13s 963ms/step - loss: 0.4415 - accuracy: 0.8241 - val_loss: 0.6705 - val_accuracy: 0.6250 - lr: 0.0316
Epoch 18/30
14/14 [==============================] - 13s 922ms/step - loss: 0.3969 - accuracy: 0.8194 - val_loss: 0.6414 - val_accuracy: 0.6250 - lr: 0.0100
Epoch 19/30
14/14 [==============================] - 12s 801ms/step - loss: 0.4398 - accuracy: 0.8194 - val_loss: 0.6804 - val_accuracy: 0.6458 - lr: 0.0100
Epoch 20/30
14/14 [==============================] - 13s 909ms/step - loss: 0.4447 - accuracy: 0.8171 - val_loss: 0.6228 - val_accuracy: 0.6875 - lr: 0.0100
Epoch 21/30
14/14 [==============================] - 13s 900ms/step - loss: 0.4305 - accuracy: 0.8194 - val_loss: 0.6366 - val_accuracy: 0.6250 - lr: 0.0100
Epoch 22/30
14/14 [==============================] - 13s 964ms/step - loss: 0.3893 - accuracy: 0.8310 - val_loss: 0.7287 - val_accuracy: 0.5417 - lr: 0.0100
Epoch 23/30
14/14 [==============================] - 13s 955ms/step - loss: 0.4122 - accuracy: 0.8333 - val_loss: 0.5883 - val_accuracy: 0.6667 - lr: 0.0032
Epoch 24/30
14/14 [==============================] - 13s 926ms/step - loss: 0.3542 - accuracy: 0.8287 - val_loss: 0.7301 - val_accuracy: 0.6458 - lr: 0.0032
Epoch 25/30
14/14 [==============================] - 13s 952ms/step - loss: 0.4013 - accuracy: 0.8056 - val_loss: 0.6179 - val_accuracy: 0.6458 - lr: 0.0032
Epoch 26/30
14/14 [==============================] - 13s 969ms/step - loss: 0.3797 - accuracy: 0.8403 - val_loss: 0.6426 - val_accuracy: 0.5833 - lr: 0.0032
Epoch 27/30
14/14 [==============================] - 13s 920ms/step - loss: 0.3662 - accuracy: 0.8403 - val_loss: 0.7135 - val_accuracy: 0.6458 - lr: 0.0032
Epoch 28/30
14/14 [==============================] - 14s 987ms/step - loss: 0.3946 - accuracy: 0.8287 - val_loss: 0.8207 - val_accuracy: 0.5625 - lr: 0.0010
Epoch 29/30
14/14 [==============================] - 12s 857ms/step - loss: 0.4087 - accuracy: 0.8079 - val_loss: 0.6405 - val_accuracy: 0.6458 - lr: 0.0010
Epoch 30/30
14/14 [==============================] - 13s 954ms/step - loss: 0.3881 - accuracy: 0.8264 - val_loss: 0.6827 - val_accuracy: 0.6250 - lr: 0.0010
```

Gambar 4.19 Proses Training VGG19



Gambar 4.20 Grafik Hasil Training VGG19

Berdasarkan gambar accuracy dari training model mencapai 0.8264 dengan nilai loss sebesar 0.3865. Proses training disini menggunakan learning rate 0.1 dengan input gambar sebesar 256 x 256 piksel. Waktu pelatihan yang dibutuhkan untuk 30 epoch dalam menjalankan training model ini yaitu 2 menit. Semakin Banyak epoch maka akin lama waktu yang dibutuhkan untuk training model.

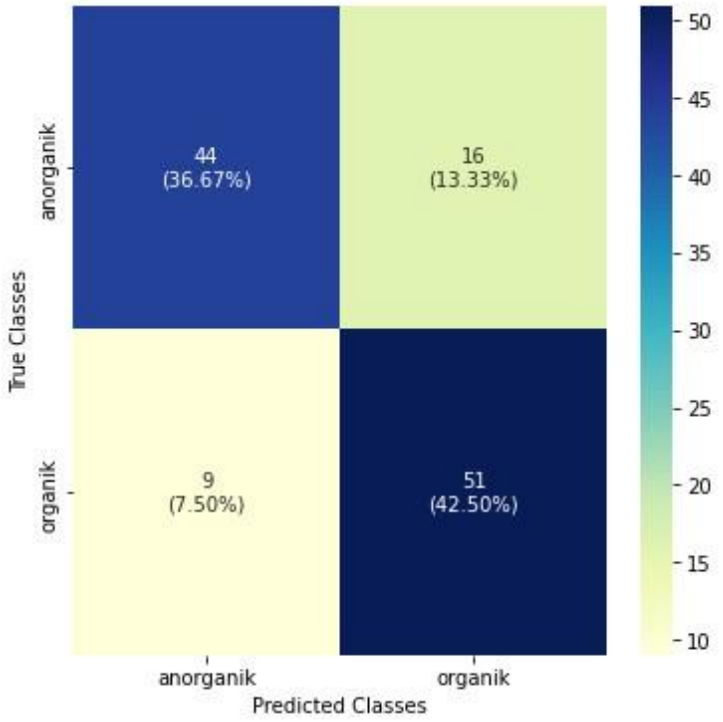
```
4/4 [==============================] - 1s 236ms/step - loss: 0.4542 - accuracy: 0.7917
Test Loss:     0.4541679322719574
Test Accuracy: 0.7916666865348816
```

Gambar 4.21 Proses Testing VGG19

Berdasarkan gambar accuracy dari proses testing mencapai 0.7916 dengan nilai loss sebesar 0.4541.

4.6 Confusion Matrix

Confusion matrix adalah representasi dari nilai prediksi model classifier dan nilai riil dalam struktur seperti tabel di mana baris mewakili nilai prediksi dan semua kolom mewakili nilai riil, semua nilai yang disajikan dalam tabel adalah jumlah Input yang diberikan ke model untuk klasifikasi.



Gambar 4.22 Confusion Matrix

Tabel di atas adalah Confusion Matrix yang digunakan untuk memprediksi anorganik vs organic.

Dalam contoh anorganik vs organik peneliti telah merancang dan melatih model untuk memprediksi dua hal yaitu anorganik atau organik. Untuk membuat confusion matrix kita perlu mengambil satu item pada satu waktu. Dalam kasus kita ambil anjing pertama kalikan membuat empat nilai yaitu TP,TN,FP,FN.

Ada empat terminologi yang digunakan dalam matriks konfusi yaitu True Positive(TP), True Negative(TN),False positive(FP),False Negative(TN),

- True positive(TP): Semua nilai yang diprediksi oleh model cocok dengan nilai sebenarnya rti model memprediksinya n benar. Seperti dalam kasus di atas adalah 20. Di mana norganik diprediksi sebagai anorganik.
- True Negative(TN): Semua nilai yang diprediksi oleh model bukan anorganik dan nilai sebenarnya juga bukan anorganik yang organik seperti dalam kasus di atas adalah 20.
- False positive(FP): Semua nilai yang diprediksi oleh model bukan anorganik tetapi nilai sebenarnya dalah anorganik yang dalam kasus ini adalah 0.
- False Negative(TN): Semua nilai yang diprediksi oleh model adalah anorganik tetapi kenyataannya adalah organik yang dalam kasus ini adalah 0.

Precision : TP / TP+FP

Precision menunjukkan seberapa tepat model pada sarana input.

Anorganik : 44 / 44+16 = 0.83

Organik : 51 / 51+9 = 0.76

Recall : TP / TP+FN

Recall menunjukkan berapa kali model benar dalam semua nilai nyata.

Anorganik : 44 / 44+9 = 0.73

Organik : 51 / 51+16 = 0.85

f1 : 2*Precision*Recall / Precision+Recall

f1 memberikan kombinasi Precision dan Recall. itu memberi bobot lebih pada Precision.

Anorganik : 2*0.83*0.73 / 0.83+0.73 = 0.78

Organik : 2*0.76*0.85 / 0.76+0.85 = 0.80

Accuracy : TP+TN / TP+TN+FP+FN

44+51 / 44+51+16+9 = 0.79

4.7 Pengaruh Nilai Batch Size

Penelitian ini juga lakukan uji coba n menggunakan nilai batch size yang berbeda. Dalam klasifikasi gambar pada umumnya banyak menggunakan nilai batch size sebesar 2 sampai 32. Penentuan nilai batch size biasanyaditentukan oleh peneliti. Peneliti menggunakan lima nilai yaitu 2, 4, 8, 16, dan 32. Penentuan nilai dari batch size ini sangat berpengaruh pada performa akurasi. Hasil batch size adalah sebagai berikut :

Tabel 4.1 Nilai Batch size

Batch Size	**Accuracy**	**Loss**	**Learning Rate**
2	67%	0.6446	0.1
4	63%	0.6541	0.1
8	70%	0.6292	0.1
16	78%	0.4440	0.1
32	79%	0.4541	0.1

Berdasarkan Tabel 4.1 nilai batch size dengan accuracy paling tinggi yaitu dengan nilai batch size 32 sebesar 79% dan paling kecil yaitu 63% di batch size 4 dengan loss 0.6541. seluruh uji coba batch size menggunakan learning rate 0.1.

4.8 Pengaruh Nilai Learning Rate

Penelitian ini juga melakukan uji coba dengan menggunakan nilai learning rate yang berbeda. Dalam klasifikasi gambar pada umumnya banyak menggunakan nilai learning rate sebesar 0.1 sampai 0.0001. Penentuan nilai learning rate biasanya ditentukan oleh peneliti. Peneliti menggunakan empat nilai yaitu 0.1, 0.01, 0.001, dan 0.0001. Penentuan nilai dari learning rate ini sangat berpengaruh pada performa akurasi. Hasil learning rate adalah sebagai berikut:

Tabel 4.2 Nilai Learning Rate

Learning Rate	**Accuracy**	**Loss**	**Batch Size**
0.1	79%	0.4541	32
0.01	76%	0.4604	32
0.001	74%	0.5905	32
0.0001	66%	0.6606	32

Berdasarkan Tabel 4.2 nilai learning rate dengan accuracy tertinggi ada da nilai learning rate 0.1 yang menghasilkan 93% sedangkan nilai terendah ada pada rning rate 0.001 yaitu 79% dengan loss 0.5407. seluruh uji coba da learning rate menggunakan batch size 32.

4.9 Pengaruh Nilai Epoch

Penelitian ini juga kukan uji coba n menggunakan epoch yang berbeda. Penentuan nilai epoch biasanya n oleh peneliti. Peneliti menggunakan empat nilai epoch yaitu 30, 50, 80, dan 100. Penentuan nilai epoch ini sangat berpengaruh pada kurasi. Hasil uji cobaepoch adalah sebagai berikut :

Tabel 4.3 Pengaruh Nilai Epoch

Epoch	**Accuracy**	**Loss**	**Learning Rate**	**Batch Size**
30	79%	0.4541	0.1	32
50	74%	0.5905	0.1	32
80	74%	0.5905	0.1	32
100	70%	0.6070	0.1	32

Berdasarkan Tabel 4.3 nilai epoch dengan accuracy tertinggi yaitu ada pada nilai epoch 30 yang menghasilkan 79% dengan loss sebesar 0.4541.

4.10 Perbandingan Arsitektur VGG16 dan VGG19

Pada penelitian ini peneliti membuat perbandingan antara arsitektur VGG16 dan VGG19. Untuk VGG16 memakai layer sebanyak 16 layer, sedangkan VGG19 menggunakan layer sebanyak 19 layer. Berikut adalah hasil perbandingan yang dilakukan :

Tabel 4.4 Perbandingan Arsirtektur

Arsitektur	**Accuracy**	**f1-Score**	**Loss**	**Batch Size**	**Learning Rate**
VGG16	76%	73.00	0.5529	32	0.1
VGG19	79%	78.00	0.4541	32	0.1

Berdasarkan perbandingan di atas maka penulis menyimpulkan bahwa arsitektur VGG19 lebih baik untuk penelitian yang dilakukan dengan uji coba perbandingan accuracy sebesar 79% dengan batch size 32 dan learning rate 0.1.

4.11 Alasan Mengapa VGG19 Lebih Baik Dari VGG16

Alasan yang memungkinkan VGG19 lebih baik akurasinya ri VGG16 yaitu dikarenakan data ng ada lebih cocok digunakan menggunakan VGG19 berdasarkan hasil percobaan yang telah dilakukan. VGG19 juga merupakan arsitektur terbaru yang diluncurkan tahun 2021.