REPOSITOR, Vol. 5, No. 3, Agustus 2023, Pp. 805-814
ISSN : 2714-7975
E-ISSN : 2716-1382

Penerapan Model EfficientNetV2-B0 pada Benchmark IP102 Dataset untuk Menyelesaikan Masalah Klasifikasi Hama Serangga

Ahmad Hanif Nurfauzi*[1], Yufis Azhar[2], Didih Rizki Chandranegara[3]
[1,2,3]Universitas Muhammadiyah Malang
ahmadhanifnurfauzi@webmail.umm.ac.id*[1], yufis@umm.ac.id^[2], didihrizki@umm.ac.id[3]

Abstrak

Hama serangga merupakan masalah yang sering di hadapi oleh petani. Karena ukurannya yang kecil dan jenis spesiesnya banyak. tak jarang petanipun kesulitan untuk menjaga tanaman mereka dari ancaman hama serangga karena penanganannya tidak memakai satu obat, melainkan dengan mencocokan spesies serangga. Sehingga karena banyaknya obat pembasmi, petanipun bingung obat mana yang tepat. Di dalam penelitian ini, telah di coba penggunaan metode deep learning arsitektur model EfficientNetV2 B0 pada dataset IP102 yang berkarakteristik imbalance dan ada jenis serangga yang identik antara satu dengan yang lain. Penelitian ini bertujuan untuk mengeksplorasi kemungkinan model kecil yang dapat di implementasikan di smartphone atau IOT yang mudah di bawa ke ladang pertanian tanpa tergantung pada internet. Model terbaik yang berhasil dibuat memperoleh akurasi 51% dengan F1-Score 50.14%.

Kata Kunci: *Deep Learning, TensorFlow, Klasifikasi Hama Serangga, EfficientNetV2B0*

Abstract

Insect pests are a problem that is often faced by farmers. Because of its small size and many types of species. it is not uncommon for farmers to have difficulty protecting their crops from the threat of insect pests because the treatment does not use one drug, but by matching the insect species. So, because there are so many pesticides, farmers are confused about which drug is right. In this research, the deep learning method of the EfficientNetV2 B0 model architecture has been tried on the IP102 dataset which has imbalance characteristics and there are insect species that are identical to one another. This study aims to explore the possibility of a small model that can be implemented on a smartphone or IOT that is easy to bring to agricultural fields without depending on the internet. The best model that was successfully made obtained an accuracy of 51% with an F1-Score of 50.14%.

Keywords: *Deep Learning, TensorFlow, Insect Pest Classification, EfficientNetV2B0*

1. Pendahuluan

Hama serangga adalah salah satu penyebab gagal panen tanaman agrikultur. Dalam tahun 2021 saja Federasi Agrikultur Dunia memperkirakan lebih dari 40% produksi dunia hilang karena hama serangga. Setiap tahun penanganannya memakan biaya sebesar 220 miliar dolar AS [1]. Banyaknya spesies serangga yang hidup dalam satu lahan membuat petani kesulitan untuk menjaga tanaman agar tidak mati. Ditambah ukurannya yang kecil sulit terlihat mata menyulitkan petani untuk menentukan obat apa yang efektif untuk menyelamatkan tanaman. Apalagi usia rata-rata pegiat agrikultur yang cenderung tua dan memiliki gangguan penglihatan. Tidak jarang ada berita gagal panen di suatu daerah akibat hama serangga.

Beruntung dengan berkembangnya Machine Learning dan Teknik Computer Vision. Klasifikasi otomatis jenis hama serangga menjadi mungkin diterapkan dan membuat pegiat Deep Learning berlomba-lomba untuk membuat Dataset, Model maupun Pipeline Training yang terbaik untuk menyelesaikan masalah petani dalam mengetahui jenis serangga yang ditemui.

Salah satu dataset yang bisa digunakan untuk training model adalah IP02. Dataset ini di publikasikan oleh Wu dan kawan-kawan dengan bantuan tenaga ahli agrikultur dan menotasikan gambar secara manual. Gambar yang terkumpul dan siap pakai adalah 75 ribu terbagi menjadi 102 kelas. Akan tetapi sebaran gambar yang ada di dataset IP102 tidak seimbang (inbalance

dataset problem). Penulis mencoba untuk membuat model mulai dari metode Support Vector Machine dan K Neighbors Classifier Pada beberapa Teknik filter seperti CH, Gabor, GIST, SIFT, SURF, LCH dan pada arsitektur deep learning seperti Alexnet, GoogleNet, VGGNet, RestNet. Dengan hasil evaluasi terbaik di dapat oleh model RestNet dengan akurasi sebesar 49.4% dan F1-Score 40.1% [2].

Pada penelitian sebelumnya Gaurav Mittal dkk. Membuat pipeline hyperstar yang diperuntukkan untuk memudahkan pengaturan Hyperparameter model. Dengan arsitektur SE-ResNext-50 Model yang mendapatkan akurasi 46.8% [3]. Selanjutnya beberapa peneliti mencoba untuk menemukan cara agar akurasi model TOP-1 bisa di atas 50%. Desain framework XCloud dengan arsitektur model DenseNet-101[4] arsitektur fitur reuse residual network [5] Mengekstrak fitur paling penting menggunakan Saliency dan FusionSum model [6]. Akan tetapi kebanyakan model menggunakan parameter berjumlah besar yang membuat ukuran model ikut besar juga.

Edson Bollis Dkk. Berhasil membuat model EfficientNet dengan parameter kecil [7] dan kedua mertic accuracy dan F1-Score di atas 60%. Dengan tes satu per satu beberapa model deep learning ternama. Mereka berhasil menemukan model terbaik yaitu EfficientNet-B0 dan berhasil menduduki peringkat pertama benchmark P102 selama beberapa bulan. Lalu di Paper kedua Edson Bollis Dkk. Berhasil membuat model EfficeintNet-B0 mengalami peningkatan akurasi di atas 65% [8]. Model dibantu dengan metode bantu Activation Map yang memvisualisasikan gambar agar mudah menilai performa model lalu ditambah memfokuskan fitur pada yang penting saja. Metode ini terbukti meningkatkan akurasi akan tetapi pemrosesan data memakan waktu lebih lama karena harus menggunakan Attention Based Feature Extraction MIL-Guided.

Yang terakhir penelitian dari Lorris Nanni Dkk [9]. Yang menggunakan metode ensamble learning dengan menggabungkan beberapa arsitektur model deep learning ResNet50, GoogleNet, ShuffleNet, MobileNetv2, dan DenseNet201. Mereka berhasil mendapatkan akurasi dan F1-score di atas 70% dan menjadikan model yang dibuat menduduki peringkat satu P102 benchmark. Model ini memiliki kelemahan yaitu parameternya yang besar menjadikan model harus di deploy di server tidak bisa disematkan dalam smartphone lantaran ukuran SDcard tidak bisa menampung model. Sementara itu keberadaan internet di area agrikultur sangat minim karena provider internet lebih fokus di tempat dengan populasi konsumer yang berpotensi mendatangkan keuntungan.

Pembuatan IP102 berlandaskan kurangnya ketersediaan data untuk masalah klasifikasi serangga. Akan tetapi, permasalahan menjadi berbindah ke persebaran data yang tidak seimbang. Walaupun memiliki 102 kelas serangga yang berbeda, banyak serangga memiliki bentuk yang sama terutama yang berukuran kecil. Masalah ini identik dengan klasifikasi sederhana anjing-kucing. Akan tetapi, ada banyak pasangan serangga yang sama. menjadikan permasalahan klasifikasi serangga lebih sulit bahkan hampir sama dengan dataset ImageNet [10].

Oleh karena itu, penelitian ini berusaha untuk menyelesaikan permasalahan klasifikasi yang memiliki imbalance dataset menggunakan arsitektur model EfficientNetV2-B0. Arsitektur ini di pilih karena ratio parameter dan kamampuan presisi model lebih baik dari arsitektur yang lain.

2. Metode Penelitian

Kebanyakan penelitian yang menggunakan dataset IP102 sebagai benchmark mencoba berbagai Teknik mulai dari penyesuaian environment training, Beberapa arsitektur model yang terkenal, hingga ensamble learning. Diantara penelitian tersebut yang memakai EfficientNet-B0 memiliki ratio akurasi dan jumlah parameter model yang paling berimbang diantara yang lain. Penelitian tersebut pun berpotensi untuk bisa diterapkan pada device IOT atau smarphone karena jumlah parameter model yang kecil dan tidak ada risiko ketergantungan dengan internet karena model tidak perlu di deploy di server.

Penelitian ini akan mencoba menerapkan Teknik Image Augmentation, hyperparameter tuning dengan optimizer dan layer model, serta menggunakan EfficientNetV2-B0 untuk mencapai akurasi yang lebih tinggi dan juga parameter yang lebih kecil.

2.1 Pengolahan Data Beserta Asal Usul

Dataset bersumber dari IP02 yang telah dianotasikan oleh para ahli agrikultur terbagi menjadi 102 kelas dengan jumlah total 75.000 buah gambar file dengan kelas gambar terbanyak sebesar 5740 buah gambar dan kelas gambar paling sedikit sebanyak 71 buah gambar. Rata-

rata gambar dalam kelase adalah 971.4 buah dan standar deviasi sebesar 737.4. Terlihat jelas bahwa dataset ini sangat tidak seimbang besar kemungkinan terjadinya penurunan akurasi akibat jumlah gambar yang berbeda antara kelas. Dalam penelitian ini akan diterapkan beberapa metode pembantu yang mungkin bisa meningkatkan akurasi model. Contoh data ada di gambar 1.

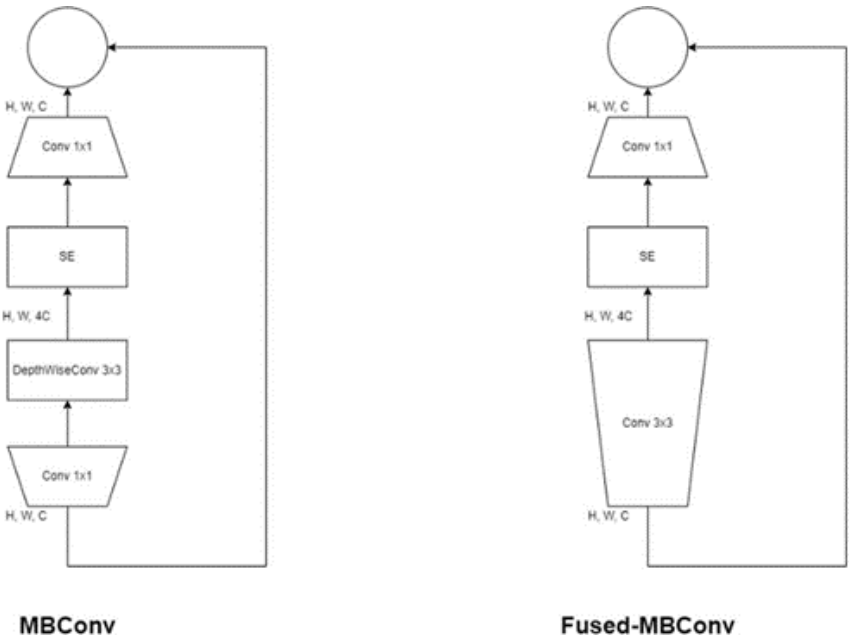
Gambar 1. Contoh Gambar Dataset IP102

2.2 Penerapan Image Augmentation

Penelitian ini akan menerapkan Image Augmentation dengan harapan bertambahnya gambar pada kelas yang memiliki data yang sedikit sehingga didapatkan data yang cukup untuk membuat performa model menjadi lebih baik [11][12]. Penelitian ini akan mencoba menggunakan beberapa tipe augmentasi seperti rescale, horizontal flip, rotation range, vertical flip, brightness range, zoom range, sheer range, width shift range, dan height shift range untuk memperkaya data dan diharapkan dapat mengingkatkan akurasi.

2.3 Arsitektur Model

Penelitian akan menggunakan dasar model EfficientNetV2-B0[13] yang di klaim lebih baik dari EfficientNet-B0 [14]. Penambahan Fused-MBConv pada EfficientNetV2 diharapkan bisa mengurangi kelemahan dari EfficientNet-B0 layer MBConv yang lama saat tahapan awal, tetapi efektif di tahapan akhir. Selain itu luasnya layer MBConv mengakibatkan jumlah parameter dan FLOP lebih kecil dari layer deep learning yang lain, tetapi terkadang layer MBConv tidak bisa menggunakan accelerators terkini. Perbedaan bisa dilihat di Gambar 2 dan Tabel 1.



Gambar 2. Perbedaan MBConv dan Fused-MBConv

Model akan dimodifikasi dengan Tujuan untuk menambah akurasi dengan manambahkan layer tambahan. Layer yang bisa ditambahkan diantaranya:

a) Flatten
Berguna untuk mengubah input multi-dimensi menjadi satu dimensi [15].
b) Pooling
Berguna untuk mengurangi dimensi dari feature map [16].
c) Dropout
Secara acak mengubah nilai input menjadi 0 sesuai dengan frekuensi yang diatur [17].
d) Batchnormalization
Metode yang digunakan untuk mempercepat proses pelatihan model dengan menerapkannormalisasi input layer dengan teknk rescale dan recenter [18].
e) Dense
Sebuah *neural network* sederhana yang dimana setiap *neuron* menerima input dari layersebelumnya [19].

2.4 Pemilihan Optimizer

Optimizer berperan besar dalam pelatihan model machine learning. Ada empat optimizer yang cocok untuk menyelesaikan problem klasifikasi. Adam mudah diterapkan dalam machine learning model. Memiliki komputasi yang efisien, memakan sedikit memori, sangat cocok untuk data/parameter yang banyak. Hyperparameter memiliki interpretasi intuitif dan hanya membutuhkan sedikit penyetelan [20].

2.5 Penerapan Evaluasi

Pengevaluasian model dilakukan dengan cara menambahk metric accuracy dan f1-score ke dalam compile model agar Ketika mengevaluasi ke data test tidak perlu menggunakan framework lain. Rumusan akurasi dan F1-Score ada di Persamaan 1, Persamaan 2, Persamaan 3, dan Persamaan 4.

$$Acc = \frac{Tebakan\ Benar}{Total\ Gambar\ yang\ ingin\ ditebak} \quad (1)$$

$$Presisi = \frac{TP}{TP + FP} \quad (2)$$

$$Recall = \frac{TP}{TP + FP} \quad (3)$$

$$F1Score = 2 \times \frac{Presisi \times Recall}{Presisi + Recall} \quad (4)$$

2.6 Penggunaan Callback

Karena proses train yang lama. Diperlukan callback yang berguna untuk menyimpan weight dan penurunan learning rate optimizer agar ditemukan global minima lebih cepat.

2.7 Tuning Augmentation, Hyperparameter, dan komposisi Layer

Penelitian ini akan menelusuri penggunaan augmentasi, nilai hyperparameter, dan susunan layer mana yang menghasilkan nilai akurasi tertinggi [21]. Hal-hal yang diperhatikan diantaranya:

A. Augmentation
1) Kombinasi tipe augmentasi yang digunakan.
2) Besar gambar (target size)
3) Batch size

B. Hyperparameter
1) Besar learning rate optimizer
2) Besar nilai dropout

C. Komposisi Layer
1) Pemakaian kombinasi layer, flatten, dropout, batchnormalization, dan dense
2) Penggunaan callback untuk penurunan akurasi, dan imagenet weight.

Tabel 1. Perbedaan EfficientNet V1 dan V2

Tahapan	Asitektur Model					
	EfficientNet-B0			EfficientNetV2-B0		
	Operator	Channel	Layers	Operator	Channel	Layers
1	Conv3x3	32	1	Conv 3x3	24	1
2	MBConv1, k3x3	16	1	Fused-MBConv1, k3x3	24	2
3	MBConv6, k3x3	24	2	Fused-MBConv4, k3x3	48	4
4	MBConv6, k5x5	40	2	Fused-MBConv4, k3x3	64	4
5	MBConv6, k3x3	80	3	MBConv4, k3x3, SE0.25	128	6
6	MBConv6, k5x5	112	3	MBConv6, k3x3, SE0.25	160	9
7	MBConv6, k5x5	192	4	MBConv6, k3x3, SE0.25	256	15
8	MBConv6, k3x3	320	1	Conv1x1, Pooling, FC	1280	1
9	Conv1x1, Pooling, FC	1280	1			

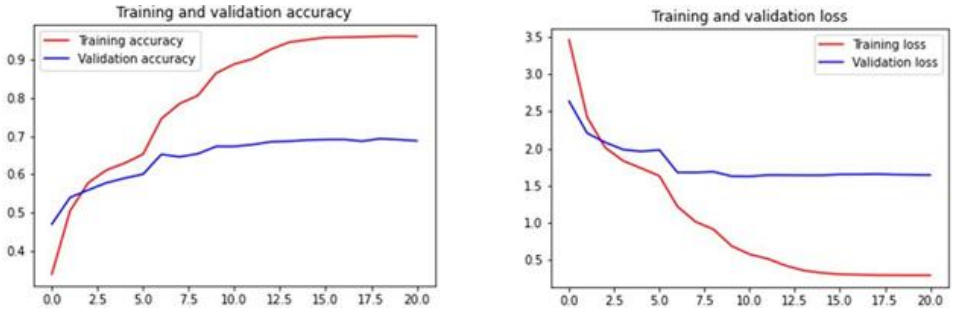
3. Hasil Penelitian dan Pembahasan

Penelitian dilakukan dengan bantuan Google Colab Pro dan dilakukan hyperparameter tuning manual dikarenakan lamanya proses training [22][23]. Berikut hasil pengamatan selama penelitian terhadap parameter tertentu:

3.1 Uji Coba Penggunaan Augmentasi yang Berbeda

3.1.1 Uji Coba yang Berfokus pada Batch Size

Pertama-tama penilitian dimulai dengan membuat model dengan image data generator rescale 1/255 dan sheer range 50. Kemudian mengatur target size pada EfficientNetv2B0 input sebesar 224,224. Batch size sebesar 64. Imagenet weight untuk membantu meringankan waktu train. Max pooling 2D. tambahan layer batchnormalization, flattern, dropout 0.3, dan dense 1000. Adam digunakan sebagai optimizer dengan learning rate sebesar 0.001 juga dibantu dengan callback reduce learning rate on plateu sebesar 0.5. Proses train akan berhenti jika nilai loss tidak berkurang selama 10 epoch. Seperti yang tertera pada Gambar 3. Hasilnya model mendapatkan akurasi train sebesar 0.9610, validasi akurasi sebesar 0.6875, tes akurasi sebesar 0.0843. Model mengalami overfit dan gagal untuk belajar fitur terpenting dari dataset IP102. Selanjutnya dilakukan uji coba mengubah batch size yang tercatat di Tabel 2.



Gambar 3. Grafik Training Uji Coba 1

Tabel 2. Uji Coba Mengubah Batch Size pada Akurasi Model di Bawah 20%

Batch Size	Train accuracy	Validasi akurasi	Test akurasi
32	0.9283	0.6767	0.1401
16	0.8476	0.6423	0.1388
128	0.9572	0.6969	0.1250

Selanjutnya uji coba diteruskan dengan mengubah *batch size* menjadi 32. Lalu penelitian dilanjutkan dengan mencoba untuk mengubah layer tambahan setelah EfficientNetV2 B0 dengan *rinician include top, batchnormalization, flatten, dropout* 0.3, *dense layer* 500, *dropout* 0.2 *dense layer* 500. Hasilnya akurasi *train* mendapat nilai 0.8502, akurasi validasi 0.6536. akurasi test meningkat ke 0.1831.

Berikutnya uji coba dilakukan dengan mengubah *pooling* menjadi *average*. Mengubah 2*dense layer* dengan jumlah 4096 dan 2 *dropout* 0.5, hasilnya akurasi masih di bawah 0.1831. Kemudian *batch size* diubah dengan rinician train 128, 32, dan val 16 dan hasilnya tetap belum bisa menjadi lebih baik. *Include top* tidak dipakai lagi karena tidak berdampak pada proses train model.

Penelitan dilanjutkan dengan mencoba untuk memperbesar *target size image data generator* menjadi 448,448 hasilnya mendapat *train* akurasi 0.9006, validasi akurasisebesar 0.6622. dan test akurasi 0.2588.

3.1.2 Uji Coba dengan Target Size

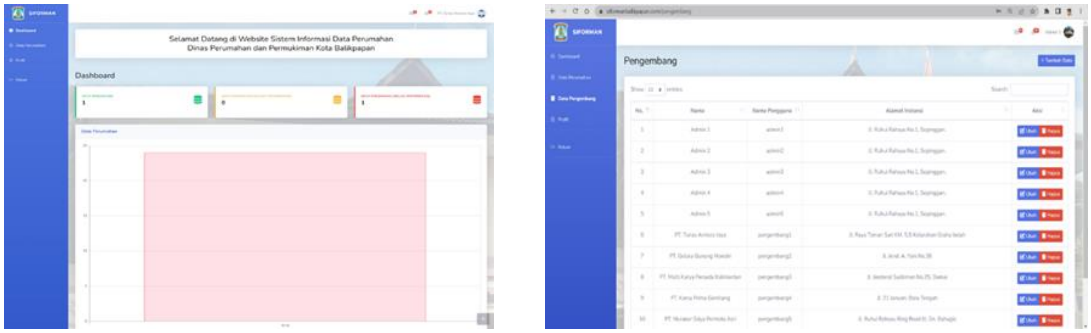
Penelitian dilanjutkan dengan berfokus pada fitur image data generator. Dengan menambahkan horizontal_flip = True, rotation_range = 90, vertical_flip = True, brightness_range = [0.2,1.0], zoom_range = [0.1, 2], Memperbesar target size sebesar 896,896 dan mengoptimalkan batch size karena keterbatasan ram GPU, menjadi 4, Menghilangkan dense layer dan dropout layer. Model berhasil mendapatkan train akurasi sebesar 0.5098, validasi akurasi 0.5417 dan test akurasi sebesar 0.3040.

3.1.3 Implementalsi Prototype

Selanjutnya mengecek efek data augmentasi. Karena pooling max terbukti meningkatkan performa. Uji coba selanjutnya menggunakan max pooling sebagai default. Hasil uji coba tercatat pada tabel 3.

Tabel 3. Uji Coba Efek Image Data Generator

Image data generator	Train accuracy	Validation accuracy	Test accuracy
Sheer range 50	0.8164	0.6282	0.2400
Rotation range 90	0.8348	0.6414	0.2371
Zoom range 0.1 2	0.6731	0.5341	0.3101
Brightness range 0.2 1	0.9435	0.4695	0.2394
Horizontal flip true	0.9218	0.5145	0.2404
Vertical flip true	0.8791	0.5196	0.2542
Width shift range -4 4	0.9491	0.6193	0.2854
Height shift range -4 4	0.9572	0.6035	0.2182



Gambar 4. Tampilan Interface Pengembang dan admin

3.2 Uji Coba pada Susunan Layer Model

Karena banyaknya parameter yang diubah-ubah penelitian difokuskan untuk mencari tahu parameter mana yang memiliki dampak terbesar terhadap performa akurasi tes model. Yang

pertama mengecek efek pooling pada arsitektur model EfficentNetV2B0. Dalam tes ini image data generator tidak digunakan sama sekali. Target size pada image data generator ditetapkan menjadi 224,224. Menggunakan ImageNet weight. Optimizer adam dengan learning rate 0.001 dan hanya menggunakan flatten sebagai transformasi model ke output sebanyak 102 kelas. Hasil uji coba tercatat di Tabel 4.

Tabel 4. Uji Coba Efek Pooling pada Model

Pooling	Train accuracy	Validation accuracy	Test accuracy
avg	0.9842	0.6289	0.2172
max	0.8943	0.5008	0.2684

Selanjutnya menguji layer tambahan dengan tanpa image data generator. Target size image data generator 224,224. Batch size 8. EfficientNetV2B0 dengan imagenet weight, Max pooling, Optimizer adam 0.001. Hasil uji coba tercatat pada Tabel 5.

Tabel 5. Uji Coba Layer Tambahan

Layer tambahan	Train accuracy	Validation accuracy	Test accuracy
Flatten, dense 102, dropout 0.1 dense 102, dropout 0.1	0.7771	0.5247	0.2674
Flatten, dense 102, dense 102.	0.8047	0.5283	0.2622

3.3 Uji Coba Batch Size dan Learning Rate

Menguji batch size dan learning rate tanpa image data generator. Nilai target size ditetapkan menjadi 224,224. Menggunakan EfficientNetV2B0 dengan imagenet weight, Max pooling, Layer flatten untuk mengubah dimensi EfficientNetv2B0 dan output sebanyak 102 kelas. Hasil uji coba tercatat pada Tabel 6.

Tabel 6. Uji Coba Batch Size dan Learning Rate

Batch Size	Learning rate	Train accuracy	Validation accuracy	Test accuracy
128	0.001	0.9998	0.6914	0.1081
8	0.0001	0.8800	0.4514	0.1344
8	0.01	0.2463	0.0782	0.0383
4	0.001	0.9463	0.5454	0.3929
2	0.00001	0.8249	0.5248	0.3976
1	0.001	0.2227	0.2190	0.2054
1	0.001	0.2227	0.2190	0.2054

3.4 Kombinasi Parameter Terbaik

1) Dari data yang di dapat, penelitian menggabungkan hasil terbaik dari semua sesi dengan tanpa parameter modifikasi *image data generator*. *Target size* (448,448), *Batch size* 4, EfficientnetV2 B0 dengan *imagenet weight*, *Max pooling*, *Optimizer* adam dengan *learning rate* 0.0001. Model berhasil mendapatkan *train* akurasi 0.9965. validasi akurasi 0.6717 dan test akurasi 0.4360.
2) Ketika *learning rate* diturunkan menjadi 0.00001. *train* akurasi berubah menjadi 0.7340. validasi akurasi berubah menjadi 0.6709 dan tes akurasi berubah sebesar 0.4899.
3) Uji coba dilanjutkan dengan menambahkan *image data generator zoom range* [0.1, 2]. Model mendapatkan *train* akurasi sebesar 0.7805, validasi akurasi sebesar 0.6895 dan test akurasisebesar 0.5134.

3.5 Perbandingan

Berdasarkan data pada tabel 7, model berhasil mendapatkan akurasi dan f1-score lebih baik dari paper pertama. Dari beberapa sekenario yang telah di coba. Terdapat beberapa hasil yang dapat membantu menjawab rumusan masalah penelitian yang telah dibuat. Rumusan masalah pertama tentang imbalance dataset. Ketidak seimbangan komposisi data memang bisa mempengaruhi akurasi model. Terutama semakin banyak kelas yang dipelajari dan hampir sama gambar antara spesies satu dengan spesies yang lain. Problem ini menjadi tantangan tersendiri pada penelitian machine learning dibidang ini.

Arsitektur model yang terbaik adalah EfficientNetV2N0 dengan tambahan layer flatten dan dense output 102. Penambahan layer yang lain akan berdampak dengan penurunan model, terutama panambahan layer normalization dan penggunaan average pooling. Kedua layer tersebut membuat nilai fitur baru sehingga ketika diuji di dataset test model akan memiliki nilai fitur baru. Akan tetapi, ketika menggunakan max pooling model mempelajari nilai tertinggi dari fitur yang ada di data train.

Hyperparameter yang paling bepengaruh adalah image augmentasi zoom range, target size, batch size. Karena kebanyakan kelas hampir sama satu sama lain. Maka augmentasi yang bisa membuat fitur lebih bervariasi adalah zoom range dan ukuran target size gambar. Penggunaan rescale membuat model mempelajari fitur baru yang tidak begitu penting, membuat akurasi pada data test turun drastis. Untuk argument flip tidak begitu meningkatkan akurasi secara signifikan. Argumen rotation range, brightness range, sheer range, width shift range, height shift range layak untuk di uji lebih dalam lagi karena penelitian ini tidak sempat meneliti lebih dalam tentang argument-argumen tersebut. kekurangan dalam pemakaian target size yang besar dan batch size yang kecil adalah proses train yang memakan waktu lama.

Untuk learning rate yang tepat tergolong kecil yaitu sebesar 0.00001. perubahan learning rate sangat sensitif karena apabila terlalu besar atau kecil nilainya. Akurasi model akan menurun. Penelitian ini pun terbatas perihal optimizer karena banyaknya hal-hal yang diperhatikan.

Tabel 7. Perbandingan dengan Penelitian Terdahulu

Metode	*Accuracy*	*F1-Score*
IP102 restnet 50	0.49.4	0.41
Efficientnetv2B0+imagenet weight, image data generator zoom range 0. 2. Target size 448,448, max pooling. Adam learning rate 0.00001 tambahan flatten dan output sebanyak 102 kelas	0.5134	0.5014

4. Kesimpulan

Model bisa mendapatkan akurasi 51% dengan bantuan ImageNet weight, image data generator, zoom range, target size (448,448), batch size 4, max pooling, optimizer adam dengan learning rate 0.00001, Dataset IP102 sangat sensitive terhadap fitur dan tidak disarankan untuk menghindari segala bentuk normalisasi (batch normalization, rescale, average pooling) karena fitur yang terpenting menjadi tidak dipelajari oleh model. Penggunaan target size yang lebih besar dari input dan batch size yang lebih kecil membantu model untuk mempelajari fitur yang tepat. Akan tetapi, model menjadi lebih lama dalam proses pelatihan.

Untuk penelitian selanjutnya di sarankan untuk menggunakan optimizer yang lain, arsitektur EfficientNetV2 yang lebih tinggi, dan mendalami image data generator untuk mencapai akurasi yang lebih baik dari penelitian ini.

Daftar Notasi

TP : True Positive
FP : False Positive
TN : True Negative
FN : False Negative

Referensi

[1] "FAO - News Article: Climate change fans spread of pests and threatens plants and crops, new FAO study." https://www.fao.org/news/story/en/item/1402920/icode (accessed Dec. 04, 2022).

[2] X. Wu, C. Zhan, Y.-K. Lai, M.-M. Cheng, and J. Yang, "IP102: A Large-Scale Benchmark Dataset for Insect Pest Recognition." pp. 8787–8796, 2019. Accessed: Dec. 04, 2022. [Online]. Available: https://github.com/

[3] G. Mittal, C. Liu, N. Karianakis, V. Fragoso, M. Chen, and Y. Fu, "HyperSTAR: Task-Aware Hyperparameters for Deep Networks," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 8733–8742, May 2020, doi: 10.48550/arxiv.2005.10524.

[4] L. Xu and Y. Wang, "XCloud: Design and Implementation of AI Cloud Platform with RESTful API Service," Dec. 2019, doi: 10.48550/arxiv.1912.10344.

[5] F. Ren, W. Liu, and G. Wu, "Feature reuse residual networks for insect pest recognition," *IEEE Access*, vol. 7, pp. 122758–122768, 2019, doi: 10.1109/ACCESS.2019.2938194.

[6] L. Nanni, G. Maguolo, and F. Pancino, "Insect pest image detection and recognition based on bio- inspired methods," *Ecol Inform*, vol. 57, p. 101089, May 2020, doi: 10.1016/J.ECOINF.2020.101089.

[7] E. Bollis, H. Pedrini, and S. Avila, "Weakly Supervised Learning Guided by Activation Mapping Applied to a Novel Citrus Pest Benchmark," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, vol. 2020-June, pp. 310–319, Apr. 2020, doi: 10.48550/arxiv.2004.11252.

[8] E. Bollis, H. Maia, H. Pedrini, and S. Avila, "Weakly supervised attention-based models using activation maps for citrus mite and insect pest classification," *Comput Electron Agric*, vol. 195, p. 106839, Apr. 2022, doi: 10.1016/J.COMPAG.2022.106839.

[9] L. Nanni, A. Manfè, G. Maguolo, A. Lumini, and S. Brahnam, "High performing ensemble of convolutional neural networks for insect pest image detection," *Ecol Inform*, vol. 67, p. 101515, Mar. 2022, doi: 10.1016/J.ECOINF.2021.101515.

[10] J. Deng, W. Dong, R. Socher, L.-J. Li, Kai Li, and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database," pp. 248–255, Mar. 2010, doi: 10.1109/CVPR.2009.5206848.

[11] L. Perez and J. Wang, "The Effectiveness of Data Augmentation in Image Classification using Deep Learning," Dec. 2017, doi: 10.48550/arxiv.1712.04621.

[12] C. Shorten and T. M. Khoshgoftaar, "A survey on Image Data Augmentation for Deep Learning," *J Big Data*, vol. 6, no. 1, pp. 1–48, Dec. 2019, doi: 10.1186/S40537-019-0197-0/FIGURES/33.

[13] M. Tan and Q. v. Le, "EfficientNetV2: Smaller Models and Faster Training," Apr. 2021, doi: 10.48550/arxiv.2104.00298.

[14] M. Tan and Q. v. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," *36th International Conference on Machine Learning, ICML 2019*, vol. 2019-June, pp. 10691–10700, May 2019, doi: 10.48550/arxiv.1905.11946.

[15] J. Jin, A. Dundar, and E. Culurciello, "Flattened Convolutional Neural Networks for Feedforward Acceleration," *3rd International Conference on Learning Representations, ICLR 2015 - Workshop Track Proceedings*, Dec. 2014, doi: 10.48550/arxiv.1412.5474.

[16] H. Gholamalinezhad and H. Khosravi, "Pooling Methods in Deep Neural Networks, a Review," Sep. 2020, doi: 10.48550/arxiv.2009.07485.

[17] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014, Accessed: Dec. 09, 2022. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html

[18] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," *32nd International Conference on Machine Learning, ICML 2015*, vol. 1, pp. 448–456, Feb. 2015, doi: 10.48550/arxiv.1502.03167.

[19] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely Connected Convolutional Networks," *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-January, pp. 2261–2269, Aug. 2016, doi: 10.48550/arxiv.1608.06993.

[20] D. P. Kingma and J. L. Ba, “Adam: A Method for Stochastic Optimization,” *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, Dec. 2014, doi: 10.48550/arxiv.1412.6980.
[21] M. Feurer and F. Hutter, “Hyperparameter Optimization,” pp. 3–33, 2019, doi: 10.1007/978-3-030-05318-5_1.
[22] M. v. Valueva, N. N. Nagornov, P. A. Lyakhov, G. v. Valuev, and N. I. Chervyakov, “Application of the residue number system to reduce hardware costs of the convolutional neural network implementation,” *Math Comput Simul*, vol. 177, pp. 232–243, Nov. 2020, doi: 10.1016/J.MATCOM.2020.04.031.
[23] K. Itoh, W. Zhang, Y. Ichioka, and J. Tanida, “Parallel distributed processing model with local space-invariant interconnections and its optical architecture,” *Applied Optics, Vol. 29, Issue 32, pp. 4790-4797*, vol. 29, no. 32, pp. 4790–4797, Nov. 1990, doi: 10.1364/AO.29.004790