Jurnal SENOPATI

Sustainability, Ergonomics, Optimization, and Application of Industrial Engineering
Jurnal homepage : ejurnal.itats.ac.id/senopati

Klasifikasi Penyakit Daun Apel Mengunakan Arsitektur CNN Dengan Transfer Learning

Aulia Tegar Rahman[1], Arief Setyanto[2], dan Hanif Al Fatta[3]
Program Studi Magister Teknik Informatika, Fakultas Ilmu kompter, Universitas Amikom Yogyakarta
Jl. Ring Road Utara, Ngringin, Condongcatur, Kec. Depok, Kabupaten Sleman, Daerah Istimewa Yogyakarta 55281

INFORMASI ARTIKEL

Halaman:
42 – 49

Tanggal penyerahan:
10 September 2024

Tanggal diterima:
11 Oktober 2024

Tanggal terbit:
30 Oktober 2024

EMAIL

[1]aulia.tegar.rahman@students.amikom.ac.id
[2]arief_s@amikom.ac.id
[3]hanif.a@amikom.ac.id

ABSTRACT

One of the subtropical agricultural products that can be grown in Indonesia is apples. In apple growing, pest control is one of the key factors for the development of apple growing, as it can affect apple productivity. Rapidly developing technology in detecting or diagnosing plant diseases can simplify the process of classifying plant diseases, especially apple leaf diseases, and support early diagnosis: deep learning. There are deep learning architectures that can be used in image classification, one of which is convolutional neural networks (CNN). CNN architecture with transfer learning method produces acceptable accuracy values, the time required to classify apple leaf diseases is short. The results of classifying apple leaf diseases using VGG16 achieved 99.31% accuracy.

Keywords: *CNN, development, transfer learning*

ABSTRAK

Salah satu hasil produk pertanian *subtropis* yang dapat ditanam di Indonesia adalah apel. Dalam budidaya apel, pengendalian hama dan penyakit merupakan salah satu faktor kunci dalam perkembangan tanaman apel, karena dapat mempengaruhi hasil apel. Salah satu teknologi yang berkembang pesat dalam pendeteksian atau diagnosis penyakit tanaman dapat menyederhanakan proses klasifikasi penyakit tanaman khususnya penyakit daun apel dan membantu dalam *diagnose* dini adalah *deep learning*. Terdapat salah satu arsitektur *deep learning* yang dapat digunakan dalam klasifikasi citra, salah satunya *Convolutional Neural Networks* (CNN). Arsitektur CNN dengan transfer learning yang menghasilkan nilai akurasi yang masih bisa diterima, waktu yang diperlukan pendek pada klasifikasi penyakit daun apel. Hasil dari klasifikasi penyakit daun apel dengan VGG16 mendapatkan akurasi sebesar 99,31 %

Kata kunci: *CNN, development, transfer learning*

PENDAHULUAN

Salah satu hasil produk pertanian subtropis yang dapat ditanam di Indonesia adalah apel. Dalam budidaya apel, pengendalian hama dan penyakit merupakan salah satu faktor kunci dalam perkembangan tanaman apel, karena dapat mempengaruhi hasil apel. Menurut data statistik Badan Pusat Statistik Nasional yang dipublikasikan pada laman website dalam kurun waktu 2016-2020 produksi apel nasional mengalami beberapa penurunan produksi apel, pada tahun 2017 terjadi penurunan produksi apel sebesar 10.781 ton dari tahun 2016 yang jumlah produksi apel sebesar 329781 ton [1]. Oleh karena itu, perlu mengidentifikasi dan mengatasi risiko penurunan produksi apel sedini mungkin dapat mencegah kerugian.

Untuk mengatasi masalah tersebut, perlu identifikasi penyakit tanaman apel agar meminimalkan biaya yang dikeluarkan, dan mengurangi pencemaran lingkungan [2]. Saat ini, ada banyak teknik untuk mendeteksi sifat penyakit tanaman. Salah satu teknologi yang berkembang

pesat dalam pendeteksian atau diagnosis penyakit tanaman dapat menyederhanakan proses klasifikasi penyakit tanaman khususnya penyakit daun apel dan membantu dalam diagnose dini adalah *deep learning* [3]. *Deep learning* merupakan metode dalam kecerdasan buatan (AI) yang dapat menyelesaikan klasifikasi melalui citra [4]. Terdapat salah satu arsitektur *deep learning* yang dapat digunakan dalam klasifikasi citra, salah satunya *Convolutional Neural Networks* (CNN) [5].

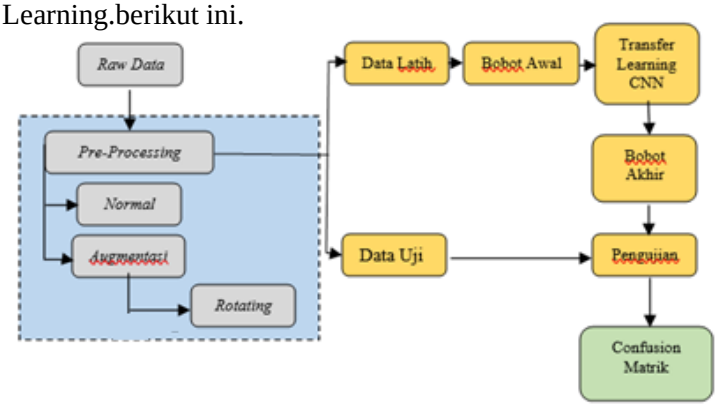
Convolutional Neural Networks (CNN) telah terbukti berguna dalam berbagai *computer vision*, seperti deteksi objek dan identifikasi penyakit dan menghasilkan tingkat akurasi yang signifikan dalam klasifikasi gambar karena mampu menghasilkan fitur sendiri yang terdapat pada citra yang kompleks [6]. Beberapa model CNN yang dapat digunakan dari eksperimen sebelumnya antara lain *GoogLeNet* [7], VGG [8], *ResNet* [9], *DenseNet* [10]. Penerapan arsitektur CNN dengan *transfer learning* yang berfokus pada transfer parameter model CNN yang baru dan akurasi lebih baik pada klasifikasi penyakit daun *apple* maupun lainnya [11]

Berdasarkan penelitian sebelumnya tentang klasifikasi citra penyakit daun apel menggunakan dataset *The Plant Pathology* 2020 dengan *transfer learning* VGG16 [12] dan klasifikasi penyakit daun apel menggunakan model *ResNet* [13], maka usulan penelitian yang akan dilakukan adalah mencari arsitektur CNN dengan *transfer learning* yang menghasilkan nilai akurasi yang masih bisa diterima, waktu yang diperlukan pendek pada klasifikasi penyakit daun apel, sebagai awal dari pendeteksian otomatis klasifikasi penyakit daun pada tanaman apel.

METODE

Metodologi penelitian berisi pembahasan yang akan digunakan dalam penelitian dan metode ya

ng digunakan untuk memecahkan permasalahan termasuk metode analisis dan penjelasan gambarnya. Tahapan yang dilakukan pada penelitian ini sehingga mendapatkan hasil klasifikasi yang optimal meliputi pengambilan data, preprosessing data, augmentasi, transfer learning CNN dan hasil klasifikasi penyakit daun apel sesuai Gambar 1. Blok Diagram Klasifikasi Penyakit Daun Apel dengan Transfer Learning.berikut ini.



Gambar 1. Blok Diagram Kasifikasi Daun Apel.

Berdasarkan Gambar 1. Blok Diagram Klasifikasi Penyakit Daun Apel dengan *Transfer Learning* dijelaskan sebagai berikut:

1. *Raw* Data
   Pengambilan awal adalah citra RGB dengan format .jpg dengan berbagai macam ukuran *pixel*.

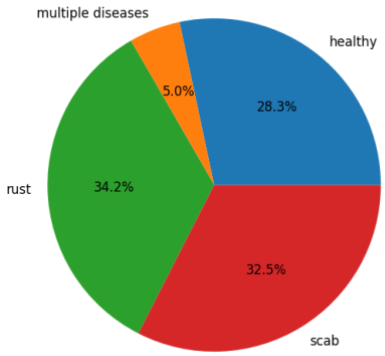
2. *Pre-processing*
   Proses ini terdiri dari 2 (dua) tahap yaitu proses *resizing* dan *cropping* data citra serta anotasi data. Proses *resizing* dan *cropping* dilakukan untuk keperluan data *input*, citra disamakan ukurannya. Sedangkan untuk kebutuhan pengujian proses *pre-processing* perlu pemberian gangguan pada citra, yaitu *rotating*. Proses anotasi data yaitu proses memberi keterangan pada citra berupa informasi.

3. Pelatihan dan Model Pelatihan
   Melakukan proses pelatihan dengan menggunakan dataset citra pelatihan pada penelitian ini. Mendapatkan bobot awal dan model CNN sehingga dihasilkan bobot hasil pelatihan yang nantinya akan digunakan pada proses pengujian.
4. Pengujian dan Model Pengujian
   Proses pengujian terhadap citra pengujian dengan menggunakan bobot hasil pelatihan. Menggunakan citra yang normal dan citra *noise* dengan *rotating*. Dalam penelitian ini menggunakan VGG16 sebagai perbandingan dengan arsitektur cnn dasar.
5. *Confusion Matrix*
   Menghitung hasil kinerja algoritma klasifikasi dengan mencari akurasi (*accuracy*).

HASIL DAN PEMBAHASAN

Pembahasan Data I

Penelitian ini menggunakan VGG16 dikarenakan memiliki tingkat akurasi yang baik. Untuk urutan penelitian dimulai dengan membagi data latih dan data uji sebesar 1821 data dengan 4 klasifikasi antara lain *healthy*, *multiple diseases*, *rust* dan *scab* yang dimana pembagian data setiap klasifikasi dalam penelitian di tampilkan pada Gambar 2. Pembagian data klasifikasi daun apel.



Gambar 2. Pembagian Data Klasifikasi Daun Apel

Berdasarkan Gambar 2. Pembagian data klasifikasi daun apel tahap selanjutnya ialah proses *pre-prosessing* data dengan membagi data validasi sebesar 356 data, data tes sebesar 1821 data dan data latih sebesar 1456 data. Data tersebut dilakukan pengukuran ulang dengan ukuran *pixel* 224 x 224 agar semua data memiliki ukuran yang sama. Selanjutnya data tersebut digunakan untuk proses algoritma CNN dengan model *layer* pada Gambar 3. Model *layer* Algoritma CNN untuk Klasifikasi Penyakit Daun Apel seperti berikut:

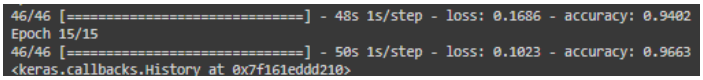
```
model = Sequential([
    layers.Rescaling(1./255.,input_shape=(224,224,3)),
    layers.Conv2D(8,(3,3),activation='relu'),
    layers.MaxPooling2D(2,2),
    layers.Dropout(0.2),
    layers.Conv2D(16,(3,3),activation='relu'),
    layers.MaxPooling2D(2,2),
    layers.Dropout(0.2),
    layers.Conv2D(32,(3,3),activation='relu'),
    layers.MaxPooling2D(2,2),
    layers.Dropout(0.2),
    layers.Flatten(),
    layers.Dense(256,activation='relu'),
    layers.Dense(64,activation='relu'),
    layers.Dense(4,activation='softmax'),
])
model.build()
```

Gambar 3. Model *layer* Algoritma CNN untuk Klasifikasi Penyakit Daun Apel

Pada Gambar 3. Model *layer* Algoritma CNN untuk Klasifikasi Penyakit Daun Apel terdapat 3 *hidden layer* dengan menggunakan *Max Pooling*. *Max pooling* merupakan hasil *output* yang dibagi dari *convolution layer* menjadi beberapa grid kecil lalu mengambil nilai maksimal dari setiap *grid* untuk menyusun matriks. *Max pooling* mengurangi *feature map* hingga 75% dari ukuran asli. Selanjut data validasi di proses dengan *epoch*. *Epoch* merupakan satukali proses pelatihan model dimana seluruh paket pelatihan yang digunakan untuk mengajar model akan diulang sebanyak periode yang telah ditentukan, dalam penelitian ini adalah 15 *epoch*. Pada tahapan *rescaling*, gambar input akan diskalakan dengan membagi tiap piksel dengan nilai 255, sehingga nilainya berkisar antara 0 dan 1. Ukuran input gambar diatur menjadi 224x224 piksel dengan 3 kanal warna (RGB). Conv2D atau *convolutional layer* menunjukkan Layer konvolusi digunakan untuk mendeteksi fitur pada gambar, seperti tepi, tekstur, dan pola lainnya. Layer pertama menggunakan 8 filter berukuran 3x3 dengan fungsi aktivasi ReLU, hal ini diikuti oleh lapisan konvolusi kedua dengan 16 filter, dan lapisan ketiga dengan 32 filter. Fungsi aktivasi ReLU (*Rectified Linear Unit*) membuat output dari layer hanya bernilai positif.

*Layer MaxPooling*2D digunakan untuk mengurangi dimensi spasial dari fitur yang dihasilkan oleh layer konvolusi. Ukuran pooling 2 x 2 berarti bahwa ukuran output akan dikurangi setengahnya di setiap arah. Dropout merupakan teknik regularisasi yang digunakan untuk mencegah model dari *overfitting*. Pada setiap iterasi, sejumlah unit acak (20% dalam hal ini) akan di-drop atau di-nol-kan selama *training*. *Layer flatten* mengubah output dari layer konvolusi dan pooling yang berbentuk matriks dua dimensi menjadi vektor satu dimensi agar bisa di-input ke dalam *layer fully connected*. Sedangkan dense (*fully connected layer*) berfungsi sebagai *classifier*. Terdapat tiga *layer fully connected*, layer pertama memiliki 256 neuron, layer kedua memiliki 64 neuron, keduanya menggunakan fungsi aktivasi ReLU. Layer terakhir memiliki 4 neuron dengan fungsi aktivasi softmax, yang digunakan untuk output klasifikasi multi-kelas (4 kelas penyakit daun apel).

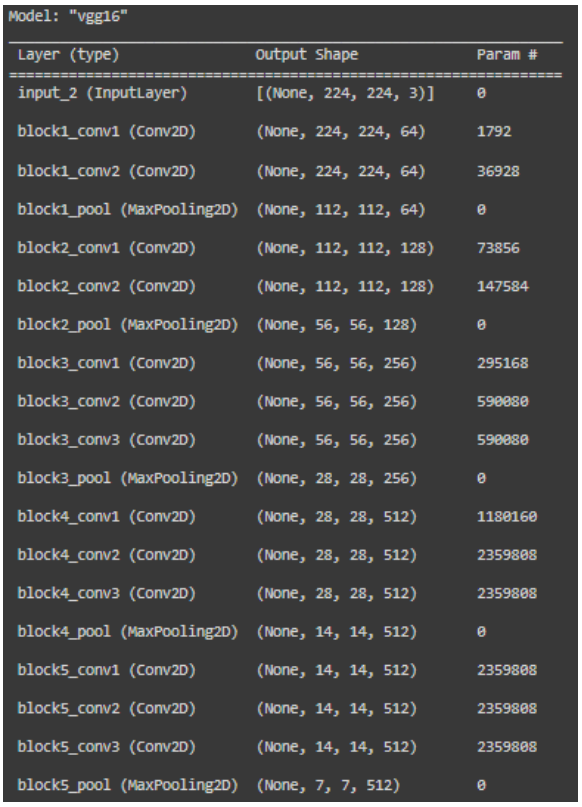
Hasil yang didapatkan dari model tersebut terdapat pada Gambar 4. Hasil 15 *Epoch* Algoritma CNN.



Gambar 4. Hasil 15 *Epoch* Algoritma CNN

Pada Gambar 4. Hasil 15 *Epoch* Algoritma CNN menunjukan bahwa tingkat akurasi model algoritma CNN tersebut sebesar 96,63 %.Model *batch processing* (46/46) memproses 46 *batch* data pada setiap *epoch*. Setiap *batch* terdiri dari sejumlah sampel gambar yang diproses sekaligus untuk mempercepat pelatihan. Waktu pelatihan (48s dan 50s per epoch) menunjukkan bahwa setiap *epoch* memerlukan waktu sekitar 48 hingga 50 detik untuk menyelesaikan seluruh *batch*. Hal ini menggambarkan kecepatan pelatihan model pada dataset yang digunakan. Pada *epoch* pertama, nilai *loss* adalah 0.1686. *Loss* mengukur seberapa jauh prediksi model dari nilai yang diharapkan, semakin kecil nilai *loss*, semakin baik model dalam melakukan prediksi. Setelah 15 epoch, nilai *loss* turun menjadi 0.1023, yang berarti bahwa model telah belajar dengan baik dan semakin mendekati hasil yang benar. Pada *epoch* awal, *accuracy* adalah 94.02%, menunjukkan bahwa pada saat itu, model dapat mengklasifikasikan gambar dengan benar sekitar 94.02% dari data pelatihan. Pada akhir epoch ke-15, *accuracy* meningkat menjadi 96.63%, yang berarti model telah semakin baik dalam melakukan prediksi, dengan tingkat akurasi sebesar 96.63%.

Selanjutnya penelitian ini menggunakan *Transfer Learning* VGG16 dengan model *layer* pada Gambar 5. Algoritma CNN dengan Arsitektur VGG16 untuk Klasifikasi Penyakit Daun Apel seperti berikut:



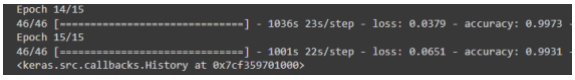
Model: "vgg16"

Layer (type)	Output Shape	Param #
input_2 (InputLayer)	[(None, 224, 224, 3)]	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2359808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0

Gambar 5. Algoritma CNN dengan Arsitektur VGG16 untuk Klasifikasi Penyakit Daun Apel

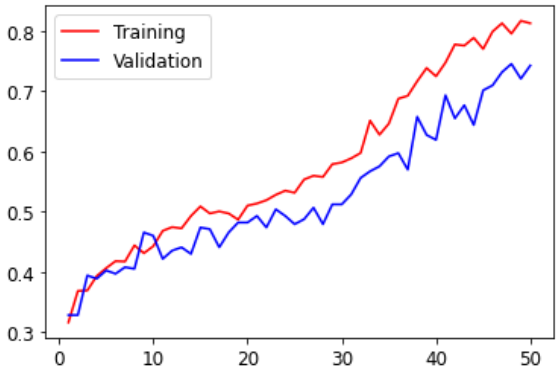
Pada Gambar 5. Algoritma CNN dengan Arsitektur VGG16 untuk Klasifikasi Penyakit Daun Apel menggunakan *Max Pooling*. Sama seperti Algoritma CNN dasar menggunakan 15 *Epoch*. Hasil yang didapatkan dari arsitektur VGG16 terdapat pada Gambar 6. Hasil 15 *Epoch* dengan Algoritma CNN menggunakan arsitektur VGG16. Layer input menerima gambar dengan ukuran 224x224 piksel dan 3 kanal (warna RGB), yang sesuai dengan standar input untuk model VGG16. Arsitektur block konvolusi (Conv2D) terdiri dari beberapa blok konvolusi yang masing-masing berisi dua atau tiga layer konvolusi 2D. Setiap layer konvolusi menggunakan filter untuk mengekstrak fitur dari gambar input. Setiap blok terdiri dari layer konvolusi dengan ukuran filter 3x3 dan stride 1, serta menggunakan fungsi aktivasi ReLU. Ukuran output dari setiap layer konvolusi menurun setelah operasi pooling diterapkan, dan jumlah filter bertambah seiring dengan kedalaman layer untuk menangkap fitur yang lebih kompleks.

Setelah setiap dua atau tiga layer konvolusi, terdapat *layer pooling* (*maxpooling*2D) yang bertujuan untuk mengurangi ukuran dimensi fitur dan menghindari *overfitting*. Pooling ini mengurangi ukuran matriks fitur dengan faktor 2x2. *Layer MaxPooling* ini penting untuk menyederhanakan representasi fitur sambil tetap mempertahankan informasi penting dari Gambar 5. Blok 1 mengandung 2 *layer* konvolusi dengan *output shape* 224 x 224 x 64, diikuti dengan *MaxPooling*2D yang mengurangi ukuran menjadi 112 x 112 x 64. Blok 2 terdiri dari 2 layer konvolusi, output shape-nya 112 x 112 x 128, lalu diikuti MaxPooling2D yang mengurangi ukuran menjadi 56 x 56 x 128. Blok 3 terdiri dari 3 layer konvolusi dengan output shape 56x56x256, dan MaxPooling2D yang mengurangi ukuran menjadi 28 x 28 x 256. Blok 4 terdiri dari 3 layer konvolusi dengan output shape 28 x 28 x 512 dan *MaxPooling*2D yang mengurangi ukuran menjadi 14 x 14 x 512. Blok 5 terdiri dari 3 layer konvolusi dengan *output shape* 14 x 14 x 512, dan *MaxPooling*2D yang mengurangi ukuran menjadi 7 x 7 x 512. Kolom Param (parameter) menunjukkan jumlah parameter yang harus dipelajari pada setiap *layer*. Semakin dalam layer konvolusi, semakin banyak pula parameternya, terutama pada *layer-layer* yang memiliki banyak filter. Jumlah parameter bertambah secara signifikan di setiap blok, mulai dari beberapa ribu hingga jutaan di blok-blok akhir. Hal ini menunjukkan bahwa model ini cukup kompleks dengan banyak parameter yang harus dioptimalkan.



Gambar 6. Hasil 15 *Epoch* dengan Algoritma CNN menggunakan arsitektur VGG16

Pada Gambar 6. Hasil 15 *Epoch* dengan Algoritma CNN menggunakan arsitektur VGG16 menunjukan hasil akurasi sebesar 99,31 %. Selanjutnya Algoritma CNN dilakukan augmentasi dengan *Random Rotation* dan *Random Zoom*. Gambar 6 menunjukkan proses pelatihan model pada epoch ke-14 dan ke-15 dari total 15 epoch. Ini berarti model dilatih sebanyak 15 kali untuk memperbaiki performanya menggunakan data yang sama. Pada epoch ke-14, *loss* adalah 0.0379, dan pada *epoch* ke-15, *loss* sedikit meningkat menjadi 0.0651. *Loss* mengukur kesalahan prediksi model, semakin kecil nilainya, semakin baik model dalam memprediksi. Peningkatan kecil pada loss setelah epoch ke-14 mungkin menunjukkan tanda *overfitting* ringan, di mana model mulai terlalu menyesuaikan dengan data pelatihan. Pada *epoch* ke-14, akurasi model mencapai 99.73%, menunjukkan bahwa pada saat itu, model mampu memprediksi dengan benar hampir 99.73% dari data yang diproses. Pada epoch ke-15, akurasi sedikit menurun menjadi 99.31%, yang tetap merupakan angka yang sangat tinggi, menunjukkan bahwa model masih berkinerja sangat baik meskipun ada sedikit penurunan akurasi. Angka pada *batch* (46/46) menunjukkan bahwa data pelatihan dibagi menjadi 46 *batch*, dan setiap *batch* diproses satu per satu pada setiap *epoch*. Setiap *batch* terdiri dari sejumlah sampel yang diproses dalam satu iterasi untuk mempercepat proses pelatihan. Waktu per *epoch* menunjukkan proses pelatihan untuk setiap epoch memerlukan waktu sekitar 22 hingga 23 detik per batch, menunjukkan kecepatan komputasi model saat dilatih menggunakan arsitektur VGG16. Perbandingan data latih dan data validasi ditunjukkan pada Gambar 7. Hasil Perbandingan data latih dan data validasi, sebagai berikut:



Gambar 7. Hasil Perbandingan data latih dan data validasi

Pada Gambar 7 Hasil Perbandingan data latih dan data validasi menunjukan perbedaan hasil akurasi dengan menggunakan 50 *Epoch* dan penggunaan data augmentasi dengan *Random Rotation* dan *Random Zoom*. Sumbu Y merepresentasikan tingkat akurasi, yaitu seberapa baik model dapat memprediksi dengan benar baik pada data latih maupun data validasi. Semakin tinggi nilai pada sumbu ini, semakin baik akurasinya. Sumbu X merepresentasikan jumlah *epoch*, yaitu berapa kali seluruh *dataset* telah diproses oleh model selama pelatihan. Setiap titik pada sumbu X menunjukkan peningkatan seiring bertambahnya jumlah *epoch*. Garis merah merupakan *training accuracy*: Garis ini menunjukkan bagaimana akurasi model pada data latih meningkat seiring bertambahnya epoch. Akurasi pada data *trainng* terus meningkat secara signifikan hingga mencapai nilai di atas 80%. Sedangkan garis biru merupakan *validation accuracy* yaitu garis ini menunjukkan akurasi pada data validasi, yang juga meningkat seiring bertambahnya epoch, tetapi dengan laju yang lebih lambat daripada data latih. Nilai akurasi untuk data validasi cenderung lebih rendah dibandingkan dengan data *training*.

KESIMPULAN

Berdasarkan hasil penelitian yang telah dilakukan dengan melalui tahap *pre-processing* data dan pembuatan model CNN dengan menggunakan arsitektur VGG16. Maka dapat disimpulkan bahwa penggunaan algoritma CNN menggunakan arsitektur VGG16 menunjukan hasil akurasi sebesar 99,31 %. sangat efektif karena dapat melakukan klasifikasi dan prediksi pada data citra daun apel dengan 4 jenis klasifikasi antara lain *healthy*,*multiple diseases*,*rust* dan *scab*. Penelitian yang telah dilakukan diharapkan dapat menjadi dasar pengembangan tambahan untuk penelitian berikutnya seperti mengumpulkan data lapangan dan beragam.

DAFTAR PUSTAKA

[1] Badan Pusat Statistik, "Badan Pusat Statistik," 2020. [Online]. Available: https://www.bps.go.id/linkTableDinamis/view/id/960. [Accessed: 20-Jul-2022].

[2] S. K, V. R. P, R. P, P. K. M, and P. S, "Apple Leaf Disease Detection using Deep Learning," in *2022 6th International Conference on Computing Methodologies and Communication (ICCMC)*, 2022, pp. 1063–1067.

[3] M. R. D. Septian, A. A. A. Paliwang, M. Cahyanti, and E. R. Swedia, "Penyakit Tanaman Apel Dari Citra Daun Dengan Convolutional Neural Network," *Sebatik*, vol. 24, no. 2, pp. 207–212, 2020.

[4] X. Yuan, C. Yu, B. Liu, H. Sun, and X. Zhu, "CGAN-IRB: A novel data augmentation method for apple leaf diseases," *Proc. - 2021 IEEE 45th Annu. Comput. Software, Appl. Conf. COMPSAC 20*pp. 192–200, Jul. 2021.

[5] Y. Nagaraju, Venkatesh, S. Swetha, and S. Stalin, "Apple and Grape Leaf Diseases Classification using Transfer Learning via Fine-tuned Classifier," in *2020 IEEE International Conference on Machine Learning and Applied Network Technologies (ICMLANT)*, 2020, pp. 1–6.

[6] X. Li and L. Rai, "Apple Leaf Disease Identification and Classification using ResNet Models," in *2020 IEEE 3rd International Conference on Electronic Information and Communication Technology (ICEICT)*, 2020, pp. 738–742.

[7] V. V. Srinidhi, A. Sahay, and K. Deeba, "Plant Pathology Disease Detection in Apple Leaves Using Deep Convolutional Neural Networks : Apple Leaves Disease Detection using EfficientNet and DenseNet," *Proc. - 5th Int. Conf. Comput. Methodol. Commun. ICCMC 2021*, pp. 1119–1127, Apr. 2021.

[8] K. P. Akshai and J. Anitha, "Plant disease classification using deep learning," *2021 3rd Int. Conf. Signal Process. Commun. ICPSC 2021*, pp. 407–411, May 2021.

[9] Li, Zewen, Fan Liu, Wenjie Yang, Shouheng Peng, and Jun Zhou. "A survey of convolutional neural networks: analysis, applications, and prospects." *IEEE transactions on neural networks and learning systems* 33, no. 12. 2021: 6999-7019.

[10] Ketkar, Nikhil, Jojo Moolayil, Nikhil Ketkar, and Jojo Moolayil. "Convolutional neural networks." *Deep learning with Python: learn best practices of deep learning models with PyTorch*. 2021: 197-242.

[11] Khan, Asifullah, Anabia Sohail, Umme Zahoora, and Aqsa Saeed Qureshi. "A survey of the recent architectures of deep convolutional neural networks." *Artificial intelligence review* 53 2020: 5455-5516.

[12] Ghosh, Anirudha, Abu Sufian, Farhana Sultana, Amlan Chakrabarti, and Debashis De. "Fundamental concepts of convolutional neural network." *Recent trends and advances in artificial intelligence and Internet of Things*. 2020: 519-567.

[13] Zhu, Feng, Ruihao Gong, Fengwei Yu, Xianglong Liu, Yanfei Wang, Zhelong Li, Xiuqi Yang, and Junjie Yan. "Towards unified int8 training for convolutional neural network." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1969-1979. 2020.

[14] Soffer, Shelly, Avi Ben-Cohen, Orit Shimon, Michal Marianne Amitai, Hayit Greenspan, and Eyal Klang. "Convolutional neural networks for radiologic images: a radiologist's guide." *Radiology* 290, no. 3. 2019: 590-606.

[15] Dhillon, Anamika, and Gyanendra K. Verma. "Convolutional neural network: a review of models, methodologies and applications to object detection." *Progress in Artificial Intelligence* 9, no. 2. 2020: 85-112.

[16] Oh, Seunghyeok, Jaeho Choi, and Joongheon Kim. "A tutorial on quantum convolutional neural networks (QCNN)." In *2020 International Conference on Information and Communication Technology Convergence (ICTC)*, pp. 236-239. IEEE, 2020.