Jurnal of Business and Audit Information System (JBASE)
Versi Online: http://journal.ubm.ac.id/index.php/jbase
DOI: dx.doi.org/ 10.30813/jbase.v8i1.8116

Vol. 8(No.1): 1 – 12. Th.2025
p-ISSN: 2615-6431
e-ISSN: 2620-7907

Perbandingan Performa Arsitektur *Convolutional Neural Network* Menggunakan *Transfer Learning* untuk Model Deteksi Kesehatan Daun

Comparison of Convolutional Neural Network Architecture Performance Using Transfer Learning for Leaf Healthiness Detection Models

Eko Ariawan[1)*]

[1)]Rekayasa Perangkat Lunak/Sekolah Science Technology Engineering and Mathematics, Universitas Prasetiya Mulya

Diajukan 21 Februari 2025 / Disetujui 20 Maret 2025

Abstrak

Serangan hama dan penyakit merupakan salah satu faktor yang berkontribusi pada kegagalan panen. Pada umumnya, para petani akan menggunakan pestisida dan beberapa kompetitor alami untuk membasmi hama dan penyakit pada tanaman. Analisis terhadap daun sedini mungkin dapat mencegah potensi gagal panen. Identifikasi daun yang terdeteksi sebagai daun yang sehat atau terkena penyakit dapat dilakukan dengan kecerdasan buatan (*Artificial Intelligence*) dengan menggunakan teknik *deep learning*. *Convolutional Neural Network* (CNN) merupakan arsitektur deep learning yang efektif digunakan dalam berbagai aplikasi pengolahan citra. Penelitian ini bertujuan untuk mengevaluasi performa arsitektur deep learning pada beberapa algoritma CNN. Citra akan melalui proses data *augmentation* berupa *rescale*, *shear*, *zoom*, dan *resizing*. Dataset diklasifikasikan kedalam tiga kategori yaitu training, testing, dan validation dimana didalam nya memiliki dua kategori yaitu daun sehat dan daun terkena penyakit. Total sampel gambar daun terkena penyakit ada sebanyak 43.019 dan 10.284 sampel gambar daun sehat. Model CNN yang akan di uji adalah ResNet50, DenseNet121, MobileNet, VGG19, dan Xception dimana model CNN tersebut akan menggunakan library Keras dari modul TensorFlow V2.16.1. Dari hasil pengujian sampel yang dilakukan didapatkan hasil model terbaik adalah Xception, DenseNet121 dan ResNet50.
Kata Kunci: *Artificial Intelligence*, *Deep Learning*, *Convolutional Neural Network*, *Transfer Learning*, Perbandingan.

Abstract

Pest and leaf diseases attacks are one of the factors that contribute to crop failure. In general, farmers will use pesticides and some natural competitors to eradicate pests and diseases in plants. Early warning of leaf disease may contribute to prevent potential crop failure. Identification of healthy leaves or diseased can be done with artificial intelligence (AI) using deep learning techniques. Convolutional Neural Network (CNN) is a deep learning architecture that is effective in various image processing applications. This study aims to evaluate the performance of deep learning architectures on several CNN algorithms. The image will go through a data augmentation process such like rescale, shear, zoom, and resizing. The dataset is classified into three categories, training, testing, and validation, with two categories of healthy and diseased leaves on those every main category. The total sample images of diseased leaves are 43,019 and 10,284 samples of healthy leaf images. The CNN models to be tested are ResNet50, DenseNet121, MobileNet, VGG19, and Xception where the CNN models will use the Keras library from the TensorFlow V2.16.1 module. Results of this research identifies that the best model are Xception, DenseNet121 and ResNet50.
Keywords: *Artificial Intelligence, Deep Learning, Convolutional Neural Network, Transfer Learning, Comparison.*

*Korespondensi Penulis:
E-mail: eko.ariawan@prasetiyamulya.ac.id

Jurnal of Business and Audit Information System (JBASE)
Versi Online: http://journal.ubm.ac.id/index.php/jbase
DOI: dx.doi.org/ 10.30813/jbase.v8i1.8116

Vol. 8(No.1): 1 – 12. Th.2025
p-ISSN: 2615-6431
e-ISSN: 2620-7907

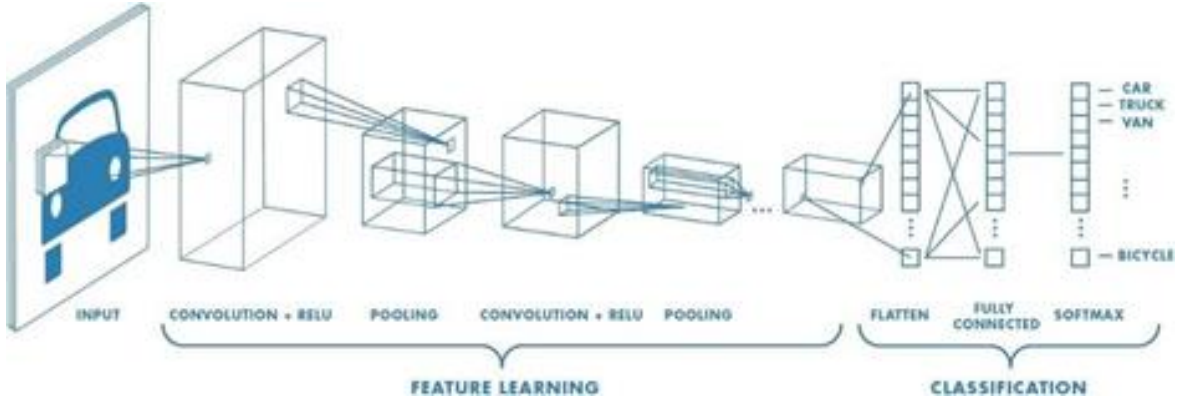
Pendahuluan

Deep learning merupakan topik yang cukup banyak menjadi sorotan saat ini. Terutama dalam pembahasan yang berkaitan dengan sebuah citra (Santoso & Ariyanto, 2018). *Deep learning* memiliki kemampuan yang signifikan untuk memodelkan data dengan metode *Convolutional Neural Network* (CNN). Banyak sekali penelitian yang menerapkan CNN seperti deteksi penggunaan masker (Putra, Sulistyowati, & Yudisthiana, 2023), klasifikasi penyakit daun padi (Khoiruddin, Junaidi, & Saputra, 2022), klasifikasi penyakit daun tomat (Nurdin, Kartika, & Najaf, 2024), klasifikasi bumbu dan rempah (Wulandari, Yasin, & Widiharih, 2020), implementasi sistem *self checkout* di toko (Anhar, 2023), klasifikasi batik (Azmi, Defit, & Sumijan, 2023), klasifikasi ikan (Ali SA & Sugiarto, 2023), deteksi COVID-19 pada X-Ray Thorax (Suryawan & Udayana, 2022), dan masih banyak sekali artikel akademik yang membahas tentang CNN. Namun rata-rata penulisan yang ada menggunakan satu algoritma CNN yang diimplementasikan terhadap topik yang sedang di teliti.

Pada penelitian ini ingin mencoba melihat algoritma CNN mana yang akan memberikan performa yang paling baik dalam mendeteksi sebuah citra daun yang sehat atau terkena penyakit. Dalam menjalankan perbandingan, platform yang akan digunakan adalah menggunakan TensorFlow yang merupakan sebuah *platform* yang lengkap untuk membentuk dan menjalankan model *Machine Learning* dari *library* Keras. Sehingga fokus dari penelitian ini bukanlah untuk membentuk algoritma namun lebih kepada analisis data dari sejumlah input untuk dilihat hasil luaran nya.

Convolutional Neural Network

Neural network adalah sebuah model pada *Artificial Intelligence* yang dapat dilatih (*training*) untuk mengenali sebuah pola (*pattern*). Secara umum arsitektur *Neural Network* terdiri dari input layer, setidaknya ada satu hidden layer dan output layer. Setiap layer akan terhubung oleh jaringan (*Neurons*). Sedangkan *Convolutional Neural Network* merupakan versi Multi Layer Perceptron yang tergolong kedalam *deep forward artificial neural network*. Cara kerja CNN menyerupai *visual cortex* dimana terdapat pola konektivitas antar neuron. Arsitektur CNN sama seperti neural network pada umumnya namun terdapat beberapa *hidden layer* yang pada umumnya berisi *convolutional layer*, *pooling layer*, *normalization layer*, *activation layer* (ReLU), *fully connected layer* dan *loss layer*.



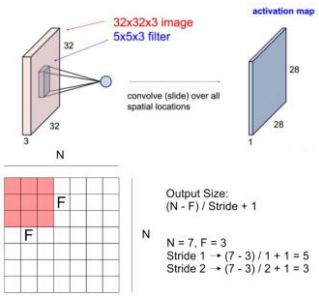
Gambar1. Cara Kerja *Convolutional Neural Network*

CNN Gambar 1, digunakan untuk klasifikasi data dengan metode supervised learning sehingga input pada jaringan neural telah memiliki target yang telah diketahui (Warsito, 2009). Model arsitektur CNN akan menggunakan pengetahuan yang telah diperoleh untuk klasifikasi dataset baru, dimana proses ini disebut sebagai *Transfer Learning*. Proses ini mampu mengekstrak lebih banyak fitur dari gambar dan menghasilkan representasi yang lebih baik.

Data pada layer convolutional akan mengalami proses konversi dan menghasilkan layer aktivasi (*Activation Map*) atau *feature map* 2D. Pada layer yang ditampilkan pada Gambar 2, data

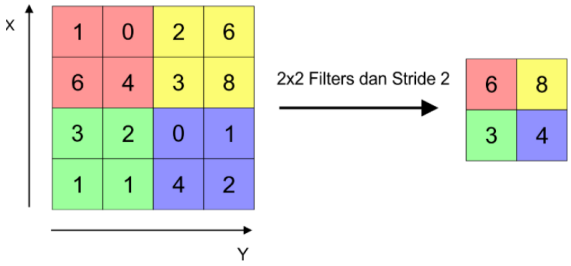
Vol. 8(No.1): 1 – 12. Th.2025
p-ISSN: 2615-6431
e-ISSN: 2620-7907

akan mengalami optimalisasi yang berasal dari parameter depth, stride dan padding (O'Shea & Nash, 2015).



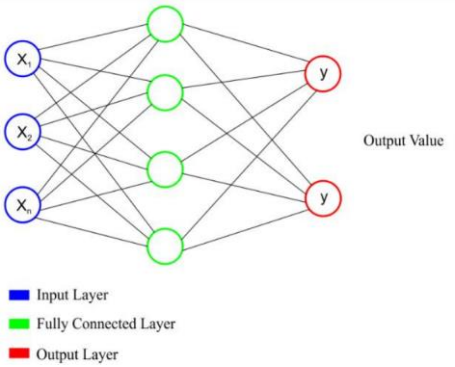
Gambar 2. *Convolutional Layer*

Fungsi aktivasi yang umumnya digunakan adalah tanh() atau ReLU(). Aktivasi ReLU() lebih populer karena hasilnya lebih baik. *Output* akan bernilai nol jika *input*nya negatif dan *output* akan sama dengan input jika inputnya positif. Setelah data melalui proses di layer *convolutional*, maka proses berikut nya adalah *pooling layer*. Biasanya menggunakan metode *Max Pooling* dan *Average Pooling*. Contoh Gambar 3 jika menggunakan *Max Pooling* 2x2 dan *Stride* 2 maka hasil yang didapatkan adalah nilai terbesar yang ada dari area tersebut. Namun pada *average pooling* akan menghasilkan nilai rata-rata dari area tersebut.



Gambar 3. Contoh *Max Pooling*

Hasil luaran dari *pooling layer* akan membentuk *multidimensional array*. Array ini perlu diproses *flatten* atau *reshape* yang menghasilkan sebuah vektor untuk data input pada *fully connected layer* pada Gambar 4.

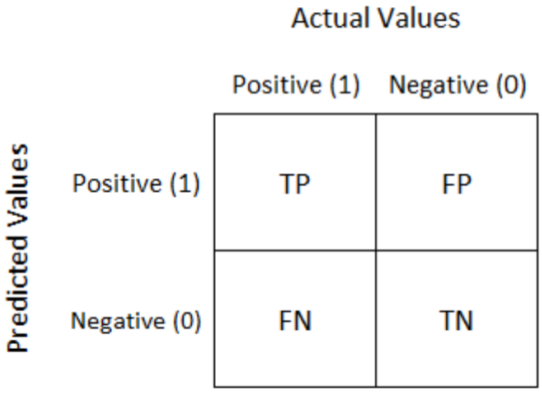


Gambar 4. *Fully Connected Layer*

Performance Metric

Nilai yang diperoleh dalam *performance metric* biasanya berupa nilai *accuracy*, *precision*, *recall* dan F1-*score* yang umumnya dihitung dengan metode *confusion matrix*, sebuah metode yang diperkenalkan oleh Karl Pearson pada tahun 1904 dengan nama awal *Contingency Table* (Pearson, 1904). Metode ini merupakan sebuah tabel dengan 4 kombinasi nilai prediksi dan aktual yaitu *True Positif*, *True Negatif*, *False Positif* dan *False Negatif*, terlihat pada Gambar 5.

Vol. 8(No.1): 1 – 12. Th.2025
p-ISSN: 2615-6431
e-ISSN: 2620-7907



Gambar 5. *Confusion Matrix*

True Positif adalah ketika prediksi positif dan benar, *True* Negatif adalah ketika prediksi negatif dan benar, *False* Positif adalah ketika prediksi positif dan salah, *False* Negatif adalah ketika prediksi negatif dan salah. Nilai prediksi adalah luaran dari program sedangkan nilai aktual adalah nilai sebenarnya dimana nilainya adalah *True* dan *False*.

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

Precision adalah analisis akurasi yang dilakukan dengan membandingkan nilai benar pada prediksi positif dan negatif. Contohnya adalah ketika akurasi yang dicapai adalah 99% namun bila precision hanya 50%, kondisi sebenarnya adalah probability benar dalam setiap kejadian hanya 50% saja. Nilai *precision* dapat dihitung dengan persamaan sebagai berikut.

$$precision = \frac{TP}{FP + TP}$$

Model *recall* adalah sebuah mekanisme untuk melihat apakah sebuah *machine learning* yang sedang diuji merupakan sebuah model yang baik untuk identifikasi sampel positif. Sehingga apabila nilai recall rendah, maka model tersebut tidak cukup baik untuk digunakan dalam identifikasi sampel positif. Sehingga *recall* dapat digunakan bersama *accuracy* dan *precision* untuk menilai apakah sebuah model konsisten baik untuk melakukan identifikasi sampel.

$$recall = \frac{TP}{FN + TP}$$

F1 score menggambarkan sebuah nilai pada model yang mengkompensasi nilai *precision* dan *recall*. F-*score* adalah matrik penilaian untuk digunakan dalam menilai model sebuah *machine learning* yang memberikan bobot yang imbang antara *Precision* dan *Recall*.

$$F1\ score = \frac{2 * precision * recall}{precision + recall}$$

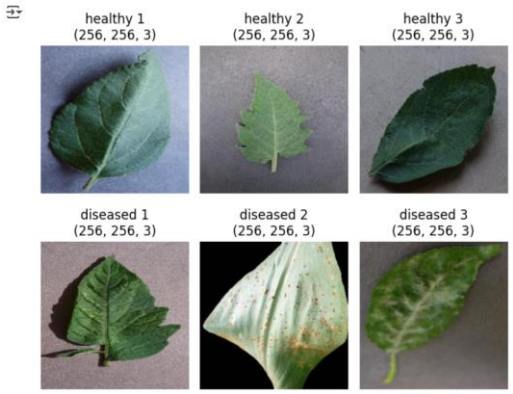
Metode Penelitian

Dataset Input

Seluruh data yang digunakan untuk proses disiapkan pada tempat terpisah antara kumpulan gambar daun sehat dan gambar daun yang terkena penyakit. Sumber data diambil dari Hewarathna

Vol. 8(No.1): 1 – 12. Th.2025
p-ISSN: 2615-6431
e-ISSN: 2620-7907

Ashen Iranga, seorang *datasets expert* yang membagikan datanya kepada umum dan dapat diunduh dari https://www.kaggle.com/datasets/asheniranga. Dataset yang ada telah melalui proses *preprocessing* berupa *cropping* ukuran menjadi sama dengan dimensi 256x256 *pixels*. Kemudian gambar akan di ubah menjadi 224x224 *pixels* sebelum gambar digunakan untuk proses *training*, *validation* dan *testing*.



Gambar 6. Menampilkan data gambar pada folder *train*

Data-data Gambar 6 harus diubah menjadi data numerik agar dapat di proses oleh algoritma CNN. TensorFlow telah memiliki fungsi untuk hal tersebut yaitu dengan menggunakan modul `ImageDataGenerator` yang merupakan bagian dari modul tensorflow.keras.preprocessing.image. Data direpresentasikan dalam angka yang memiliki rentang nilai 0-255 dimana setiap pixel diwakili oleh 3 kanal warna (R,G,B)

```
tf.random.set_seed(42)

train_datagen = ImageDataGenerator(
    rescale=1.0 / 255,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode="nearest"
)

val_datagen = ImageDataGenerator(rescale=1.0 / 255)

test_datagen = ImageDataGenerator(rescale=1.0 / 255)

img_size = (224, 224)

train_data = train_datagen.flow_from_directory(
    train_path,
    target_size=img_size,
    batch_size=32,
    class_mode="categorical",
    seed=42
)

val_data = val_datagen.flow_from_directory(
    val_path,
    target_size=img_size,
    batch_size=32,
    class_mode="categorical",
    seed=42
)

```

Vol. 8(No.1): 1 – 12. Th.2025
p-ISSN: 2615-6431
e-ISSN: 2620-7907

```
test_data = test_datagen.flow_from_directory(
    test_path,
    target_size=img_size,
    batch_size=32,
    class_mode="categorical",
    shuffle=False
)
```

```
Found 38104 images belonging to 2 classes.
Found 9458 images belonging to 2 classes.
Found 5741 images belonging to 2 classes.
```

Gambar 7. Implementasi modul ImageDataGenerator

Modul ImageData Generator yang ditampilkan pada Gambar 7 memiliki metode flow_from_directory untuk membaca gambar dari folder yang telah ditentukan dengan beberapa parameter seperti target_size untuk menyeragamkan ukuran pixel menjadi 224x224, batch_size 32 yang berarti berapa jumlah gambar yang akan digunakan untuk diproses oleh model dalam setiap kali iterasi pelatihan. Dengan ukuran *batch* yang digunakan adalah 32, maka akan terbentuk sejumlah data gambar yang akan digunakan pada setiap *batch* nya. Karena terdapat 38.104 gambar sampel yang digunakan untuk training maka 38.104/32 adalah 1.190,75 dan dengan pembulatan keatas maka jumlah batch dalam setiap iterasi ada 1.191, ditampilkan pada Gambar 8.

```
print(f"Jumlah batch (training): {len(train_data)}")

batch_1_train = train_data[0]

batch_1_train_data = train_data[0][0]
batch_1_train_label = train_data[0][1]
print(f"Jumlah gambar pada batch 1: {len(batch_1_train_data)}")
print(f"Jumlah label pada batch 1: {len(batch_1_train_label)}")

image_1_batch_1_train_data = train_data[0][0][0]

image_1_batch_1_train_label = train_data[0][1][0]
print(f"\nBatch 1 Gambar 1 train (data):\n{image_1_batch_1_train_data}")
print(f"\nBatch 1 Gambar 1 train (Label):\n{image_1_batch_1_train_label}")

```

```
Jumlah batch (training): 1191
Jumlah gambar pada batch 1: 32
Jumlah label pada batch 1: 32

Batch 1 Gambar 1 train (data):
[[[0.71325505 0.64266676 0.64266676]
  [0.7023227  0.6317345  0.6317345 ]
  [0.69076866 0.6201804  0.6201804 ]
  ...
  [0.68793255 0.64087373 0.64087373]
  [0.6228955  0.57583666 0.57583666]
  [0.7156626  0.6686038  0.6686038 ]]

  ...

Batch 1 Gambar 1 train (Label):
[1. 0.]
```

Gambar 8. Implementasi jumlah sampel gambar pada setiap iterasi

CNN *Modelling*

Penelitian ini akan menggunakan metode *transfer learning* dengan *pretrained* model. Ini merupakan model yang telah dilatih pada sebuah dataset yang besar dan kompleks yang memakan

waktu yang sangat lama dan sumber daya komputasi yang besar untuk mengidentifikasi dan mempelajari pola dari jutaan gambar. Dengan demikian waktu yang digunakan akan relatif lebih singkat. Proses pemodelan CNN menggunakan library Keras dari TensorFlow V2.16.1 yaitu ResNet50, DenseNet121, MobileNet, VGG19 dan Xception, pada Gambar 9.

```
base_model = VGG19(include_top=False, weights='imagenet', input_shape=(224, 224, 3))

model_vgg = Sequential()

model_vgg.add(base_model)

model_vgg.add(Flatten())
model_vgg.add(Dense(128, activation='relu'))
model_vgg.add(Dense(2, activation='softmax'))

base_model.trainable = False

model_vgg.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-
applications/vgg19/vgg19_weights_tf_dim_ordering_tf_kernels_notop.h5
80134624/80134624 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
Model: "sequential"
Total params: 23,236,034 (88.64 MB)
Trainable params: 3,211,650 (12.25 MB)
Non-trainable params: 20,024,384 (76.39 MB)
```

Gambar 9. Implementasi *transfer learning* model CNN

Variabel base_model digunakan untuk membentuk model CNN dengan parameter `weights='imagenet'` yang berarti bobot pelatihan mengikuti pada parameter sebelumnya yang telah ditentukan pada dataset ImageNet, parameter `include_top=False` berarti kita tidak memasukkan layer *fully connected* teratas dari model yang digunakan. Pada model ini kita menambahkan 3 layer diatas yaitu lapisan flatten dan lapisan dense yang memiliki 128 *neuron* dengan fungsi aktivasi relu dan lapisan dense yang memiliki 2 neuron dengan aktivasi *softmax* yang akan merepresentasikan hasil kategori daun *diseased* atau *healthy*.

Tabel 1. Model CNN

Model	Ringkasan Model
VGG19	Layer (type) | Output Shape | Param # vgg19 (Functional) | (None, 7, 7, 512) | 20,024,384 flatten (Flatten) | (None, 25088) | 0 dense (Dense) | (None, 128) | 3,211,392 dense_1 (Dense) | (None, 2) | 258
ResNet50	Layer (type) | Output Shape | Param # resnet50 (Functional) | (None, 7, 7, 2048) | 23,587,712 flatten_7 (Flatten) | (None, 100352) | 0 dense_14 (Dense) | (None, 128) | 12,845,184 dense_15 (Dense) | (None, 2) | 258

DenseNet121

Layer (type)	Output Shape	Param #
densenet121 (Functional)	(None, 7, 7, 1024)	7,037,504
flatten_2 (Flatten)	(None, 50176)	0
dense_4 (Dense)	(None, 128)	6,422,656
dense_5 (Dense)	(None, 2)	258

MobileNet

Layer (type)	Output Shape	Param #
mobilenet_1.00_224 (Functional)	(None, 7, 7, 1024)	3,228,864
flatten_6 (Flatten)	(None, 50176)	0
dense_12 (Dense)	(None, 128)	6,422,656
dense_13 (Dense)	(None, 2)	258

Xception

Layer (type)	Output Shape	Param #
xception (Functional)	(None, 7, 7, 2048)	20,861,480
flatten_5 (Flatten)	(None, 100352)	0
dense_10 (Dense)	(None, 128)	12,845,184
dense_11 (Dense)	(None, 2)	258

Sebelum menuju proses *training* dan *testing*, model akan dikompilasi dengan *optimizer* Adam (Adaptive Moment Estimation). Optimizer Tabel 1 bertujuan untuk meningkatkan efisiensi training nanti. Dari hasil penelitian yang dilakukan oleh Andi Nurdin, *optimizer* Adam terbukti menunjukkan performa yang baik (Nurdin, Kartika, & Najaf, 2024).

Training dan *Testing*

Model CNN yang telah terbentuk kemudian dilanjutkan pada tahap *training* dan *testing* dengan konfigurasi epoch 2. Pada tahap ini akan terbentuk nilai loss dan akurasi. Hasil inilah yang akan kemudian ditabulasi dan kemudian dianalisis untuk melihat performa dari CNN yang memberikan hasil yang terbaik. Karena jumlah iterasi yang cukup banyak dan sumberdaya komputasi yang terbatas maka dengan konfigurasi epoch yang sedikit ini diharapkan sudah cukup dapat memberikan gambaran akurasi dari masing-masing model, ditampilkan pada Gambar 10.

```
1. history_vgg = model_vgg.fit(
2.         train_data,
3.         validation_data=val_data,
4.         epochs=2,
5. )
```

```
Epoch 1/2
1191/1191 ━━━━━━━━━━━━━━━━━━━━ 588s 483ms/step - accuracy: 0.8312 - loss: 0.4222 - val_accuracy:
0.9008 - val_loss: 0.2320
Epoch 2/2
1191/1191 ━━━━━━━━━━━━━━━━━━━━ 561s 471ms/step - accuracy: 0.8896 - loss: 0.2413 - val_accuracy:
0.9096 - val_loss: 0.2048
```

Gambar 10. Implementasi *training dataset*

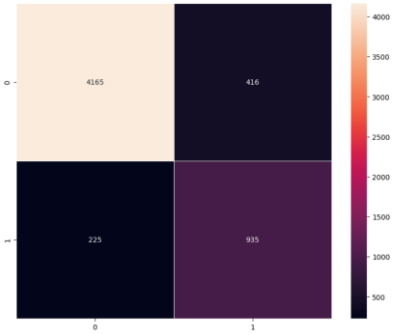
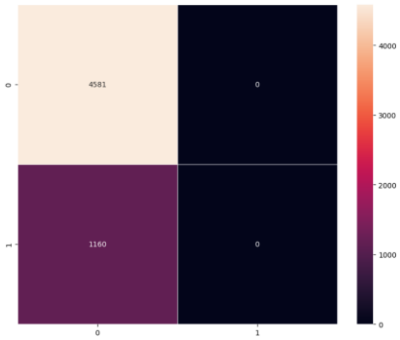
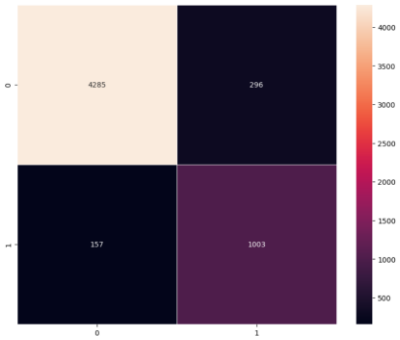
Hasil Dan Pembahasan

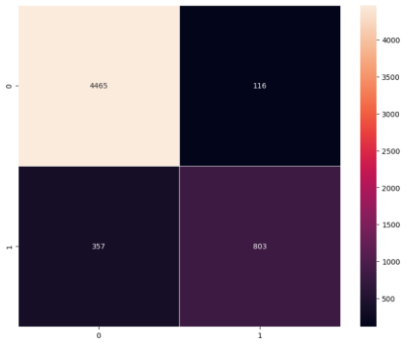
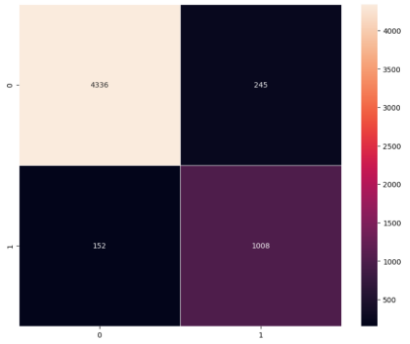
Tahap terakhir setelah proses training model CNN yang cukup lama ini selesai adalah pengujian yang dianalisis dengan menggunakan confusion matrix. Nilai yang diperoleh dari *confusion matrix* akan digunakan untuk perhitungan pada *performance metric* yang meliputi *accuracy*,

Vol. 8(No.1): 1 – 12. Th.2025
p-ISSN: 2615-6431
e-ISSN: 2620-7907

precision, *recall* dan F1-*score*. Dengan metode ini nantinya akan dilihat model CNN mana yang memiliki nilai *performance metric* yang terbaik.

Tabel 2. *Performance Metric*

Model	***Confusion Matrix Heat Map***	***Performance Metric***
VGG19		$accuracy = \frac{4165 + 935}{4165 + 225 + 935 + 416}$ accuracy = 0,8883469 = 88,84% $precision = \frac{4165}{4165 + 416}$ precision = 0,9091901 = 90,92% $recall = \frac{4165}{4165 + 935}$ recall = 0,8166667 = 81,67% $F1 = \frac{2 * 81{,}67 * 90{,}92}{81{,}67 + 90{,}92}$ F1-score = 0,8604712 = 86,05%
ResNet50		$akurasi = \frac{4581 + 0}{4581 + 1160 + 0 + 0}$ akurasi = 0,7979446 = 79,80% $precision = \frac{4581}{4581 + 0}$ precision = 1 = 100% $recall = \frac{4581}{4581 + 0}$ recall = 1 = 100% $F1 = \frac{2 * 100 * 100}{100 + 100}$ F1-score = 100%
DenseNet121		$akurasi = \frac{4285 + 1003}{4285 + 157 + 1003 + 296}$ akurasi = 0,9210938 = 92,11% $precision = \frac{4285}{4285 + 296}$ precision = 0,9353852 = 93,54% $recall = \frac{4285}{4285 + 1003}$ recall = 0,81032526 = 81,03% $F1 = \frac{2 * 81{,}03 * 93{,}54}{81{,}03 + 93{,}54}$ F1-score = 86,84%

MobileNet		$akurasi = \frac{4465 + 803}{4465 + 357 + 803 + 116}$ akurasi = 0,91761017 = 91,76% $precision = \frac{4465}{4465 + 116}$ precision = 0,9746780 = 97,47% $recall = \frac{4465}{4465 + 803}$ recall = 0,847570 = 84,76% $F1 = \frac{2 * 84,76 * 97,47}{84,76 + 97,47}$ F1-score = 90,67%
Xception		$akurasi = \frac{4336 + 1008}{4336 + 152 + 1008 + 245}$ akurasi = 0,93084828 = 93,09% $precision = \frac{4336}{4336 + 245}$ precision = 0,9465182 = 94,65% $recall = \frac{4336}{4336 + 1008}$ recall = 0,8113772 = 81,14% $F1 = \frac{2 * 81,14 * 94,65}{81,14 + 94,65}$ F1-score = 87,38%

Dari ke lima model CNN yang di uji, pada Tabel 2 terlihat bahwa model Xception nilai akurasi adalah yang tertinggi. Dengan model kedua yang memberikan nilai akurasi terbaik adalah DenseNet121. Namun model ResNet50 dengan nilai akurasi paling rendah malah memberikan nilai precision, recall dan F1-score yang paling baik artinya model ini mampu melakukan identifikasi yang terbaik, dapat dilihat pada Tabel 3.

Tabel 3. Perbandingan hasil pengujian

Model	*Accuracy*	*Precision*	*Recall*	*F1-score*
VGG19	88,84%	90,92%	81,67%	86,05%
ResNet50	79,80%	100%	100%	100%
DenseNet121	92,11%	93,54%	81,03%	86,84%
MobileNet	91,76%	97,47%	84,76%	90,67%
Xception	93,09%	94,65%	81,14%	87,38%

Simpulan

Berdasarkan hasil percobaan yang dilakukan dengan dataset yang cukup besar diperoleh hasil bahwa model CNN dengan akurasi terbaik adalah Xception atau DenseNet121 yang hanya selisih 0,98% namun ResNet50 memiliki sedikit anomali dimana hasil *precision*, recall, dan F1-*score* menunjukkan 100%. Pada umumnya jika ketiga nilai tersebut 100% maka nilai *accuracy* juga akan

menunjukkan 100% sehingga seharusnya model CNN terbaik adalah ResNet50 berdasarkan pengujian yang dilakukan.

Mochamad Taufik Ali SA pada publikasi nya berjudul Implementasi *Convolutional Neural Network* (CNN) untuk klasifikasi Ikan Cupang Berbasis *Mobile* yang menggunakan model VGG16 dengan akurasi 94,35% untuk identifikasi Ikan Cupang Big Ear , 86,24% untuk identifikasi Ikan Cupang Nemo, 85,56% untuk identifikasi Ikan Cupang Serit, 78,85% untuk identifikasi Ikan Cupang Halfmoon, dan 74,44% untuk identifikasi Ikan Cupang Bluerim dapat memperoleh hasil akurasi yang lebih baik jika dicoba pada penelitian berikutnya menggunakan model ResNet50, Xception atau DenseNet121 daripada menggunakan model VGG16.

Untuk lebih mendapatkan hasil yang lebih akurat, jumlah hyperparameter epoch yang digunakan dapat ditingkatkan setidaknya 15. Menurut Indra Griha Tofiq Isa pada penelitian yang dilakukan pada percobaan hyperparameter tuning epoch, ditemukan bahwa ketika epoch mencapai angka tersebut terdapat peningkatan secara signifikan pada akurasi (Isa & Junedi, 2022). Dan jika tidak terbatas pada sumberdaya komputasi maka jumlah epoch dapat diuji lebih lanjut untuk mendapatkan titik optimal dimana akurasi paling tinggi akan didapatkan.

Daftar Pustaka

Ali SA, M. T., & Sugiarto, B. (2023). Implementasi Convolutional Neural Network (CNN) untuk Klasifikasi Ikan Cupang Berbasis Mobile. *Digital Transformation Technology (Digitech), 3*(2), 712-723. doi:https://doi.org/10.47709/digitech.v3i2.3245

Anhar, R. A. (2023). Perancangan dan Implementasi Self-Checkout System pada Toko Ritel menggunakan Convolutional Neural Network (CNN). *ELKOMIKA: Jurnal Teknik Energi Elektrik, Teknik Telekomunikasi, & Teknik Elektronika, 11*(2), 466-478. doi:http://dx.doi.org/10.26760/elkomika.v11i2.466

Azmi, K., Defit, S., & Sumijan. (2023). Implementasi Convolutional Neural Network (CNN) Untuk Klasifikasi Batik Tanah Liat Sumatera Barat. *Jurnal Unitek, 16*(1), 28-40. doi:https://doi.org/10.52072/unitek.v16i1.504

Isa, I. G., & Junedi, B. (2022). Hyperparameter Tuning Epoch dalam Meningkatkan Akurasi Data Latih dan Data Validasi pada Citra Pengendara. *Prosiding Seminar Nasional Sains dan Teknologi*, 231-237. doi:http://dx.doi.org/10.36499/psnst.v12i1.6697

Khoiruddin, M., Junaidi, A., & Saputra, W. A. (2022). Klasifikasi Penyakit Daun Padi Menggunakan Convolutional Neural Network. *Journal of Dinda : Data Science, Information Technology, and Data Analytics, 2*(1), 37-45. doi:https://doi.org/10.20895/dinda.v2i1.341

Nurdin, A., Kartika, D. S., & Najaf, A. R. (2024). Klasifikasi Penyakit Daun Tomat Dengan Metode Convolutional Neural Network Menggunakan Arsitektur Inception-V3. *Jurnal Ilmiah Informatika (JIF), 12*(2), 114-119. doi:https://doi.org/10.33884/jif.v12i02.9162

O'Shea, K., & Nash, R. (2015). An Introduction to Convolutional Neural Networks. 1-10. doi:https://doi.org/10.48550/arXiv.1511.08458

Pearson, K. (1904). *Mathematical Contributions to the Theory of Evolution.* London: Dulau and Co.

Putra, D. H., Sulistyowati, & Yudisthiana, V. (2023). Perbandingan Tingkat Akurasi Arsitektur Convolutionsl Neural Network untuk Model Deteksi Penggunaan Masker Secara Otomatis. *Jurnal Ilmu Pengetahuan dan Teknologi (IPTEK), 7*, 25-32. doi:https://doi.org/10.31543/jii.v7i1

Santoso, A., & Ariyanto, G. (2018). Implementasi Deep Learning Berbasis Keras untuk Pengenalan Wajah. *Emitor: Jurnal Teknik Elektro, 18*(1), 15-21. doi:http://dx.doi.org/10.23917/emitor.v18i01.6235

Suryawan, I. T., & Udayana, I. A. (2022). Optimasi Convolutional Neural Network Untuk Deteksi Covid-19 pada X-ray Thorax Berbasis Dropout. *Jurnal Teknologi Informasi dan Ilmu Komputer, 9*(3), 551-558. doi:https://doi.org/10.25126/jtiik.2022935143

Warsito, B. (2009). *Kapita Selekta Statistika Neural Network.* BP UNDIP Semarang.

Wulandari, I., Yasin, H., & Widiharih, T. (2020). Klasifikasi Citra Digital Bumbu Dan Rempah Dengan Algoritma Convolutional Neural Network (CNN). *Jurnal Gaussian, 9*(3), 273-282. doi:https://doi.org/10.14710/j.gauss.9.3.273-282