Jurnal Teknik dan Science
JTS Vol 3 No. 3 Oktober 2024 | P-ISSN: 2828-7002 E-ISSN: 2828-6871, Page 56-69

ANALISIS PERBANDINGAN HASIL KLASIFIKASI JENIS PENYAKIT TANAMAN TOMAT MENGGUNAKAN ARSITEKTUR MOBILENET, DENSENET121, DAN XCEPTION

Kuwat Setiyanto[1], Michael Bolang[2]
[1,2] Sistem Informasi, kuwatsetiyanto@gmail.com, Universitas Gunadarma

ABSTRACT

Machine learning can be applied in various needs, such as image classification. Plant disease classification is essential and significantly supports the agricultural sector in this modern era. With an application capable of classifying diseases in crops, farmers can accurately identify the diseases affecting their harvest and address them more efficiently and effectively compared to traditional methods, which can be more time-consuming. This research aims to determine the best TensorFlow architecture among the three architectures used in this study, namely MobileNet, DenseNet121, and Xception, to classify 9 types of tomato plant diseases and 1 healthy tomato plant. The study concludes that DenseNet121 is the best architecture for classifying the 9 types of tomato plant diseases and 1 healthy tomato plant. During testing, the DenseNet121 model achieved an accuracy, precision, recall, and F-1 score of approximately 0.987 or 98.7%. Xception ranked second with all four metrics scoring around 0.986 or 98.6%, while MobileNet ranked last with metrics scoring approximately 0.973 or 97.3%.

Keywords: CNN, DenseNet121, Image Classification, Machine Learning, MobileNet, TensorFlow, Transfer Learning, Xception

ABSTRAK

Machine learning dapat diterapkan dalam berbagai kebutuhan, seperti klasifikasi gambar. Klasifikasi penyakit tanaman sangat diperlukan dan sangat membantu sektor pertanian di jaman yang serba modern saat ini. Dengan adanya aplikasi yang dapat mengklasifikasikan penyakit pada suatu tanaman, maka para petani dapat mengetahui penyakit yang diderita oleh tanaman yang dipanen dengan akurat dan dapat menanggulanginya dengan lebih efisien dan efektif dibandingkan menggunakan metode tradisional yang bisa memakan waktu lebih lama. Penelitian ini bertujuan untuk mengetahui arsitektur tensorflow yang paling bagus diantara 3 arsitektur yang digunakan dalam penelitian ini yaitu MobileNet, DenseNet121, dan Xception untuk mengklasifikasikan 9 jenis penyakit tanaman tomat dan 1 tanaman tomat sehat. Dari penelitian ini dapat disimpulkan bahwa arsitektur DenseNet121 merupakan arsitektur terbaik untuk mengklasifikasikan 9 jenis penyakit tanaman tomat dan 1 tanaman tomat sehat. Dalam pengujian, model DenseNet121 mencapai kisaran 0,987 atau 98,7% pada nilai accuracy, precision, recall, dan F-1 score pada proses pengujian. Xception berada di peringkat kedua dengan nilai keempat metrik di kisaran 0,986 atau 98,6% dan MobileNet di peringkat terakhir dengan nilai keempat metrik di kisaran 0,973 atau 97,3%.

Kata Kunci: CNN, DenseNet121, Klasifikasi Gambar, Machine Learning, MobileNet, TensorFlow, Transfer Learning, Xception

1. PENDAHULUAN

Pertumbuhan teknologi yang semakin berkembang pesat dari tahun ke tahun. Dengan pesatnya perkembangan teknologi, berbagai macam teknologi sudah mulai diterapkan dalam berbagai sektor dalam kehidupan manusia seperti sektor pertanian, perikanan, pariwisata, dan sebagainya. Penerapan teknologi dalam sektor-sektor tersebut diharapkan dapat membantu dan mempermudah manusia dalam melaksanakan pekerjaannya dan mendapatkan hasil yang lebih baik dengan waktu yang lebih singkat.

Salah satu sektor terpenting adalah sektor pertanian yang merupakan salah satu sektor paling penting dalam kehidupan manusia sebagai salah satu sektor utama penyedia pangan. Penanganan sektor pertanian harus dipikirkan dan dilaksanakan dengan sebaik-baiknya untuk mendapatkan hasil yang optimal dan sesuai harapan.

Jika tidak ditangani dengan baik, maka dapat menimbulkan masalah yang serius bagi kesejahteraan kehidupan manusia terutama dalam bagian pangan.

Salah satu teknologi yang dapat membantu penanganan sektor pertanian adalah machine learning. Machine learning adalah subfield dari Artificial Intelligence yang berfokus pada pengembangan algoritma yang memungkinkan komputer untuk belajar dari data tanpa perlu diprogram secara eksplisit. Machine learning dapat diterapkan dalam berbagai kebutuhan, seperti klasifikasi gambar, rekomendasi produk, segmentasi pelanggan, dan analisis sentimen.

Klasifikasi penyakit tanaman sangat diperlukan dan sangat membantu sektor pertanian di jaman yang serba modern saat ini. Dengan adanya aplikasi yang dapat mengklasifikasikan penyakit pada suatu tanaman, maka para petani dapat mengetahui penyakit yang diderita oleh tanaman yang dipanen dengan akurat dan dapat menanggulanginya dengan lebih efisien dan efektif dibandingkan menggunakan metode tradisional yang bisa memakan waktu lebih lama.

2. METODE PENELITIAN

Terdapat 4 tahapan yang dilakukan pada penelitian ini agar penelitian dapat berjalan dengan baik dan sesuai tujuan.

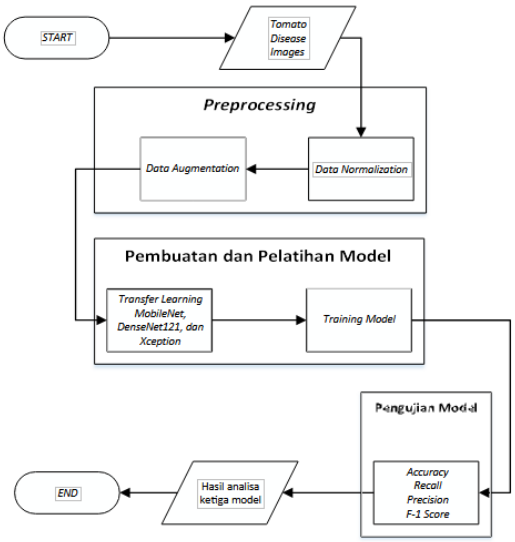
1. Pengumpulan Dataset
   Tahap pertama adalah pengumpulan dataset, yaitu tahap untuk mengumpulkan dataset dari kaggle. Dataset yang digunakan terdiri dari 9 jenis penyakit tanaman tomat yang sering dijumpai, yaitu yellow leaf curl virus, mosaic virus, target spot, spider mites, bacterial spot, leaf mold, early blight, late blight, dan septoria leaf spot, dan 1 jenis sehat tanaman tomat yaitu healthy. Datasetdibagi dengan rasio 80% data latih, 10% data validasi, dan 10% data uji.
2. Preprocessing
   Tahap kedua adalah preprocessing. Tahap ini melakukan serangkaian proses berupa data normalization dan data augmentation. Dataset dinormalisasikan terlebih dahulu dalam proses data normalization dan kemudian diolah pada proses data augmentation yang bertujuan untuk memperbanyak jumlah gambar pada data training.
3. Pembuatan dan pelatihan model
   Tahap ketiga adalah pembuatan dan pelatihan model. Tahap ini melalui serangkaian proses berupa pembuatan model dan pelatihan model berbasis MobileNet, DenseNet121, dan Xception. Proses pembuatan model menggunakan metode transfer learning guna mempersingkat waktu perancangan model dan meningkatkan akurasi. Tahapan proses pelatihan model mengimplementasikan perancangan model MobileNet, DenseNet121, dan Xception yang telah dibuat dengan melakukan pelatihan terhadap dataset hasil preprocessing yang berupa gambar tanaman tomat.
4. Pengujian model
   Tahap keempat adalah pengujian model. Model diuji dengan data testing sebanyak 1000 data gambar. Pengujian model memiliki tujuan untuk mengetahui keberhasilan proses pelatihan model dilakukan pada data gambar baru. Hasil pengujian model menjadi standar untuk mengukur kinerja model. Proses pengukuran performa model berdasarkan perhitungan 4 jenis metrik, yaitu accuracy, precission, recall, dan F-1 score. Nilai metrik yang dihasilkan dari pengujian model akan digunakan sebagai perbandingan antara 3 arsitektur yang diimplementasikan dalam penelitian ini.

3. ANALISIS DAN PEMBAHASAN

Gambaran Umum

Metode yang digunakan pada penelitian ini terdiri dari serangkaian tahapan proses sesuai dengan flowchart.

Tahap pertama yang dilakukan adalah pengumpulan dataset gambar yang berisi berbagai macam penyakit tanaman tomat yang dikumpulkan dari situs kaggle. Data yang berhasil dikumpulkan membentuk dataset yang berisi 9 kelas penyakit tanaman tomat yaitu yellow leaf curl virus, mosaic virus, target spot, spider mites, bacterial spot, leaf mold, early blight, late blight, dan septoria leaf spot dan 1 kelas tanaman tomat sehat



Gambar 1. Flowchart Tahapan Metode Penelitian

yaitu healthy. Tahap berikutnya yang dilakukan adalah membagi dataset dalam rasio yang sudah ditentukan yaitu 80% untuk data training, 10% untuk data validation, dan 10% data testing.

Tahap selanjutnya adalah tahap preprocessing. Tahap preprocessing terdiri dari 2 proses yaitu data normalization dan data augmentation. Kedua tahap ini bertujuan untuk menambah variasi pada gambar pada setiap kelas dalam data training dan untuk menormalisasikan piksel pada seluruh gambar di data training, validation, dan testing.

Tahapan selanjutnya adalah pembuatan dan pelatihan model yang terdiri dari serangkaian proses. Proses pertama yaitu proses pembuatan model yang dimulai dengan melakukan transfer learning menggunakan arsitektur MobileNet, DenseNet121, dan Xception. Pada proses ini digunakan transfer learning dengan tujuan mempersingkat waktu pelatihan model deep learning. Kemudian, dikukan metode fine-tuning untuk mendapatkan akurasi yang lebih tinggi dan mencegah dan/atau mengurangi kemungkinan terjadinya overfitting pada pelatihan model. Proses selanjutnya adalah pelatihan model pada data training yang divalidasi dengan data validation. Proses pelatihan akan dilaksanakan sebanyak 15 epoch, yang berarti akan dilakukan 15 kali pengulangan proses pelatihan pada data training.

Tahapan selanjutnya adalah pengujian model menggunakan data testing yang berjumlah 1000 gambar tomat. Pada tahapan ini dilakukan evaluasi hasil pengujian model menggunakan fungsi precision_recall_fscore_support dan classification_report dari library Scikit-learn. Evaluasi yang dilakukan menggunakan 4 jenis metrik, yaitu accuracy, precision, recall, dan F-1 Score. Hasil evaluasi tersebut menjadi standar untuk mengukur performa model. Setelah itu, setiap model akan dibandingkan antara satu sama lain menggunakan nilai keempat metrik tersebut.

Pengumpulan Dataset Gambar Penyakit Tanaman Tomat

Pada tahap pengumpulan data, dataset dikumpulkan melalui internet. Data gambar penyakit tanaman tomat diambil dari kaggle. Dataset penyakit tanaman tomat yang berhasil dikumpulkan sebanyak 10000 total gambar penyakit tanaman tomat yang terbagi dalam 9 kelas penyakit(yellow leaf curl virus, mosaic virus, target spot, spider mites, bacterial spot, leaf mold, early blight, late blight, dan septoria leaf spot) dan 1 kelas tanaman tomat sehat(healthy) dimana setiap kelas memiliki total 1000 gambar tomat. Untuk setiap kelas dipastikan memiliki jumlah gambar atau data yang sama sehingga tidak terjadi bias terhadap suatu atau beberapa kelas supaya model yang dihasilkan merupakan model yang cukup bagus dan sesuai.

Table 1. Tabel Contoh Gambar dari Kelas pada Dataset Penyakit Tomat

No	Nama Kelas	Contoh Gambar
1.	yellow leaf curl virus	

2.	mosaic virus	
3.	target spot	
4.	spider mites	
5.	bacterial spot	
6.	leaf mold	
7.	early blight	
8.	late blight	
9.	septoria leaf spot	

10.	healthy	

Pembagian Dataset

Pada tahapan ini, membagi dataset yang berisi 10000 gambar tomat menjadi 3 dataset, yaitu dataset untuk training, validation, dan testing. Pembagian dataset menggunakan rasio yang sudah ditentukan oleh penulis berdasarkan standar yang sudah direkomendasikan, yaitu 80 persen untuk dataset training, 10 persen untuk dataset validation, dan 10 persen untuk dataset training. Pada tahapan ini, penulis menggunakan train_test_split dari library sklearn untuk membagi dataset yang diimplementasikan menggunakan blok kode program dibawah ini.

```
train_df, ts_df = train_test_split(df, train_size = 0.8, shuffle = True, random_state = 42)

valid_df, test_df = train_test_split(ts_df, train_size = 0.5, shuffle = True, random_state = 42)
```

Preprocessing

Dalam tahapan preprocessing terdapat dua tahapan, yaitu data augmentation dan data normalization. Kedua tahapan tersebut bertujuan agar data-data di dalam dataset dapat memenuhi kriteria tertentu berdasarkan keinginan dan siap diproses untuk menjadi model machine learning.

Tahapan data normalization merupakan tahapan dimana proses penyesuaian skala fitur sehingga data dapat dibandingkan secara konsisten dan model dapat belajar lebih efektif. Tujuan dari tahapan ini adalah untuk membantu dalam meningkatkan stabilitas dan efisiensi proses pelatihan. Pada tahapan ini, nilai piksel dibagi pada setiap gambar dengan 255.

Tahapan data augmentation merupakan tahapan untuk memperbesar dataset dengan variasi pada gambar-gambar yang ada dengan cara melakukan transformasi pada gambar asli. Tujuannya dapat mencegah overfitting dan meningkatkan akurasi. Pada proses image augmentation dilakukan konfigurasi dengan menambahkan parameter rotation_range dengan nilai sebesar 30, width_shift_range sebesar 20%, height_shift_range sebesar 20%, shear_range sebesar 20%, zoom_range sebesar 20%, horizontal_flip bernilai True, dan terakhir adalah fill_mode bernilai nearestcolor_mode.

Pada tahapan ini digunakan ImageDataGenerator untuk menormalisasikan dataset sehingga mempermudah tahapan preprocessing dari library sklearn untuk membagi dataset yang diimplementasikan menggunakan blok kode program dibawah ini.

```
batch_size = 16

img_size = (224, 224)
train_gen = ImageDataGenerator(rescale=1. / 255,

rotation_range=30, 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_gen = ImageDataGenerator(rescale=1. / 255) test_gen = ImageDataGenerator(rescale=1. /
255)
```

```
train_set = train_gen.flow_from_dataframe( train_df,
x_col= 'filepaths', y_col= 'labels', target_size= img_size, class_mode= 'categorical', color_mode= 'rgb', shuffle= True, batch_size= batch_size)

valid_set = valid_gen.flow_from_dataframe( valid_df,
x_col= 'filepaths', y_col= 'labels', target_size= img_size, class_mode= 'categorical', shuffle= False, batch_size= batch_size)

test_set = test_gen.flow_from_dataframe( test_df,
x_col= 'filepaths', y_col= 'labels', target_size= img_size,
class_mode= 'categorical', shuffle= False, batch_size= batch_size)
```

Pembuatan dan Pelatihan Model

Tahapan ini memiliki 2 proses. Proses pertama yaitu membuat model menggunakan teknik transfer learning menggunakan arsitektur MobileNet, DenseNet121, dan Xception dan proses kedua merupakan kompilasi dan pelatihan model dengan menggunakan model MobileNet, DenseNet121, dan Xception yang sudah melalui proses fine-tuning dan data yang sudah dinormalisasi dan diaugmentasi pada tahap sebelumnya.

Transfer Learning menggunakan MobieNet, DenseNet121, dan Xception

MobileNet adalah arsitektur jaringan neural yang dirancang khusus untuk perangkat mobile dan aplikasi dengan sumber daya terbatas. DenseNet121 adalah varian dari arsitektur DenseNet yang menghubungkan setiap lapisan ke setiap lapisan lainnya dengan koneksi yang padat. Xception adalah arsitektur jaringan neural yang didasarkan pada konsep depthwise separable convolutions, yang merupakan varian dari arsitektur Inception. Ketiga arsiteruktur memiliki kelebihan dalam image classificication.

Pada proses pembuatan model digunakan beberapa input layer. Input layer pertama digunakan layer GlobalAveragePooling2D. Kemudian digunakan layer BatchNormalization. Selanjutnya, digunakan layer Dense yang memiliki 512 unit dan aktivasi ReLu yang diikuti layer BatchNormalization dan layer Dropout dengan rate yaitu 0,5. Kemudian digunakan layer Dense yang memiliki 256 unit dan aktivasi ReLu yang diikuti layer BatchNormalization dan layer Dropout dengan rate yaitu 0,5. Layer terakhir merupakan output layer dengan menggunakan layer Dense dengan unit sebanyak kelas gambar tomat yaitu 10 kelas dan fungsi aktivasi softmax. Proses transfer learning menggunakan MobileNet, DenseNet121, dan Xception diimplementasikan pada blok program di bawah ini.

```
img_size = (224, 224)

channels = 3

img_shape = (img_size[0], img_size[1], channels) class_counts = len(list(train_gen.class_indices.keys()))
base_model = EfficientNetB3(weights='imagenet', include_top=False, input_shape = img_shape, pooling= None)
```

```
base_model1 = MobileNet(weights='imagenet', include_top=False, input_shape
= img_shape, pooling= None)

x = base_model1.output

x = GlobalAveragePooling2D()(x) x = BatchNormalization()(x)
x = Dense(512, activation='relu')(x) x = BatchNormalization()(x)
x = Dropout(0.5)(x)

x = Dense(256, activation='relu')(x) x = BatchNormalization()(x)
x = Dropout(0.5)(x)

output = Dense(class_counts, activation = "softmax")(x) MobileNet_model =
Model(inputs=base_model1.input, outputs=output)
base_model2 = DenseNet121(weights='imagenet', include_top=False, input_shape = img_shape,
pooling= None)

x = base_model2.output

x = GlobalAveragePooling2D()(x) x = BatchNormalization()(x)
x = Dense(512, activation='relu')(x) x = BatchNormalization()(x)
x = Dropout(0.5)(x)

x = Dense(256, activation='relu')(x) x = BatchNormalization()(x)
x = Dropout(0.5)(x)

output = Dense(class_counts, activation = "softmax")(x) DenseNet121_model =
Model(inputs=base_model2.input, outputs=output)
base_model3 = Xception(weights='imagenet', include_top=False, input_shape = img_shape,
pooling= None)

x = base_model3.output

x = GlobalAveragePooling2D()(x) x = BatchNormalization()(x)
x = Dense(512, activation='relu')(x) x = BatchNormalization()(x)
x = Dropout(0.5)(x)

x = Dense(256, activation='relu')(x) x = BatchNormalization()(x)
x = Dropout(0.5)(x)

output = Dense(class_counts, activation = "softmax")(x)
Xception_model = Model(inputs=base_model3.input, outputs=output)
```

Kompilasi dan Pelatihan Model

Tahap pertama yang dilakukan adalah kompilasi model. Pada tahap ini digunakan Adamax sebagai Optimizer karena Adamax mampu memberikan optimasi yang sangat baik saat proses pelatihan dalam penelitian yang dilakukan. Learning Rate yang digunakan adalah 0,0001 sebagai nilai dari. Loss function Categorical Crossentrophy digunakan karena dataset yang digunakan merupakan multi-class classification atau dataset yang memiliki lebih dari satu kelas. Implementasi kode dari tahap kompilasi model diimplementasikan pada blok program di bawah ini.

```
#MobileNet MobileNet_model.compile(optimizer=Adamax(learning_rate=0.0001),
loss='categorical_crossentropy', metrics=['accuracy'])
MobileNet_model.summary()

#DenseNet121 DenseNet121_model.compile(optimizer=Adamax(learning_rate=0.0001),
loss='categorical_crossentropy', metrics=['accuracy'])

DenseNet121_model.summary()
#Xception Xception_model.compile(optimizer=Adamax(learning_rate=0.0001),
loss='categorical_crossentropy', metrics=['accuracy'])


Xception_model.summary()
```

Tahap selanjutnya yang dilakukan adalah pelatihan model. Tahap pelatihan model memiliki tujuan supaya model yang telah dibuat dapat mempelajari fitur-fitur yang terdapat pada dataset. Pada penelitian ini diberikan nilai epoch sebanyak 15 dengan nilai batch size adalah 16 berdasarkan tahapan preprocessing yang sudah dilakukan sebelumnya. Implementasi kode dari tahap pelatihan model diimplementasikan pada blok program di bawah ini.

```
epochs = 15 #MobileNet
MobileNet_history = MobileNet_model.fit(train_set,

epochs=epochs, verbose=1, validation_data=valid_set)
#DenseNet121

DenseNet121_history = DenseNet121_model.fit(train_set,

epochs=epochs, verbose=1, validation_data=valid_set)
#Xception

Xception_history = Xception_model.fit(train_set,

epochs=epochs, verbose=1, validation_data=valid_set)
```

Model yang telah dibuat akan dilatih dengan data training yang akan divalidasi dengan data validation. Hasil keseluruhan dari proses pelatihan model MobileNet, DenseNet121, dan Xception dapat dilihat pada gambar dibawah ini

Gambar 2. Hasil Keseluruhan Pelatihan Model MobileNet

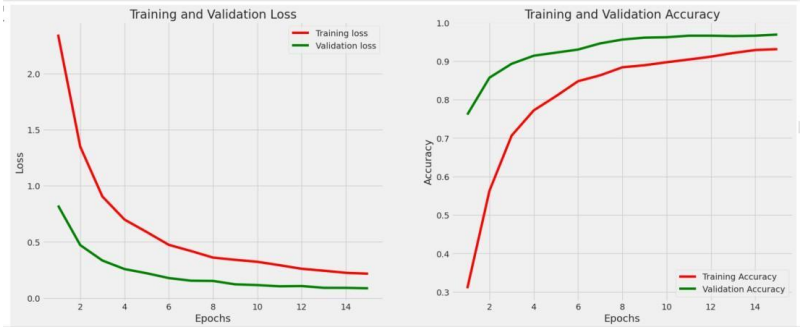
```
Epoch 1/15
500/500 ━━━━━━━━━━━━━━━━━━━━ 338s 342ms/step - accuracy: 0.2282 - loss: 2.8011 - val_accuracy: 0.7840 - val_loss: 0.7240
Epoch 2/15
500/500 ━━━━━━━━━━━━━━━━━━━━ 198s 308ms/step - accuracy: 0.6245 - loss: 1.1738 - val_accuracy: 0.9070 - val_loss: 0.3069
Epoch 3/15
500/500 ━━━━━━━━━━━━━━━━━━━━ 155s 306ms/step - accuracy: 0.7890 - loss: 0.6619 - val_accuracy: 0.9460 - val_loss: 0.1694
Epoch 4/15
500/500 ━━━━━━━━━━━━━━━━━━━━ 202s 306ms/step - accuracy: 0.8506 - loss: 0.4578 - val_accuracy: 0.9590 - val_loss: 0.1273
Epoch 5/15
500/500 ━━━━━━━━━━━━━━━━━━━━ 152s 300ms/step - accuracy: 0.8845 - loss: 0.3609 - val_accuracy: 0.9600 - val_loss: 0.1195
Epoch 6/15
500/500 ━━━━━━━━━━━━━━━━━━━━ 205s 307ms/step - accuracy: 0.9133 - loss: 0.2681 - val_accuracy: 0.9670 - val_loss: 0.1068
Epoch 7/15
500/500 ━━━━━━━━━━━━━━━━━━━━ 154s 305ms/step - accuracy: 0.9249 - loss: 0.2362 - val_accuracy: 0.9790 - val_loss: 0.0650
Epoch 8/15
500/500 ━━━━━━━━━━━━━━━━━━━━ 153s 302ms/step - accuracy: 0.9337 - loss: 0.2083 - val_accuracy: 0.9840 - val_loss: 0.0640
Epoch 9/15
500/500 ━━━━━━━━━━━━━━━━━━━━ 201s 300ms/step - accuracy: 0.9490 - loss: 0.1724 - val_accuracy: 0.9750 - val_loss: 0.0878
Epoch 10/15
500/500 ━━━━━━━━━━━━━━━━━━━━ 152s 301ms/step - accuracy: 0.9502 - loss: 0.1710 - val_accuracy: 0.9800 - val_loss: 0.0644
Epoch 11/15
500/500 ━━━━━━━━━━━━━━━━━━━━ 150s 297ms/step - accuracy: 0.9547 - loss: 0.1476 - val_accuracy: 0.9870 - val_loss: 0.0465
Epoch 12/15
500/500 ━━━━━━━━━━━━━━━━━━━━ 151s 299ms/step - accuracy: 0.9562 - loss: 0.1431 - val_accuracy: 0.9840 - val_loss: 0.0508
Epoch 13/15
500/500 ━━━━━━━━━━━━━━━━━━━━ 151s 299ms/step - accuracy: 0.9622 - loss: 0.1301 - val_accuracy: 0.9870 - val_loss: 0.0498
Epoch 14/15
500/500 ━━━━━━━━━━━━━━━━━━━━ 201s 298ms/step - accuracy: 0.9601 - loss: 0.1163 - val_accuracy: 0.9900 - val_loss: 0.0541
Epoch 15/15
500/500 ━━━━━━━━━━━━━━━━━━━━ 204s 302ms/step - accuracy: 0.9681 - loss: 0.1048 - val_accuracy: 0.9900 - val_loss: 0.0413
```

Gambar 3. Hasil Keseluruhan Pelatihan Model DenseNet121

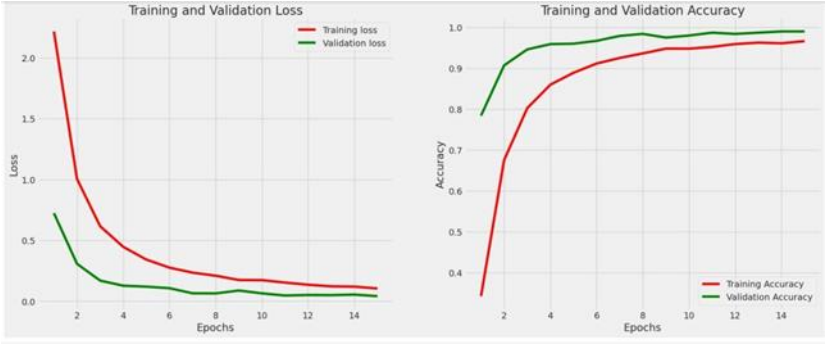
```
Epoch 1/15
/usr/local/lib/python3.10/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121: UserWarning: Your `PyD
  self._warn_if_super_not_called()
500/500 ━━━━━━━━━━━━━━━━━━━━ 2301s 4s/step - accuracy: 0.2034 - loss: 2.9807 - val_accuracy: 0.7200 - val_loss: 0.8946
Epoch 2/15
500/500 ━━━━━━━━━━━━━━━━━━━━ 160s 317ms/step - accuracy: 0.5462 - loss: 1.4463 - val_accuracy: 0.8400 - val_loss: 0.4727
Epoch 3/15
500/500 ━━━━━━━━━━━━━━━━━━━━ 160s 317ms/step - accuracy: 0.7089 - loss: 0.8990 - val_accuracy: 0.9090 - val_loss: 0.2821
Epoch 4/15
500/500 ━━━━━━━━━━━━━━━━━━━━ 160s 316ms/step - accuracy: 0.7887 - loss: 0.6410 - val_accuracy: 0.9300 - val_loss: 0.2058
Epoch 5/15
500/500 ━━━━━━━━━━━━━━━━━━━━ 201s 316ms/step - accuracy: 0.8408 - loss: 0.4849 - val_accuracy: 0.9620 - val_loss: 0.1306
Epoch 6/15
500/500 ━━━━━━━━━━━━━━━━━━━━ 160s 317ms/step - accuracy: 0.8689 - loss: 0.3986 - val_accuracy: 0.9650 - val_loss: 0.1118
Epoch 7/15
500/500 ━━━━━━━━━━━━━━━━━━━━ 202s 317ms/step - accuracy: 0.8981 - loss: 0.3180 - val_accuracy: 0.9750 - val_loss: 0.0890
Epoch 8/15
500/500 ━━━━━━━━━━━━━━━━━━━━ 162s 320ms/step - accuracy: 0.9068 - loss: 0.2979 - val_accuracy: 0.9750 - val_loss: 0.0921
Epoch 9/15
500/500 ━━━━━━━━━━━━━━━━━━━━ 161s 318ms/step - accuracy: 0.9280 - loss: 0.2314 - val_accuracy: 0.9760 - val_loss: 0.0697
Epoch 10/15
500/500 ━━━━━━━━━━━━━━━━━━━━ 199s 314ms/step - accuracy: 0.9307 - loss: 0.2107 - val_accuracy: 0.9820 - val_loss: 0.0690
Epoch 11/15
500/500 ━━━━━━━━━━━━━━━━━━━━ 202s 312ms/step - accuracy: 0.9407 - loss: 0.1783 - val_accuracy: 0.9830 - val_loss: 0.0661
Epoch 12/15
500/500 ━━━━━━━━━━━━━━━━━━━━ 159s 315ms/step - accuracy: 0.9484 - loss: 0.1640 - val_accuracy: 0.9850 - val_loss: 0.0521
Epoch 13/15
500/500 ━━━━━━━━━━━━━━━━━━━━ 158s 314ms/step - accuracy: 0.9504 - loss: 0.1568 - val_accuracy: 0.9850 - val_loss: 0.0622
Epoch 14/15
500/500 ━━━━━━━━━━━━━━━━━━━━ 202s 314ms/step - accuracy: 0.9577 - loss: 0.1334 - val_accuracy: 0.9850 - val_loss: 0.0569
Epoch 15/15
500/500 ━━━━━━━━━━━━━━━━━━━━ 159s 313ms/step - accuracy: 0.9554 - loss: 0.1436 - val_accuracy: 0.9860 - val_loss: 0.0497
```

Gambar 4. Hasil Keseluruhan Pelatihan Model Xception

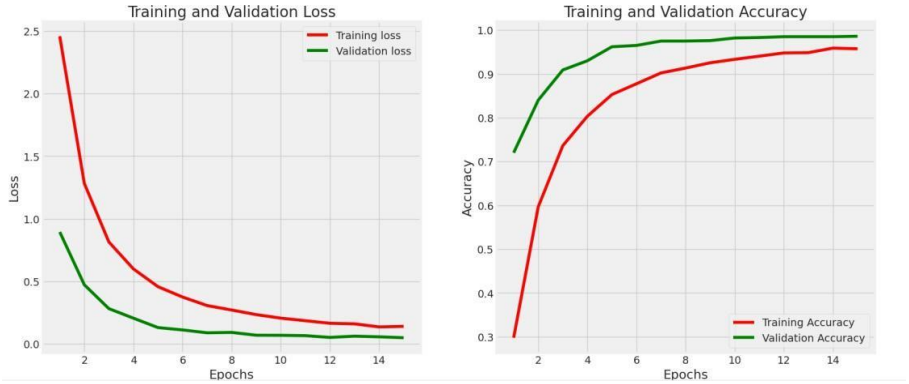
Hasil dari pelatihan setiap model direpresentasikan dalam bentuk grafik akurasi dan loss yang bisa dilihat pada gambar dibawah ini



Gambar 5. Grafik Akurasi dan Loss Model MobileNet



Gambar 6. Grafik Akurasi dan Loss Model DenseNet121



Gambar 7. Grafik Akurasi dan Loss Model Xception

Gambar diatas memiliki grafik dari akurasi model yang menunjukkan perbedaan hasil dari pelatihan model pada data training dan data validation. Garis tingkat akurasi pada pelatihan model menggunakan data training ditunjukkan dengan menggunakan garis berwarna merah, sedangkan garis tingkat akurasi pada pelatihan model menggunakan data validation ditunjukkan dengan menggunakan garis berwarna hijau. Berdasarkan grafik akurasi pada ketiga gambar tersebut, dapat dilihat bahwa semakin bertambahnya epoch, maka semakin meningkat hasil akurasinya.

Selain grafik akurasi, ketiga gambar tersebut memiliki grafik loss model yang menunjukkan perbedaan hasil dari pelatihan model pada data training dan data validation. Garis tingkat loss pada pelatihan model menggunakan data training ditunjukkan dengan menggunakan garis berwarna merah pada grafik, sedangkan garis tingkat loss pada pelatihan model menggunakan data validation ditunjukkan dengan menggunakan garis berwarna hijau pada grafik. Berdasarkan grafik pada ketiga gambar tersebut, dapat dilihat bahwa semakin bertambahnya epoch, maka semakin berkurang nilai loss dari model.

Pengujian Model

Tahapan pengujian model dalam penelitian ini menggunakan data testing yang berjumlah 1000 data yang terbagi dalam 10 kelas, yaitu yellow leaf curl virus, mosaic virus, target spot, spider mites, bacterial spot, leaf mold, early blight, late blight, septoria leaf spot, dan healthy. Pada penelitian ini digunakan precision_recall_fscore_support dan classification_report dari Scikit-learn yang dilakukan pada data uji untuk mendapatkan metric accuracy, precision, recall, dan F-1 Score. Implementasi kode dari tahap pengujian model diimplementasikan pada blok program di bawah ini.

```
# Fungsi untuk evaluasi model dan classification report def classification_report_metrics(model,
test_set, classes):
y_true = test_set.classes

y_pred = np.argmax(model.predict(test_set), axis=1)
report = classification_report(y_true, y_pred, target_names=classes) precision, recall, f1, _ =
precision_recall_fscore_support(y_true, y_pred,
average='weighted')

mse = mean_squared_error(y_true, y_pred) rmse = np.sqrt(mse)
accuracy = accuracy_score(y_true, y_pred)

print("Classification Report:\n", report) print(f"Accuracy: {accuracy:.4f}") print(f"Precision:
{precision:.4f}") print(f"Recall: {recall:.4f}")
print(f"F1-Score: {f1:.4f}")

print(f"MSE: {mse:.4f}")

print(f"RMSE: {rmse:.4f}")


# Function untuk plot confusion matrix

  def plot_confusion_matrix(model, test_set, classes): y_true = test_set.classes
y_pred = np.argmax(model.predict(test_set), axis=1) cm = confusion_matrix(y_true, y_pred)
plt.figure(figsize=(10, 10))

plt.imshow(cm, interpolation='nearest', cmap=plt.cm.Blues) plt.title('Confusion Matrix')
plt.colorbar()

tick_marks = np.arange(len(classes)) plt.xticks(tick_marks, classes, rotation=45) plt.yticks(tick_marks,
classes)


thresh = cm.max() / 2.

for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
```

Hasil evaluasi model yang didapatkan untuk menampilkan accuracy, precision, recall, dan F-1 score dari model MobileNet, DenseNet121, dan Xception dapat dilihat pada gambar berikut.

Classification Report:

	precision	recall	f1-score	support
Tomato___Bacterial_spot	0.98	0.98	0.98	96
Tomato___Early_blight	0.98	0.95	0.96	113
Tomato___Late_blight	0.95	0.98	0.96	89
Tomato___Leaf_Mold	1.00	0.97	0.99	106
Tomato___Septoria_leaf_spot	0.95	0.98	0.97	99
Tomato___Spider_mites	0.96	0.96	0.96	100
Tomato___Target_Spot	0.93	0.95	0.94	114
Tomato___Tomato_Yellow_Leaf_Curl_Virus	1.00	0.98	0.99	104
Tomato___Tomato_mosaic_virus	0.99	1.00	0.99	78
Tomato___healthy	1.00	1.00	1.00	101
accuracy			0.97	1000
macro avg	0.97	0.97	0.97	1000
weighted avg	0.97	0.97	0.97	1000

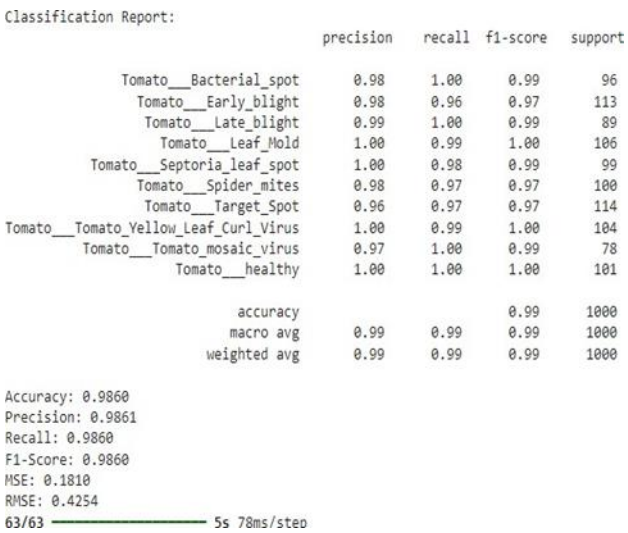
Accuracy: 0.9730
Precision: 0.9734
Recall: 0.9730
F1-Score: 0.9731
MSE: 0.2410
RMSE: 0.4909

Classification Report:

	precision	recall	f1-score	support
Tomato___Bacterial_spot	0.98	1.00	0.99	96
Tomato___Early_blight	1.00	0.98	0.99	113
Tomato___Late_blight	0.99	1.00	0.99	89
Tomato___Leaf_Mold	1.00	0.98	0.99	106
Tomato___Septoria_leaf_spot	0.97	0.99	0.98	99
Tomato___Spider_mites	0.98	0.97	0.97	100
Tomato___Target_Spot	0.97	0.97	0.97	114
Tomato___Tomato_Yellow_Leaf_Curl_Virus	1.00	0.98	0.99	104
Tomato___Tomato_mosaic_virus	0.99	1.00	0.99	78
Tomato___healthy	0.99	1.00	1.00	101
accuracy			0.99	1000
macro avg	0.99	0.99	0.99	1000
weighted avg	0.99	0.99	0.99	1000

Accuracy: 0.9870
Precision: 0.9871
Recall: 0.9870
F1-Score: 0.9870
MSE: 0.1480
RMSE: 0.3847

Gambar 8. Hasil Evaluasi Model MobileNet Hasil Evaluasi Model DenseNet121



```
Classification Report:
                                        precision    recall  f1-score   support

                Tomato___Bacterial_spot       0.98      1.00      0.99        96
                  Tomato___Early_blight       0.98      0.96      0.97       113
                   Tomato___Late_blight       0.99      1.00      0.99        89
                     Tomato___Leaf_Mold       1.00      0.99      1.00       106
            Tomato___Septoria_leaf_spot       1.00      0.98      0.99        99
                 Tomato___Spider_mites       0.98      0.97      0.97       100
                   Tomato___Target_Spot       0.96      0.97      0.97       114
Tomato___Tomato_Yellow_Leaf_Curl_Virus       1.00      0.99      1.00       104
          Tomato___Tomato_mosaic_virus       0.97      1.00      0.99        78
                      Tomato___healthy       1.00      1.00      1.00       101

                              accuracy                           0.99      1000
                             macro avg       0.99      0.99      0.99      1000
                          weighted avg       0.99      0.99      0.99      1000

Accuracy: 0.9860
Precision: 0.9861
Recall: 0.9860
F1-Score: 0.9860
MSE: 0.1810
RMSE: 0.4254
63/63 ━━━━━━━━━━━━━━━━━━━━ 5s 78ms/step
```

Gambar 9. Hasil Evaluasi Model Xception

Analisis Hasil Penelitian

Dari hasil pengujian model, nilai keempat metrik yang didapatkan yaitu nilai accuracy, precision, recall, dan F-1 score. Keempat nilai tersebut akan dibandingkan satu sama lain untuk menemukan arsitektur yang paling bagus untuk mengklasifikan penyakit tanaman tomat berdasarkan dataset yang digunakan.

Table 2. Tabel Nilai Accuracy, Precision, Recall, dan F-1 Score dari Model MobileNet, DenseNet121, dan Xception

No	Arsitektur	Accuracy	Precision	Recall	F-1 Score
1.	MobileNet	0,973	0,9734	0,973	0,9731
2.	DenseNet121	0,987	0,9871	0,987	0,987
3.	Xception	0,986	0,9861	0,986	0,986

Berdasarkan data tabel diatas, peneliti mengambil kesimpulan dalam penelitian ini bahwa arsitektur terbaik untuk digunakan dalam model klasifikasi jenis penyakit tanaman tomat adalah DenseNet121 dengan nilai keempat metriknya mengungguli nilai keempat metrik dari 2 arsitektur lain yaitu MobileNet dan Xception dengan nilai accuracy, precision, recall, dan F-1 score berada di kisaran 0,987 atau 98,7%. Xception menduduki peringkat kedua dengan perbedaan tipis dengan tempat pertama yaitu sebesar 0,001 dengan nilai accuracy, precision, recall, dan F-1 score di kisaran 0,986 atau 98,6%. MobileNet menduduki peringkat terakhir dengan nilai accuracy, precision, recall, dan F-1 score di kisaran 0,973 atau 97,3%. Berdasarkan nilai yang didapatkan, peneliti dapat menyimpulkan model Xception dan MobileNet memiliki performa yang terbilang juga bagus untuk proses klasifikasi jenis penyakit tanaman tomat walaupun menempati peringkat kedua dan peringkat terakhir.

MobileNet memiliki keunggulan berupa ukuran model yang kecil, kecepatan inferensi tinggi, dan latensi rendah, menjadikannya ideal untuk aplikasi real-time di perangkat dengan sumber daya terbatas. Namun, MobileNet memiliki kekurangan dalam hal akurasi yang lebih rendah dan kemampuan generalisasi yang terbatas pada dataset yang kompleks. DenseNet121 unggul dalam efisiensi penggunaan parameter, peningkatan aliran gradien, dan pengurangan risiko overfitting, terutama pada dataset yang lebih kecil. Namun, arsitektur DenseNet121 lebih kompleks dan membutuhkan waktu pelatihan yang lebih lama serta memori yang lebih tinggi karena banyaknya koneksi antar-lapisan. Xception menawarkan akurasi tinggi dan efisiensi komputasi dalam tugas klasifikasi gambar, tetapi membutuhkan sumber daya komputasi yang signifikan, waktu pelatihan yang lebih lama, dan implementasi yang lebih kompleks dibandingkan model lainnya.

4. KESIMPULAN DAN SARAN

Kesimpulan yang dapat diambil berdasarkan penelitian yang telah dilakukan yaitu analisa perbandingan hasil klasifikasi jenis penyakit tanaman tomat menggunakan arsitektur MobileNet, DenseNet121, dan Xception telah berhasil dilakukan. Peneliti dapat menyimpulkan bahwa arsitektur DenseNet121 adalah arsitektur terbaik untuk mengklasifikasikan jenis penyakit tanaman tomat dengan nilai accuracy, precision, recall, dan F-1 score berada di kisaran 0,987 atau 98,7% dan diikuti Xception di peringkat kedua dengan perbedaan tipis di tempat pertama yaitu sebesar 0,001 dengan nilai accuracy, precision, recall, dan F-1 score di kisaran 0,986 atau 98,6%.

MobileNet menempati peringkat terakhir dengan nilai accuracy, precision, recall, dan F-1 score di kisaran 0,973 atau 97,3%. Walaupun model Xception dan MobileNet menempati peringkat kedua dan terakhir, peneliti menganalisa bahwa kedua model tersebut juga memiliki performa yang terbilang sangat bagus untuk proses klasifikasi jenis penyakit tanaman tomat dalam penelitian ini.

MobileNet memiliki keunggulan berupa ukuran model yang kecil, kecepatan inferensi tinggi, dan latensi rendah, menjadikannya ideal untuk aplikasi real-time di perangkat dengan sumber daya terbatas. Namun, MobileNet memiliki kekurangan dalam hal akurasi yang lebih rendah dan kemampuan generalisasi yang terbatas pada dataset yang kompleks. DenseNet121 unggul dalam efisiensi penggunaan parameter, peningkatan aliran gradien, dan pengurangan risiko overfitting, terutama pada dataset yang lebih kecil. Namun, arsitektur DenseNet121 lebih kompleks dan membutuhkan waktu pelatihan yang lebih lama serta memori yang lebih tinggi karena banyaknya koneksi antar-lapisan. Xception menawarkan akurasi tinggi dan efisiensi komputasi dalam tugas klasifikasi gambar, tetapi membutuhkan sumber daya komputasi yang signifikan, waktu pelatihan yang lebih lama, dan implementasi yang lebih kompleks dibandingkan model lainnya.

Berdasarkan kesimpulan diatas, dapat diberikan beberapa saran. Pertama, diharapkan untuk penelitian selanjutnya dapat mengembangkan model dengan menggunakan arsitektur yang berbeda seperti ResNet, VGG, dan EfficientNet sehingga dapat dibandingkan dengan DenseNet121 untuk mencari arsitektur yang lebih baik. Kedua, diharapkan model yang didapatkan di penelitian ini dapat dikembangkan dalam suatu aplikasi klasifikasi gambar dalam versi android sehingga membuat pengguna dapat mengklasifikasikan penyakit dari suatu tanaman tomat dengan akses yang lebih mudah. Ketiga, diharapkan penelitian ini dapat dikembangkan lebih lanjut untuk bukan hanya mengklasifikasikan penyakit tanaman tomat tetapi juga untuk klasifikasi penyakit tanaman lain.

DAFTAR PUSTAKA

[1] Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Springer.
[2] Brownlee, J. (2016). Machine Learning Mastery with Python: Understand Your Data, Create Accurate Models, and Work Projects End-to-End. Machine Learning Mastery.
[3] Chollet, F. (2017). Deep Learning with Python. Manning Publications.
[4] McKinney, W. (2018). Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython (2nd ed.). O'Reilly Media.
[5] Muller, A. C., & Guido, S. (2016). Introduction to Machine Learning with Python: A Guide for Data Scientists. O'Reilly Media.
[6] Murphy, K. P. (2012). Machine Learning: A Probabilistic Perspective. MIT Press.
[7] Russell, S., & Norvig, P. (2021). Artificial Intelligence: A Modern Approach (Fourth Edition). Pearson.
[8] Raschka, S., & Mirjalili, V. (2019). Python Machine Learning: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow 2. Packt Publishing.
[9] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press. https://www.deeplearningbook.org/
[10] Schmidhuber, J. (2015). Deep Learning in Neural Networks: An Overview. Neural Networks, 61, 85-117.
[11] Shorten, C., & Khoshgoftaar, T. M. (2019). A Survey on Image Data Augmentation for Deep Learning. Journal of Big Data, 6(1), 60.
[12] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research, 15, 1929-1958.
[13] Chollet, F. (2017). Xception: Deep Learning with Depthwise Separable Convolutions. arXiv preprint arXiv:1610.02357. *https://arxiv.org/abs/1610.02357*
[14] Gal, Y., & Ghahramani, Z. (2016). Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. arXiv preprint arXiv:1506.02142. *https://arxiv.org/abs/1506.02142*
[15] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., & Sergey, I. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. arXiv preprint arXiv:1704.04861. *https://arxiv.org/abs/1704.04861*
[16] Huang, G., Liu, Z., van der Maaten, L., & Weinberger, K. Q. (2017). "Densely Connected Convolutional Networks." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). *https://ieeexplore.ieee.org/document/8099726*
[17] Ioffe, S., & Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. arXiv preprint arXiv:1502.03167. *https://arxiv.org/abs/1502.03167*
[18] Kingma, D. P., & Ba, J. (2014). Adam: A Method for Stochastic Optimization. arXiv preprint arXiv:1412.6980. *https://arxiv.org/abs/1412.6980*
[19] Pan, S. J., & Yang, Q. (2010). A Survey on Transfer Learning. IEEE Transactions on Knowledge and Data Engineering, 22(10), 1345-1359. *https://ieeexplore.ieee.org/document/5288526*
[20] Hunter, J. D. (2007). Matplotlib: A 2D Graphics Environment. Computing in Science & Engineering, 9(3), 90-95. *https://ieeexplore.ieee.org/document/4160265*

[21] Waskom, M. L. (2021). seaborn: statistical data visualization. Journal of Open Source Software, 6(60), 3021. *https://doi.org/10.21105/joss.03021*
[22] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep Learning. Nature, 521(7553), 436-444. *https://www.nature.com/articles/nature14539*
[23] Kaggle. (2024). Datasets. *https://www.kaggle.com/datasets*
[24] Keras. (2024). GlobalAveragePooling2D Layer. *https://keras.io/api/layers/pooling_layers/global_average_pooling2d/*
[25] NumPy. (2024). NumPy Documentation. *https://numpy.org/doc/stable/*
[26] Pandas Documentation. (2024). Pandas Documentation. *https://pandas.pydata.org/pandas-docs/stable/*
[27] Python Software Foundation. (2024). Python Documentation. *https://docs.python.org/3/*
[28] TensorFlow. (2024). Image Augmentation. *https://www.tensorflow.org/tutorials/images/data_augmentation*