Jurnal Teknik Informatika (JUTIF)
Vol. 6, No. 1, February 2025, pp. 369-386

DOI: https://doi.org/10.52436/1.jutif.2025.6.1.4144
p-ISSN: 2723-3863
e-ISSN: 2723-3871

ENHANCED SPEED AND ACCURACY IN COCOA FRUIT DISEASE IDENTIFICATION USING THE INCEPTION-RESNET CONVOLUTIONAL NEURAL NETWORK (CNN) ALGORITHM

Adi Novanto[1], Yogiek Indra Kurniawan[2], Dadang Iskandar*[3]

[1,2,3]Informatics, Engineering Faculty, Universitas Jenderal Soedirman, Indonesia
Email: [1]adinovanto07@gmail.com, [2]yogiek@unsoed.ac.id, [3]dadang.iskandar@unsoed.ac.id

(Article received: November 23, 2024; Revision: February 17, 2025; published: February 19, 2025)

Abstract

The increase in world cocoa consumption is not accompanied by an increase in production, causing a problem of supply shortages in the world. One of the causes is a disease that attacks cocoa fruit which can result in unproductive plants and the spread of an epidemic. Prevention that can be done is the identification of cocoa fruit diseases. Identification of cocoa fruit diseases requires knowledge by farmers, resulting in misidentification. In addition, other factors can arise such as the number of farmers who check, the area of cocoa fruit plantations, and the urgency of identification. To help overcome these problems, the Convolutional Neural Network (CNN) model was developed. Several previous studies have used a combination of methods and architectures. Ciro Rodriguez's research was carried out by combining the Histograms of Oriented Gradient (HoG), Local Binary Pattern and Support Vector Machine (SVM) architecture techniques which produced an accuracy level of 75%. While Maulidiah h's research used the Mask Region-based algorithm producing an F1-score of 0.907. And at a conference the MobileNet V2 algorithm was used with a confidence level of 80%. To expand the field of knowledge, the Inception and ResNet architectures were used. The data used were images obtained from Davao City, Philippines. The model obtained from the analyzed dataset obtained the best results of 0.99, a specificity value of 0.99, and an F1-score value of 0.99. The model configuration used was a learning rate value of 0.0001, an RMSProp optimization function, an initialization function (x) He uniform, an initialization function (y) Glorot normal, and a batch size of 32.

Keywords: *cacao fruit, convolutional neural network*, *deep learning, digital image processing, inception, resnet.*

PENINGKATAN KECEPATAN DAN AKURASI IDENTIFIKASI PENYAKIT BUAH KAKAO MENGGUNAKAN ALGORITMA CONVOLUTIONAL NEURAL NETWORK (CNN) INCEPTION-RESNET

Abstrak

Peningkatan konsumsi buah kakao dunia tidak disertai peningkatan produksi menimbulkan permasalahan kekurangan pasokan di dunia. Salah satu penyebabnya adalah penyakit yang menyerang buah kakao yang dapat mengakibatkan tumbuhan tidak produktif hingga penyebaran wabah. Pencegahan yang dapat dilakukan adalah identifikasi penyakit buah kakao. Identifikasi penyakit buah kakao memerlukan pengetahuan oleh para petani, sehingga menyebabkan kesalahan identifikasi. Selain itu, dapat muncul faktor lain seperti jumlah petani yang memeriksa, luas kebun buah kakao, hingga urgensi identifikasi. Untuk membantu mengatasi permasalahan tersebut, dikembangkan model *Convolutional Neural Network* (CNN). Beberapa penelitian sebelumnya telah menggunakan kombinasi metode dan arsitektur. Penelitian Ciro Rodriguez dilakukan penggabungan teknik *Histograms of Oriented Gradient* (HoG), *Local Binary Pattern* dan arsitektur *Support Vector Machine* (SVM) yang menghasilkan tingkat akurasi 75%. Sedangkan penelitian Maulidiah h menggunakan algoritma *Mask Region-based* menghasilkan F1-score 0.907. Dan sudah pada sebuah konferensi digunakan algoritma *MobileNet V2* dengan hasil *conffidence level* sebesar 80%. Sehingga untuk memperluas bidang pengetahuan digunakan arsitektur *Inception* dan *ResNet*. Data yang digunakan berupa citra yang diperoleh dari Kota Davao, Filipina. Model yang diperoleh dari *dataset* yang dianalisis mendapatkan hasil terbaik 0.99, nilai *specificity* sebesar 0.99, dan nilai *F1-score* sebesar 0.99. Konfigurasi model yang digunakan yaitu nilai learning rate 0.0001, fungsi optimasi RMSProp, fungsi inisialisasi (x) He uniform, fungsi inisialisasi (y) Glorot normal, serta dengan ukuran *batch* 32.

Kata kunci: buah kakao, *convolutional neural network*, *deep learning*, pengolahan citra digital, *inception, resnet.*

1. PENDAHULUAN

Tanaman kakao (Theobroma cacao L.) merupakan pohon tropis penghasil biji kakao dibudidayakan oleh petani di dataran rendah tropis Amerika Tengah dan Selatan, Afrika Barat, dan Asia Tenggara [1]. Menurut data statistik, komoditas kakao di Indonesia tahun (2016-2020) berkontrobusi terhadap Produk Domestik Bruto (PDB) Negara Indonesia mengalami kenaikan dari 3,57 persen pada 2016 menjadi 3,64 persen pada tahun 2020. Data tersebut didukung dengan nilai ekspor dalam kakao pada tahun 2016 sebesar 330.030 ton dan 377.849 ton pada 2020. Kenaikan angkat tersebut menunjukan bahwa Indonesia memiliki daya saing yang saat ini berada pada peringkat ketiga dunia setelah Ghana dan Pantai Gading [2].

Meskipun adanya peningkatan konsumsi kakao di dunia, produktivitas kakao mengalami penurunan di beberapa negara produsen kakao karena beberapa faktor penyebab seperti perkebunan yang menua, tanah yang terdegradasi, hama dan penyakit, dan lain-lain. Beberapa jenis penyakit yang timbul pada kakao diantaranya adalah *frosty pod rot* (Moniliophthora roreri), *black pod* (Phytophthora spp) *witches' broom* (Cripinellis perniciosa), *Cacao swollen shoot virus*, *Vascular streak dieback* (Ceratobasidium theobromae), dan *Cocoa pod borer* (Conopomorpha cremerella) [3]. Salah satu metode pencegahan yang dapat dilakukan adalah adalah identifikasi buah kakao.

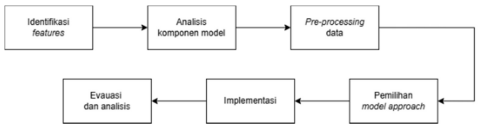
Pada penelitian oleh Ciro Rodriguez, menggunakan penggabungan teknik *Histograms of Oriented Gradient* (HoG) dan *Local Binary Pattern* kemudian diklasifikasikan dengan algoritma *Support Vector Machine* (SVM) menghasilkan akurasi 75% [4]. Ada juga penelitian oleh Nurul Maulidiah menggunakan algoritma *Mask Region-based* menghasilkan F1-score 0.907 [5]. Deteksi penyakit pada kakao juga dilakukan dalam konferensi dengan algoritma *MobileNet* V2 dengan hasil *conffidence level* sebesar 80% [6]. Deteksi penyakit juga diterapkan pada tanaman padi menggunakan *deep learning* dengan modifikasi algoritma VGG19 [7]. Selain itu pada penelitian Yunfeng Chen yang menggunakan *Inception ResNet* memperoleh hasil perolehan akurasi 98.66%. Hasil pengujian tersebut lebih baik dibandingkan dengan penggunaan algoritma lain [8]. Pernyataan ini didukung oleh [9] bahwa penggunakan CNN khususnya dengan pemanfaatan *image segmentation* pada *deep learning* memperoleh hasil akurasi lebih baik sebesar 98.6%.

Pemilihan algoritma yang dipakai sangat berpengaruh terhadap capaian yang diinginkan dan limitasi dari perangkat yang akan digunakan. Pengurangan penggunaan sumber daya dapat dilakukan algoritma *ResNet-50* dengan bantuan *residual network*. Dan untuk meningkatkan akurasi ditambahkan algoritma *Inception* untuk melakukan proses ekstraksi *features* dalam berbagai skala [10], [11]. Hal tersebut dikuatkan pada penelitian oleh Mustofa dan Ali [12] menjelaskan bahwa kombinasi kedua arsitektur dapat meningkatkan performa model dengan pemanfaatan pembuatan modifikasi model [13].

Sehingga dalam penelitian ini, akan digunakan kombinasi antara *Residual Network* dan *Inception* untuk membuat sebuah model identifikasi penyakit pada tanaman kakao yang efektif untuk membantu petani dan pengusaha dalam pencegahan penyebaran dan deteksi sejak dini penyakit tanaman kakao agar meminimalisir terjadinya kerugian yang timbul akibat munculnya penyakit terhadap ekonomi [14][15].

2.1. METODE PENELITIAN

Proses penelitian digambarkan dalam sebubah diagram dengan pelaksanaannya dilakukan secara berurutan dan bertahap dalam setiap langkahnya. Alur proses penelitian digambarkan pada gambar 1.



Gambar 1. Metodologi penelitian

2.2. Identifikasi *features*

Pada metode *Support Vector Machine* (SVM) diperlukan bantuan dalam proses ekstraksi *feature* sebuah model seperti *Histograms of Oriented Gradient* (HoG) atau *Local Binary Pattern* dan biasanya digunakan untuk ukuran *datasets* kecil. Arsitektur lainnya seperti *VGG19* memiliki penggunaan sumber daya tinggi saat pelatihan, sedangkan *MobileNet V2* kurang sesuai untuk performa klasifikasi tinggi dan cenderung berfokus implementasi model tertanam pada *mobile*[16]. Disisi lain arsitektur *Inception* dan *ResNet* sering digunakan memiliki kinerja yang bagus dan hasil akurasi baik. Kemampuan ekstraksi fitur dan penggunaan sumber daya lebih baik daripada beberapa arsitektur diatas seperti VGG19[17]. Arsitektur *Inception* memiliki beberapa jenis yaitu *Inception* v1, v2, v3 dan v4. Secara umum *Inception* terdiri dari kluster *layer* yaitu *factorized convolution layers, convolution layer,* dan *pooling layers* [18]. Dalam iterasi pertama arsitektur *Inception* memperkenalkan konsep *Inception modules* yang menggunakan *multiple convolution* (1x1, 3x3, 5x5) secara paralel untuk menangkap *features* dari sebuah

dataset. Salah satu komponen utama dalam versi ini adalah reduksi dimensi. Hal itu dibuktikan dengan adanya 1x1 *convolution*, dan juga menggunakan *GlobalAveragePooling* pada *layer* terakhir di *inception module*. Berikut penjelasan dari masing-masing *layer* yang ada pada *Inception* atau GoogleNet yang diperkenalkan pertama kali dijelaskan pada tabel 1 [19].

Tabel 1. Keterangan *layers* pada arsitektur *Inception*

type	*patch size / stride*	*Output size*	*depth*	*# 1x1*	*# 3x3 reduce*	*# 3x3*	*# 5x5 reduce*	*# 5x5*	*pool proj*	*params*	*ops*
convolution	7x7/2	112x112 x 64	1							2.7K	34M
max pool	3x3/2	56x56x 64	0								
convolution	3x3/1	56x56x192	2		64	192				112K	360M
max pool	3x3/2	28x28x192	0								
inception(3a)		28x28x256	2	64	96	128	16	32	32	159K	128M
inception(3b)		28x28x480	2	128	128	192	32	96	64	380K	304M
max pool	3x3/2	14x14x480	0								
inception (4a)		14x14x512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14x14x512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14x14x512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14x14x528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14x14x832	2	256	160	320	32	128	128	840K	170M
max pool	3x3/2	7x7x832	0								
inception (5a)		7x7x832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7x7x1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7x7/1	1x1x1024	0								
dropout (40%)		1x1x1024	0							1000K	1M
linear		1x1x1000	1								
softmax		1x1x1000	0								

Residual Network (ResNet) adalah jenis arsitektur *deep neural network* yang diperkenalkan oleh Kaiming He dkk yang dirancang untuk mengatasi masalah *gradient vanishing* di *deep leraning*. *Residual Network* (ResNet) mengurangi masalah ini dengan memperkenalkan *skip connections* atau *shortcut* yang memungkinkan jaringan mempelajari pemetaan *residual*. Setiap *layer* pada jaringan menerima masukan tidak hanya dari *layer* sebelumnya, tetapi juga dari *layer* sebelum-sebelumnya. Hal ini dicapai dengan menambahkan *input* suatu *layer* ke *output layer* tersebut melewati satu atau lebih *layer*. Operasi penambahan dilakukan berdasarkan elemen dapat berupa *identity mapping* yaitu meneruskan *input x* langsung ke *output.* Dan dapat berupa *residual mapping* yaitu memperlajari *features* lalu dikeluarkan pada *output*. Keluaran dari model *residual network* dapat dihitung dengan persamaan (1) [20].

$$h(x) = f(x) + x \quad (1)$$

Dari perhitungan di atas akan diperoleh nilai keluaran dari *residual network* h(x) dengan proses perhitungan *convolutional layers* f(x) di mana nilai x merupakan *input* dari *layer* x. Beberapa arsitektur *ResNet*, masing-masing dengan jumlah lapisan yang berbeda: *ResNet-18, ResNet-34, ResNet-50, ResNet-101,* dan *ResNet-152*. Ukuran *input* yang paling umum digunakan untuk arsitektur ini adalah 224x224 *pixel*, tetapi dapat bervariasi tergantung pada tugas dan *dataset* tertentu. Ukuran tersebut juga dapat disesuaikan menggunakan manipulasi citra dengan ukuran seperti 256x256 atau 299x299. Meskipun banyak versi dari arsitektur *ResNet*, secara garis besar komponen yang digunakan sama hanya berbeda pada jumlah penerapan lapisan yang digunakan dalam pembuatannya[21]. Berikut adalah lapisan yang terdapat pada arsitektur *ResNet* dijelaskan pada tabel 2.

Tabel 2. Keterangan *layers* pada arsitektur *ResNet*

layer name	*output size*	*18-layer*	*34-layer*	*50-layer*	*101-layer*
conv1	$112x112$	$7x7\ 64, stride\ 2$			
conv2_x	$56x56$	$3x3\ max\ pool, stride$			
		$\begin{bmatrix} 3x3, 64 \\ 3x3, 64 \end{bmatrix} x\ 2$	$\begin{bmatrix} 3x3, 64 \\ 3x3, 64 \end{bmatrix} x\ 3$	$\begin{bmatrix} 1x1, 64 \\ 3x3, 64 \\ 1x1, 256 \end{bmatrix} x\ 3$	$\begin{bmatrix} 1x1, 64 \\ 3x3, 64 \\ 1x1, 256 \end{bmatrix} x\ 3$
conv3_x	$28x28$	$\begin{bmatrix} 3x3, 128 \\ 3x3, 128 \end{bmatrix} x\ 2$	$\begin{bmatrix} 3x3, 128 \\ 3x3, 128 \end{bmatrix} x\ 4$	$\begin{bmatrix} 1x1, 128 \\ 3x3, 128 \\ 1x1, 512 \end{bmatrix} x\ 4$	$\begin{bmatrix} 1x1, 128 \\ 3x3, 128 \\ 1x1, 512 \end{bmatrix} x\ 4$
conv4_x	$14x14$	$\begin{bmatrix} 3x3, 256 \\ 3x3, 256 \end{bmatrix} x\ 2$	$\begin{bmatrix} 3x3, 256 \\ 3x3, 256 \end{bmatrix} x\ 6$	$\begin{bmatrix} 1x1, 256 \\ 3x3, 256 \\ 1x1, 1024 \end{bmatrix} x\ 6$	$\begin{bmatrix} 1x1, 256 \\ 3x3, 256 \\ 1x1, 1024 \end{bmatrix} x\ 23$
conv5_x	$7x7$	$\begin{bmatrix} 3x3, 512 \\ 3x3, 512 \end{bmatrix} x\ 2$	$\begin{bmatrix} 3x3, 512 \\ 3x3, 512 \end{bmatrix} x\ 3$	$\begin{bmatrix} 1x1, 512 \\ 3x3, 512 \\ 1x1, 2048 \end{bmatrix} x\ 3$	$\begin{bmatrix} 1x1, 512 \\ 3x3, 512 \\ 1x1, 2048 \end{bmatrix} x\ 3$

	$1x1$		$Average\ pool, 1000 - d\ fc, softmax$		
FLOPs	$1.8\ x\ 10^9$	$3.6\ x\ 10^9$	$3.8\ x\ 10^9$	$7.6\ x\ 10^9$	

2.3. Analisis komponen model

Merupakan proses untuk mengetahui komponen utama yang membangun sebuah arsitektur model. Masing-masing komponen sebuah arsitektur memiliki kegunaan dan keunggulan masing-masing. Dengan melakukan analisis komponen yang ada pada arsitektur *Inception* dan *ResNet*, maka pada pembangunan arsitektur dalam penelitian ini dapat menjadi optimal dan efektif. Penerapannya dapat menggunakan keseluruhan komponen yang ada pada masing-masing arsitektur atau hanya mengambil komponen inti yang terdapat pada arsitektur.

2.4. Pre-processing data

Sebelum *datasets* digunakan dalam penelitian, dilakukan pengolahan data agar data dapat digunakan pada proses pelatihan dan pengujian pembuatan arsitektur. Hal tersebut karena setiap arsitektur jaringan syaraf tiruan memiliki kriteria yang harus dipenuhi. Pemrosesan ini dilakukan dengan implementasi algoritma komputer. Teknik seperti *image enhancement, restoration, normalization, segmentation*, dan *compression* digunakan untuk meningkatkan gambar yang diperoleh dari berbagai sumber seperti kamera, satelit, dan perangkat lainnya.

2.4.1. Geometric Transformation

Untuk melakukan *image enhancement* dapat digunakan transformasi geometri, ini merujuk pada proses pengubahan *spatial pixel* untuk memperoleh hasil *translation, scaling, rotation*, dan *warping*. Proses ini dilakukan dengan membentuk ulang gambar. *Affine transformation* adalah teknik penggabungan teknik diatas dengan mempertahankan tingkat liniearitas dan dapat dilakukan pembentukan ulang pada citra. Disisi lain, *perspective transformation* melakukan simulai tiga dimensi dengan mengubah proyeksi gambar. Salah satu teknik dalam transformasi geometri untuk mempertahankan kualitas gambar adalah teknik *interpolation*. Ketika citra hendak diperbesar, maka prosesnya disebut *up-sampling*. Proses ini menambahkan *pixel* baru dengan memperkirakan nilainya berdasarkan nilai *pixel* di sekitarnya. Teknik *interpolation* yang umum digunakan seperti interpolasi *bilinear* atau *bicubic* yang memastikan transisi antar *pixel* dengan baik. Sebaliknya, *down-sampling* atau pengurangan ukuran gambar, melibatkan penghapusan *pixel* dengan menggunakan metode seperti *nearest-neighbor* [22].

2.4.2. Min-max scalling

Min-max scaling atau biasa disebut dengan *min-max normalization*, adalah sebuah teknik yang digunakan pada berbagai bidang, termasuk pemrosesan gambar melakukan memetakan dan menormalkan nilai kumpulan data. Saat diterapkan pada data gambar, *min-max scalling* mengubah nilai *pixel* di setiap *channel* gambar ke rentang tertentu. Dampak dari melakukan normalisasi pada *machine learning* adalah meningkatkan kecepatan *learning phase* dengan menormalkan nilai *input* untuk setiap atribut yang diukur pada tahap pelatihan.

Normalisasi menggunakan *min-max* adalah untuk mentransformasikan data asli secara linier. Asumsikan $\min_A$ dan $\max_A$ sebagai nilai minimum dan maksimum dari atribut A. Normalisasi *min-max* memetakan nilai v_i dari A ke V_i dalam rentang [$\min_A$ baru, $\max_A$ baru]. Operasi *min-max* dapat dilakukan dengan persamaan (2) [23].

$$V_i = \left(\frac{v_i - min_A}{max_A - min_A}\right) * (new_{maxA} - new_{minA}) + new_{minA} \quad (2)$$

Dengan perhitungan pada persamaan (2) akan diperoleh nilai hasil pemetaan (V_i) dengan menggunakan variabel nilai data sebelum diolah (v_i), nilai minimum data A (minA) dan nilai maksimul data A (maxA).

2.4.3. Background removal

Penghapusan latar belakang (background removal) merupakan salah satu cara yang digunakan untuk meningkatkan hasil dari pembentukan model arsitektur. Diperoleh peningkatan hasil akurasi model sebesar 5% dari perbandingan tersebut. Hasil tersebut dapat bervariasi dengan faktor arsitektur, jumlah klasifikasi kelas, hingga jumlah data yang digunakan [24]. Untuk melakukan penghapusan latar belakang sebuah citra ada banyak teknik yang dapat digunakan kedua diantaranya adalah *grabcut*, *alpha matting*. Cara kerja dari teknik *grabcut* adalah dengan membuat sebuah area pada citra yang dikehendaki secara kasar. Kemudian dari area tersebut akan dibuat *node* dari *pixel* yang ada pada area tersebut untuk dideteksi area objek dan area latar belakang.

Selain itu, teknik lain yang digunakan adalah *alpha matting*. Merupakan teknik dasar dalam pemrosesan gambar yang memungkinkan akurasi kontrol aspek transparansi dalam suatu gambar. Dengan memberikan nilai antara 0 dan 1 untuk setiap piksel, *alpha matte* menentukan seberapa banyak gambar berada pada latar belakang yang terlihat dan seberapa banyak yang transparan seperti yang ditampilkan pada gambar 2.

Gambar 2. Contoh implementasi *alpha matte*

Proses penghapusan latar belakang diatas memanfaatkan *image segmentation* untuk

memisahkan objek utama pada gambar dengan latar belakang.

2.5. Pemilihan model *approach*

Pembuatan sebuah model dapat dilakukan dengan penggunaan arsitektur secara utuh, atau pembuatan arsitektur dengan implementasi modifikasi. Modifikasi dapat berupa proses ekstraksi *features* atau dengan mengintegrasikan beberapa arsitektur lain untuk mengambil keunggulan dari sebuah arsitektur. Metode integrasi yang dapat dilakukan dapat berupa *fusion approach* dan *ensemble approach*.

2.5.1. Fusion approach

Fusion approach mengacu pada proses integrasi model, sumber data, atau bermacam teknik pembelajaran suatu model menghasilkan hasil yang lebih tepat dan akurat. Dengan melakukan penggabungan informasi dari berbagai sumber, *fusion approach* dapat menghasilkan model dengan menggunakan keunggulan dari model lain dalam mempelajari pola, *features*, dan gambaran pada suatu *datasets*. Untuk melakukan pembuatan sebuah model dengan *fusion approach*, dapat digunakan strategi *early fusion*, *late fusion,* dan *hybrid fusion* [25]. *Early fusion* (feature level fusion) adalah sebuah strategi yang digunakan untuk melakukan kombinasi data mentah atau *features* sebelum dilakukan proses pelatihan model.

Disisi lain, strategi *late fusion* (decision level fusion) digunakan ketika ingin mengintegrasikan jenis modalitas yang berbeda, tetapi dilakukan pemrosesan secara independen. Seperti yang ada pada penelitian yang dilakukan oleh [26] dijelaskan bahwa strategi *late fusion* sangat efektif digunakan ketika memiliki data yang dihasilkan oleh beberapa sensor dengan kondisi dan ukuran data tertentu. Sehingga, keunikan dari masing-masing sumber data akan berkontribusi membantu meningkatkan kinerja model secara keseluruhan [27].

Strategi terakhir yang dapat digunakan menggunakan *fusion approach* adalah dengan *hybrid fusion*. Merupakan strategi yang menggabungkan strategi *early fusion* dan *late fusion* secara bersamaan. Proses ini dilakukan untuk menciptakan sebuah solusi terpadu dalam penanganan kasus klasifikasi dengan ukuran data besar dan tingkat kerumitan data tinggi [28].

2.5.2. Ensemble approach

Merupakan pendekatan dengan integrasi beberapa model. Ada beberapa teknik teknik yang digunakan seperti *bagging*, *boosting*, dan *stacking* menjadi landasan bagi banyak strategi *ensemble* yang ada saat ini.*Bagging* (Bootstrap Aggregating) digunakan untuk mengurangi *varians* model dengan melatih beberapa versi model pada subset data yang berbeda. Beberapa model yang terbentuk, semua digunakan untuk melakukan prediksi dengan teknik pembentukan prediksi akhir berdasarkan suara mayoritas terbanyak atau rata-rata dari beberapa model [29].

Teknik lain yang digunakan dalam *ensemble approach* adalah teknik *boosting*. *Boosting* berfokus pada peningkatan akurasi sebuah model dengan melatih model secara berurutan pada pelatihan model berikutnya. Setiap iterasi pelatihan menggunakan *datasets* yang sama pada *boosting*. Teknik selanjutnya adalah *stacking* yaitu dengan membagi sebuah arsitektur menjadi dua bagian yaitu *base model* dan *meta model*. *Base model* adalah sebuah sebuah lapisan terdiri atas beberapa model yang dilatih menggunakan *datasets training*.

Lapisan *meta-model*, hasil yang diperoleh dari *base model* dijadikan sebuah sebagai *features* untuk pelatihan *meta-model*. Lapisan ini nantinya akan mengenali dan mempelajari cara penggabungan prediksi dari *base model* [30]. Teknik terakhir adalah *voting*, yaitu pendekatan dengan melakukan pemilihan hasil prediksi dari beberapa sumber model untuk meningkatkan akurasi dan mengurangi *overfitting*.

2.6. Implementasi

Merupakan proses pembuatan arsitektur modifikasi dengan menggunakan jenis arsitektur asal untuk dilakukan peningkatan kinerja dan metode pendekatan yang dipilih. Proses implementasi ini dilakukan untuk melihat hasil yang diperoleh dari penelitian yang dilakukan berdasarkan *datasets* yang digunakan dalam studi kasus. Hasil berupa bentukan arsitektur yang dapat digunakan untuk membuat model untuk dilakukan analisis dan evaluasi bentukannya.

2.7. Evaluasi dan analisis

Model yang terbentuk akan dilakukan evaluasi dan analisis terkait kinerja yang dihasilkan. Untuk mengetahui hasil yang diberikan dari sebuah bentukan model digunakan metode *confussion matrix*. Metode ini dilakukan dengan visualisasi terhadap tingkatan algoritma pada kelas yang berbeda dan tidak bergantung pada algoritma. *Confusion matrix* terdiri atas 2x2 matrix. Dengan besaran tersebut dapat diperoleh empat kemungkinan keluaran yang didefinisikan pada tabel 3.

Tabel 3. *Confussion matrix*

		Predicted		**Total**
		Good	**Bad**	
Reality	Good	True Positive	False Negative	n_g
	Bad	False Positive	True Negative	n_b

Berdasarkan tabel 3 maka dapat dilakukan formulasi untuk melakukan perhitungan terhadap nilai *accuracy, error rate, precision, recall,* dan *specificity.* Nilai akurasi adalah rasio *True Positive* dan *True Negative* terhadap jumlah total yang dapat dihitung dengan menggunakan persamaan (3). Nilai *error rate (ERR)* adalah rasio *False Positive* dan *False Negative* terhadap jumlah total yang dapat dihitung dengan persamaan (4). Nilai *precision*

adalah akurasi positif atau tingkat positif sebenarnya yang ditunjukkan pada persamaan (5). Nilai *recall* adalah nilai akurasi model dalam identifikasi semua data *True Positives* (TP) yang ditunjukan pada persamaan (6). Nilai *specificity* adalah keakuratan negatif atau tingkat negatif sebenarnya ditunjukkan pada persamaan (7). Dan untuk perhitungan *F1-score* adalah dengan menghitung rata-rata harmonik antara *precision* dengan *recall* sesuai pada persamaan (8) [31].

$$accuracy = \frac{TP + TN}{n_b + n_g} \quad (3)$$

$$ERR = \frac{FN + FP}{n_b + n_g} \quad (4)$$

$$precision = \frac{TP}{(True\ Positives\ +\ False\ Positives)} \quad (5)$$

$$recall = \frac{TP}{(True\ Positives\ +\ False\ Negatives)} \quad (6)$$

$$specificity = \frac{TN}{True\ Negatives\ +\ False\ Positives} \quad (7)$$

$$F1 = 2\ \frac{Precision * Recall}{Precision + Recall} \quad (8)$$

2. HASIL DAN PEMBAHASAN

2.1 Identifikasi *features*

Dalam beberapa studi yang peneliti temukan, kedua arsitektur memiliki ukuran masukan yang berbeda yaitu pada [32] dan [33]. Pemilihan jenis arsitektur pada penelitian ini sesuai dengan tujuan penelitian untuk memperoleh efisiensi proses pelatihan dengan tetap memperoleh hasil yang baik. Berikut adalah perbandingan dari masing-masing arsitektur berdasarkan iterasi versi yang ada disajikan pada tabel 4.

Tabel 4. Perbandingan *image size* arsitektur

No	Arsitektur	*Input image*
1.	Inception	224x224x3
2.	InceptionV2	299x299x3
3.	InceptionV3	299x299x3
4.	InceptionV4	299x299x3
5.	ResNet-18	224x224x3
6.	ResNet-34	224x224x3
7.	ResNet-50	224x224x3
8.	ResNet-101	224x224x3

Sehingga, ditentukan arsitektur yang akan digunakan adalah *InceptionV3* dengan *ResNet-50*. Untuk mengakomodasi kedua masukan yang berbeda, dilakukan *resizing* citra pada masukan *ResNet-50* yaitu 224x224x3, karena akan menimbulkan degradasi performa yang tidak terlalu besar. Sehingga dalam implementasi penggunaan arsitektur akan menggunakan tinggi 299, lebar 299 dengan *colormode* 3 *channel* atau RGB.

2.2 Analisis komponen model

Sebelum proses pembuatan model, dilakukan analisa terhadap komponen-komponen model, kegunaan komponen model, hingga keunggulan dari penggunaan komponen tersebut. Proses ini bertujuan agar dapat mengetahui pendekatan model yang cocok untuk arsitektur *InceptionV3* dan *ResNet-50*. Sehingga akan digunakan beberapa komponen dari masing-masing arsitektur untuk membentuk sebuah model yaitu *inception module, factorization, auxiliary classifier, activation function,* dan *dropout* dari arsitektur *InceptionV3*. Sedangkan untuk komponen dari *ResNet-50* yaitu *activation function, batch normalization*, dan *shortcut*. Untuk penjelasan dari masing-masing komponen yang akan digunakan dapat dilihat pada tabel 5.

Tabel 5. Komponen yang digunakan pada model

No	Komponen	Keterangan
1.	*Inception Module*	Modul yang memiliki ukuran filter berbeda (1x1, 3x3, dan 5x5) dengan lapisan *pooling* untuk ekstraksi *features*.
2.	*Factorization*	Modul untuk mengurangi jumlah parameter dalam jaringan. Proses ini dilakukan dengan menguraikan konvolusi besar menjadi konvolusi yang lebih kecil.
3.	*Auxiliary Classifiers*	Modul yang ditambahkan pada ditengah arsitektur untuk membantu mengenali lebih banyak pola pada *dataset*.
4.	*Activation function*	Modul untuk mengubah keluaran linier jaringan menjadi keluaran non-linier. Agar dapat lebih mudah menangani pengenalan data pada model kompleks.
5.	*Dropout*	Modul untuk mengatur masukan dari unit masukan secara acak dengan nilai nol pada saat pelatihan model.
6.	*Batch normalization*	Modul untuk melakukan penyeimbangan pelatihan dengan menormalisasikan fungsi aktivasi.
7.	*Shortcut*	Modul untuk menghubungkan modul masukan langsung ke modul keluaran dengan cara i*dentity mapping* atau *projection mapping*.

2.3 *Pre-processing data*

Data yang digunakan dalam penelitian ini diambil dari sumber terbuka yang dibagikan oleh Zaldy Jr. Pagaduan berupa *file* citra dengan tiga kategori kondisi buah kakao. *Dataset* diambil pada tahun 2020 dilakukan di Negara Filipina. Penggunaan data ini dilakukan untuk mencerminkan hasil yang relevan terkait situasi dan kondisi yang ada di Indonesia. Sebelum *dataset* digunakan dalam penelitian, dilakukan penamaan label pada masing-masing kategori. Label kategori tersebut terdapat pada tabel 6.

Tabel 6. Label kelas pada *dataset* citra

No	Kelas	Label
1.	Sehat	0
2.	*Black Pod Rot*	1
3.	*Pod borer*	2

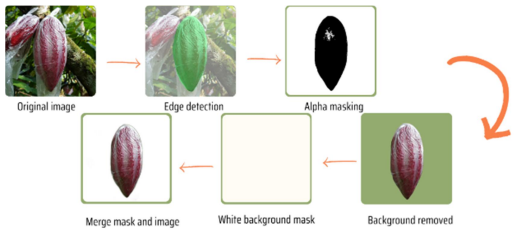
Kemudian, citra yang telah dikelompokan dilakukan seleksi secara manual untuk menguji kelayakan data. Data citra yang layak memiliki beberapa kriteria seperti buah yang terlihat dengan jelas, buah yang tidak tertutupi objek lain, buah

kakao yang tidak bersinggungan dengan buah lain, dan fokus objek pada citra tidak pada buah kakao. Citra yang lolos uji kelayakan pada proses pemilihan manual, dilakukan pemerataan ke seluruh kelas yang ada. Proses tersebut bertujuan untuk menghindari beberapa masalah seperti hasil bias terhadap sebuah kelas, performa yang lemah, akurasi lemah pada kelas minoritas, ketidakstabilan proses pelatihan. Setelah data citra dilakukan proses seleksi manual, pemerataan data diperoleh jumlah total citra yang akan digunakan berjumlah 481 citra dengan detail pembagian ditampilkan dalam tabel 7.

Tabel 7. Label kelas pada *dataset* citra

No	Kelas	Label	Jumlah data
1.	Sehat	0	200
2.	*Black Pod Rot*	1	195
3.	*Pod borer*	2	86
		Jumlah	481

Untuk menghasilkan model yang baik, dilakukan juga proses penghilangan latar belakang pada citra di semua kelas. Langkah tersebut dilakukan karena *dataset* memiliki objek lain pada citra yang dapat mengganggu proses pelatihan dan pengujian model [34]. Proses dilakukan dengan melakukan penghilangan latar belakang pada folder masing-masing kelas, kemudian ditambahkan latar belakang berwarna netral yaitu warna putih pada citra. Hasil pengolahan citra menjadikan model dapat fokus terhadap *features* utama dan tidak terganggu dengan latar belakang, membuat sampel *dataset* yang konsisten, dan ada separasi antar kelas. Alur dari penghapusan latar belakang ditunjukkan melalui gambar 3.



Gambar 3. Alur penghapusan latar belakang

Terlihat pada gambar 3, citra dimulai dengan data awal yang dimasukan kemudian dipisahkan antara latar belakang dengan objek menggunakan *edge detection*. Lalu, objek diberikan *alpha masking* untuk memudahkan proses penghilangan latar belakang. Dan objek buah yang sudah dipisahkan ditambahkan latar belakang warna putih untuk memudahkan proses pelatihan dengan melakukan penggabungan antar kedua *layers*.

Dari data tersebut dibagi menjadi data pelatihan, data validasi dan data testing dengan rasio *default* 80:10:10 dari total 481 citra. Rasio tersebut dipilih untuk memaksimalkan *datasets* sebagai data latih. Proses pembagian data dilakukan dengan menyimpan data asli dan melakukan penyalinan data dengan tujuan proses pelatihan dapat lebih dinamis ketika ingin dilakukan perubahan *dataset*, ukuran rasio, dan parameter augmentasi pada *dataset* sesuai. *Dataset* yang telah dibagi kemudian akan dilakukan transformasi geometri untuk menambah variasi pada data. Proses transformasi menggunakan bantuan modul *ImageDataGenerator* yang dapat diatur parameter augmentasinya. Berikut adalah parameter augmentasi yang digunakan dijelaskan pada tabel 8.

Tabel 8. Parameter augmentasi data citra

No	Jenis data	Parameter
1.	Latih	rescale=1/255.0, shear_range=0.2, zoom_range=0.2, rotation_range=7, width_shift_range=0.2, height_shift_range=0.2
2.	Validasi	rescale=1/255.0
3.	Uji	rescale=1/255.0

Dari tabel 8 terlihat bahwa data latih mengalami banyak augmentasi seperti normalisasi data dengan *min-max scalling*, rotasi, hingga perpindahan citra. Parameter yang digunakan diatas menyesuaikan dengan studi kasus penelitian bahwa citra dapat memiliki tingkat perbesaran yang berbeda, sudut pengambilan yang berbeda, dan rotasi yang berbeda. Nilai-nilai diatas berdasarkan hasil percobaan mandiri yang menghasilkan kesesuaian citra tanpa menimbulkan efek berlebih atau menghasilkan citra yang terlalu esktrim. Hal ini juga bertujuan melatih model agar mengenali data-data baru yang bisa terbentuk dari *datasets* yang dimiliki.

Namun untuk data validasi dan uji tidak diterapkan augmentasi kecuali normalisasi data. Alasannya adalah untuk menguji dan memvalidasi model dengan membandingkan data yang telah diaugmentasi agar model dapat melakukan generalisasi. Setelah dilakukan augmentasi, proses selanjutnya adalah menyesuaikan ukuran citra dengan ukuran *input layer* arsitektur yaitu 299x299 dengan mengelompokan data menjadi *batch* berukuran 16 dan 32. Hasil pengolahan tersebut berupa subset yang terbentuk adalah *train_generator, validation_generator*, dan *test_generator* yang siap digunakan untuk membuat model klasifikasi.

2.4 Model *approach*

Dalam melakukan pembentukan arsitektur untuk membuat model klasifikasi, dilakukan percobaan dengan menggunakan beberapa metode yaitu dengan pendekatan *fusion*, *ensemble*, dan *hybrid*. Prosesnya diawali dengan melihat keunggulan yang dapat digunakan pada suatu arsitektur, jenis-jenis *layer* yang ada pada arsitektur, hingga ukuran dari masing-masing *layer* dan filter. Setelah itu dari kedua jenis arsitektur akan dilakukan pengujian terhadap *dataset* yang dimiliki untuk

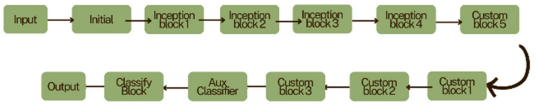
melihat kecocokan arsitektur dan hasil pembentukan model yang dihasilkan.

Penelitian dilakukan dengan melakukan *fusion approach* untuk menggabungkan kekuatan yang dimiliki oleh arsitektur *InceptionV3* dalam melakukan proses penangkapan *features* dalam *datasets*. Hal ini juga dipengaruhi oleh jumlah *datasets* yang tidak seimbang dapat memicu bias terhadap hasil kelas mayoritas.

3. Implementasi

3.1 Arsitektur

Implementasi arsitektur model menggunakan *library* tensorflow 2.18.0 yang dikembangkan oleh Google. Proses implementasi dilakukan pada media utama *google collab* untuk membantu proses pelatihan model. Implementasi menggunakan *backbone architecture InceptionV3* dengan *fusion architecure ResNet*. Berikut adalah gambaran arsitektur yang dihasilkan ditampilkan pada gambar 4.

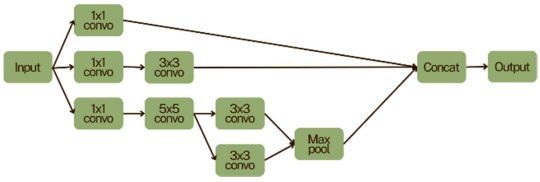


Gambar 4. *Overview* arsitektur yang dikembangkan

Dari gambar 4 memperlihatkan gambaran besar arsitektur yang digunakan, terdapat beberapa bagian *block* yang ada pada arsitektur. Pembentukan arsitektur diatas menerapkan teori penggunaan *fusion approach* dengan metode *early fussion*. Metode ini akan melakukan ekstraksi *features* menggunakan *Inception* sebagai *backbone architecture* dengan penggabungan *ResNet* di *custom block* setelah *Inception block*.

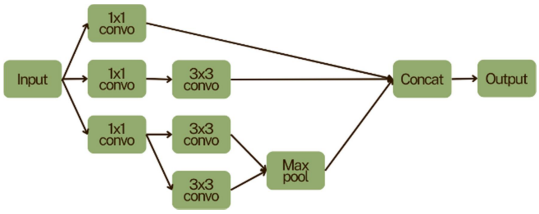
Gambar 5. *Overview initial block* arsitektur

Lapisan pertama yang akan digunakan dalam arsitektur ini adalah *initial block*. Pada lapisan ini citra yang masuk akan dilakukan ekstrasi *features* awal. Proses ekstraksi ini dilakukan untuk mempersiapkan data sebelum dilakukan pada lapisan selanjutnya. Lapisan ini digambarkan pada gambar 5 dengan melakukan ekstraksi menggunakan ukuran *kernel* 3x3 sebagai proses ekstraksi awal citra.



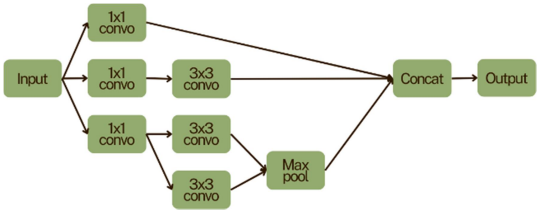
Gambar 6 *Overview Inception block* 1 arsitektur

Setelah proses ekstraksi awal *features* dilakukan, selanjutnya citra akan masuk ke dalam *Inception block* 1. Seperti gambar 6, citra akan dilakukan pemrosesan menggunakan kombinasi ukuran *kernel* yaitu 3x3 dan 5x5. Hal tersebut untuk memperoleh hasil ekstraksi *features* lebih bervariasi. Sehingga model dapat memahami konteks secara keseluruhan menggunakan ukuran *kernel* 5x5 dan *local features* menggunakan ukuran *kernel* 3x3.



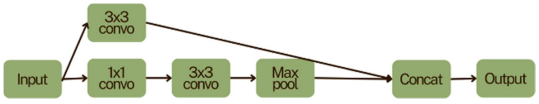
Gambar 7. *Overview Inception block* 2 arsitektur

Citra kemudian akan masuk ke dalam *Inception block* 2 untuk dilakukan proses ektraksi lebih lanjut. Pada gambar 7 ditunjukan bahwa citra akan melewati ukuran *kernel* 3x3 pada beberapa cabang. Salah satu cabang memiliki 2 ukuran *kernel* 3x3 dengan maksud menggantikan ukuran *kernel* yang ada pada *Inception block* sebelumnya agar lebih efisien dalam proses pelatihan.



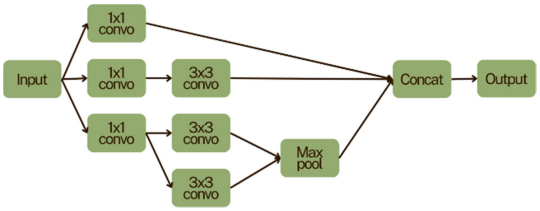
Gambar 8. *Overview Inception block* 3 arsitektur

Pada *Inception block* 3 citra akan dilakukan ekstraksi *features* lebih lanjut untuk mendapatkan informasi lebih banyak pada citra. Lapisan ini digambarkan pada gambar 8 yang sama dengan lapisan *Inception block* 2



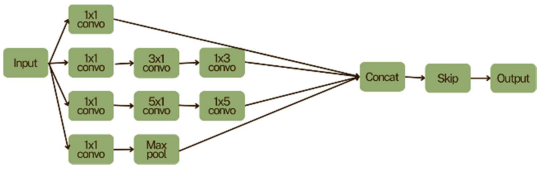
Gambar 9. *Overview Inception block* 4 arsitektur

Pada gambar 9, arsitektur akan melakukan reduksi spasial pada *features map* yang telah dihasilkan pada proses sebelumnya. Hal ini bertujuan untuk meningkatkan arsitektur dalam mempelajari hal abstrak yang ada pada citra dan lebih membantu arsitektur dalam jaringan yang lebih dalam untuk lebih mendapatkan informasi *features* citra. Meskipun dilakukan reduksi spasial dengan menggunakan *downsampling*, proses ini akan mempertahankan informasi yang telah didapatkan pada proses sebelumnya dengan menggunakan *stride* 2.



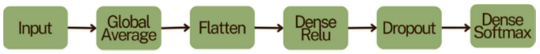
Gambar 10. *Overview Inception block* 5 arsitektur

Setelah itu, citra akan masuk ke dalam lapisan *Inception block* 5 untuk dilakukan ekstraksi *features* lebih lanjut. Seperti pada gambar 10, ditunjukan bahwa proses ini serupa dengan beberapa *Inception block* sebelumnya. Penggunaan ukuran *kernel* 3x3 dengan melakukan faktorisasi ukuran *kernel* 5x5 dipilih untuk mengurangi sumber daya yang digunakan ketika proses pelatihan atau pembuatan model.



Gambar 11. *Overview custom block* arsitektur

Setelah proses ekstraksi *features* dilalui pada lapisan *Inception*, citra kemudian akan dimasukan ke dalam beberapa lapisan *custom block.* Lapisan yang digambarkan pada gambar 11 dibuat menggunakan kombinasi antara *Inception* dan *ResNet*, yaitu penggunaan proses *Inception* sebagai panduan utama dengan melakukan penambahan *shortcut* atau *skip connection*. Hal ini bertujuan agar data yang dilalui pada lapisan *shortcut* dapat diolah lebih lanjut untuk mendapatkan informasi mendalam tanpa dilakukan pemrosesan terlebih dahulu menggunakan proses *projection*.



Gambar 12. *Overview classify block* arsitektur

Setelah semua proses dilakukan pada pemrosesan sebelumnya, kemudian citra akan dilakukan pembuatan klasifikasi berdasarkan jumlah kelas yang ditentukan. Pada penelitian ini jumlah kelas yang terdapat pada *dataset* yang digunakan berjumlah 3. Gambar 12 menunjukkan bahwa sebelum dilakukan klasifikasi akhir, dilakukan pengurangan dimensi spasial menggunakan *Global Average Pooling*. Setelah itu dilakukan konversi *features map* menjadi vektor 1 dimensi menggunakan *block Flatten*. Lalu dilakuan pembuatan klasifikasi menggunakan *fully connected layer* dan pembuangan neuron secara acak menggunakan fungsi *Dropout*.

3.2 Model

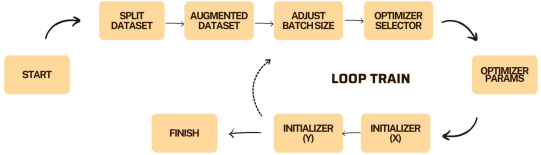
Dari arsitektur yang telah dikembangkan, kemudian akan dibuat beberapa model dengan menggunakan parameter-parameter berbeda untuk membandingkan hasil dari masing-masing bentukan model. Parameter yang digunakan dalam pembuatan model adalah *learning rate, batch size, initializers*. *Learning rate* dan *batch size* merupakan parameter yang termasuk dalam kategori *hyperparameter* . Artinya adalah nilai dari parameter terserbut dapat memiliki pengaruh yang besar ketika diubah. Dalam penelitian ini parameter-parameter akan digunakan menjadi beberapa konfigurasi yang ditampilkan pada tabel 9. Parameter tersebut dikombinasikan untuk membuat model dengan contoh konfigurasi yang terlihat pada gambar 13.

Tabel 9. Konfigurasi parameter pembentukan model

No	Parameter	Konfigurasi
1.	*Learning rate*	0.01, 0.001, dan 0.0001
2.	*Batch sizes*	16 dan 32
3.	*Initializers (x)*	*He normal dan He uniform*
4.	*Initializers (y)*	*Glorot Normal dan Glorot uniform*

Gambar 13. Contoh kombinasi parameter pembentukan model

Setelah didapatkan kombinasi parameter yang digunakan, dilakukan proses pelatihan model yang akan terus berjalan hingga kombinasi yang terbentuk telah habis. Proses ini akan berjalan otomatis dalam sebuah perulangan dengan menerapkan pencatatan (logging) dan penyimpanan otomatis menggunakan *model checkpoint*. Kedua fungsi tersebut memanfaatkan *callback* pada setiap prosesnya. Model yang terbentuk akan disimpan dengan melihat nilai dari ”val_accuracy” dalam setiap *epoch* sebagai penentu proses penyimpanan atau pembaharuan model yang akan disimpan. Model akan disimpan jika terdapat peningkatan nilai ”val_accuracy” yang dihasilkan dari nilai sebelumnya. Proses yang terjadi pada pembuatan model ditampilkan pada gambar 14.



Gambar 14. Alur pembuatan model

Dalam proses perulangan perhitungan kesalahan prediksi menggunakan *loss function CategoricalFocalCrossEntropy* untuk membantu permasalahan *imbalanced class* yang berarti jumlah data antar masing-masing kelas tidak seimbang dengan memberikan nilai *alpha* dengan nilai 0.24 untuk class_0, nilai 0.12 untuk class_1, dan nilai 0.63 untuk class_2. Variabel ini sudah ditentukan berdasakan beberapa hasil pengujian menggunakan epoch kecil sebesar 10 pada beberapa konfigurasi dan fungsi optimasi yang digunakan. Sehingga untuk proses pelatihan selanjutnya, nilai tersebut yang akan menjadi parameter pembantu dalam pembuatan

model. Selain itu, digunakan juga parameter *label smoothing* yang membantu proses identifikasi agar tidak menganut ketentuan *hard labeling*. Sehingga akan memunculkan presentase kemungkinan yang paling besar (highest confidance) pada jenis kelas yang mungkin menjadi bagian dari citra yang dideteksi.

Selain itu dari gambar 14 terlihat ada dua jenis *initializers* yaitu *He initializers* dan *Glorot initializers*. Keduanya digunakan untuk membantu model mengurangi permasalahan *generalization* atau kendala model dalam mengenali data baru dari *datasets* yang dimiliki. Masalah lainnya juga dapat terjadi ketika data tidak seimbang adalah *overfitting* yaitu model sangat baik dalam mengenali data latih termasuk pada data kotor (noise) yang berpengaruh pada buruknya kinerja dalam mengenali data baru. *He initializers* memiliki dua jenis yaitu *normal* dan *uniform*. Sedangkan untuk *glorot initializers* terdapat jenis *normal* dan *uniform*. Keduanya digunakan berkesinambungan untuk menguji kecocokan kombinasi yang dapat digunakan antar keduanya dalam studi kasus penelitian ini.

4. Evaluasi dan analisis

Berdasarkan kombinasi parameter yang digunakan dalam proses pembentukan model, didapatkan total keseluruhan model yang berhasil terbentuk sebanyak 48 model. Dari jumlah tersebut terbagi menjadi 24 model pada fungsi optimasi RMSProp dan 24 model pada fungsi optimasi Adam. Keduanya juga masih dibagi menjadi 2 bagian yang masing-masing terdiri atas 12 model dengan *batch size* 16 dan 12 model dengan *batch size* 32. Pada data yang diperoleh dari keseluruhan model diperoleh bahwa dalam proses pembuatan model arsitektur *InceptionV3* dengan *ResNet-50* pemilihan fungsi optimasi sangat berpengaruh terhadap proses pembuatan. Hal tersebut dapat menyebabkan *hyperparameter* yang digunakan memiliki hasil yang berbeda meskipun sudah diatur dengan konfigurasi serupa. Oleh karena itu data yang ditampilkan oleh masing-masing fungsi optimasi, keduanya menunjukan hasil yang tidak sama.

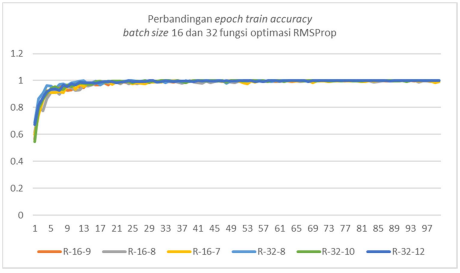
Ketika dibandingkan dengan penggunaan fungsi optimasi RMSProp, hasil yang diperoleh oleh fungsi optimasi Adam sedikit berbeda mengingat fungsi tersebut tidak memiliki hasil yang terlalu signifikan jika membandingkan kedua jenis *batch size*. Selain itu, fungsi optimasi Adam lebih sensitif terhadap terhadap ukuran *batch*. Itu dapat terjadi karena fungsi optimasi Adam bekerja dengan melakukan adaptasi *learning rate* yang mengakibatkan gradien dengan nilai kecil akan dilakukan penyesuaian. Sedangkan pada fungsi optimasi RMSProp, hal tersebut tidak dilakukan karena optimasi ini lebih berfokus pada nilai *variance* yaitu nilai kuadrat gradien rata-rata yang selalu berubah. Sehingga didapatkan hasil berbeda meskipun memiliki konfigurasi pembuatan model yang serupa dengan penggantian variabel pada jenis fungsi optimasi yang digunakan. Berikut adalah hasil perbandingan *batch size* yang diperoleh pada fungsi optimasi RMSProp ditampilkan pada tabel 10.

Tabel 10. Perbandingan hasil *batch size* pada keluaran model fungsi optimasi RMSProp

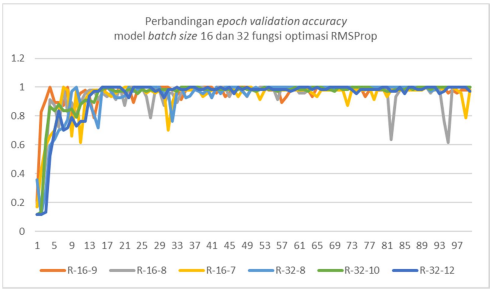
No	*Parameters*	*Mean*		*Standar deviation*		*Min*		*Max*	
		16	32	16	32	16	32	16	32
1	*Train accuracy*	0.9093	0.9488	0.1452	0.0882	0.2891	0.3548	1.0000	1.0000
2	*Validation accuracy*	0.8562	0.8761	0.2087	0.2045	0.1489	0.1194	1.0000	1.0000
3	*Train loss*	0.0651	0.0625	0.0133	0.0102	0.0564	0.0565	0.1640	0.1817
4	*Validation loss*	0.1281	0.1703	0.3479	0.5120	0.0555	0.0557	3.6345	3.7524

Dari tabel 10 diperoleh hasil bahwa penggunaan *batch size* 32 pada fungsi optimasi RMSProp lebih dapat memberikan hasil yang baik dibandingkan dengan penggunaan *batch size* 16. Meskipun secara kasat mata *batch size* 16 akan mendapatkan hasil yang lebih baik seperti pada rata-rata *validation loss* dan tingkat persebaran data yang memiliki nilai rendah 0.3479, terdapat faktor lain berupa *learning rate* dan fungsi inisialisasi yang digunakan. Sehingga dengan penyesuaian yang tepat *batch size* yang besar dapat diberikan kompensasi oleh parameter yang digunakan. Hal tesebut terlihat pada nilai parameter uji yang dimiliki pada *batch* 32 menunjukan hasil positif dengan hasil akurasi latih dan uji lebih tinggi daripada *batch size* 16. Kedua nilai akurasi juga memiliki persebaran yang rendah, mengindikasikan bahwa nilai akurasi yang dihasilkan tidak menjauh terhadap nilai rata-rata yang dihasilkan.

Dengan pertimbangan yang telah dijelaskan diatas, maka dipilih model terbaik yang dapat digunakan pada *batch size* 16 dan *batch size* 32. Model R-16-9, R-16-8 dan R-16-7 akan digunakan sebagai model terbaik yang ada pada *batch size* 16. Sedangkan model R-32-8, R-32-10, dan R-32-12 sebagai model terbaik pada *batch size* 32. Untuk memilih model yang baik dari fungsi optimasi RMSProp, akan dibandingkan kinerja model melalui nilai akurasi dan *loss*.

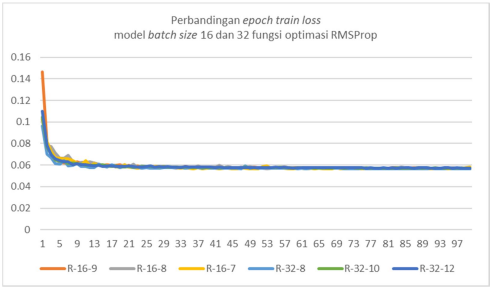


Gambar 15. Perbandingan nilai *train accuracy* fungsi optimasi RMSProp dengan *batch size* 16 dan 32 pada setiap *epoch*

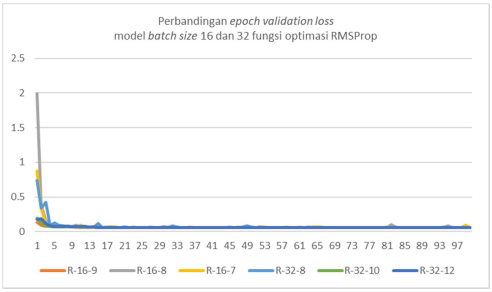


Gambar 16. Perbandingan nilai *validation accuracy* fungsi optimasi RMSProp dengan *batch size* 16 dan 32 pada setiap *epoch*

Berdasarkan gambar 15, ditunjukan bahwa keempat model dapat menghasilkan akurasi latih yang baik. Model tersebut juga dapat mencapai nilai maksimal dengan cepat karena tidak membutuhkan iterasi *epoch* lebih dari 25. Iterasi yang terjadi pada pembentukan model dengan laju dinamis menjadikan model yang dihasilkan memiliki nilai akurasi baik. Meskipun hasil akhir yang dicapai serupa atau sama, tetapi dalam proses pengenalan pola *datasets* model yang menggunakan *batch size* 32 memiliki laju yang lebih dinamis terutama pada model R-32-10 dan R-32-12. Kedua model memiliki laju yang baik dihasilkan menggunakan *learning rate* 0.0001. Hal tersebut terlihat dari gambar 16.



Gambar 17. Perbandingan nilai *train loss* fungsi optimasi RMSProp dengan *batch size* 16 dan 32 pada setiap *epoch*



Gambar 18 Perbandingan nilai *validation loss* fungsi optimasi RMSProp dengan *batch size* 16 dan 32 pada setiap *epoch*

Terlihat dari gambar 17 dan 18 perbandingan antara kedua *batch size* pada nilai *lost* yang diperoleh ketika proses latih dan validasi. Perbandingan tersebut menghasilkan bahwa keduanya dapat melakukan penyesuaian parameter dengan baik, ditunjukkan melalui grafik yang berjalan dinamis dan tidak ada anomali yang terjadi pada proses iterasi *epoch*. Setelah diperoleh hasil perbandingan antara *batch size* yang ada pada model dengan fungsi optimasi RMSProp, selanjutnya akan dilihat perbandingan *batch size* pada fungsi optimasi Adam. Data perbandingan antara kedua ukuran *batch* fungsi optimasi Adam dapat dilihat pada tabel 10.

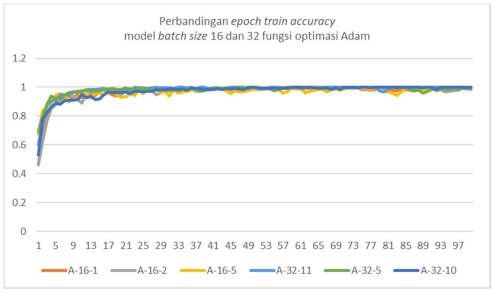
Seperti yang tampak pada tabel 10, ditunjukkan bahwa model yang menggunakan *batch size* 16 memiliki kinerja yang secara konsisten memberikan hasil kinerja yang baik pada semua parameter. Hal tersebut mendukung penjelasan sebelumnya yang memberikan titik berat fungsi optimasi Adam pada tingkat sensitivitasnya terhadap *learning rate*, *batch sizes*, dan *gradien*t. Dengan menggunakan konfigurasi yang sama pada fungsi optimasi RMSProp, diperoleh hasil bahwa kombinasi antara *learning rate* dan *batch size* 32 pada fungsi optimasi Adam kurang sesuai digunakan jika dibandingkan dengan *batch size* 16.

Tabel 11. Perbandingan hasil *batch size* pada keluaran model fungsi optimasi Adam

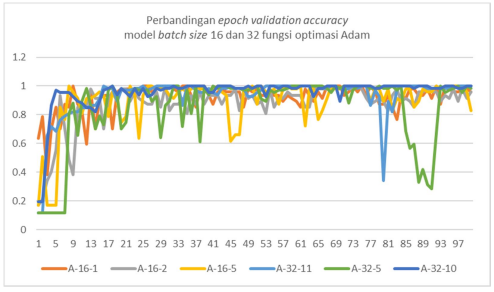
No	*Parameters*	*Mean*		*Standar deviation*		*Min*		*Max*	
		16	32	16	32	16	32	16	32
1	*Train accuracy*	0.9648	0.9488	0.0719	0.0773	0.3516	0.3180	1.0000	1.0000
2	*Validation accuracy*	0.9111	0.8177	0.1377	0.2509	0.1702	0.1194	1.0000	1.0000
3	*Train loss*	0.0610	0.0628	0.0079	0.0090	0.0566	0.0566	0.1668	0.1887
4	*Validation loss*	0.0917	0.2020	0.2108	0.5430	0.0555	0.0557	3.6345	3.7524

Selain itu, akurasi validasi data pada subset data validasi secara keseluruhan memiliki perbedaan hasil sebesar kurang lebih 10% lebih baik dibandingkan dengan penggunaan *batch size* 32. Untuk melihat perbandingannya pada model terbaik yang dipilih, berikut data grafik pada parameter *train accuracy, validation accuracy, train loss,* dan *validation loss* pada model A-16-1, A-16-2 dan A-16-5 akan digunakan sebagai model terbaik yang ada pada *batch size* 16. Sedangkan model R-32-11, R-32-5, dan R-32-10 sebagai model terbaik pada *batch size* 32.

Ketika dilakukan pelatihan, model yang menggunakan kedua ukuran *batch* hampir menghasilkan pola yang sama seperti tampak pada gambar 19. Hanya tedapat sedikit perbedaan pada *epoch* awal kedua ukuran *batch* yang dapat diabaikan karena keduanya mampu mencapai titik akurasi maksimal dalam kurun waktu yang singkat.

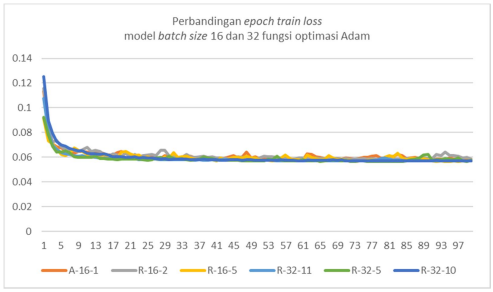


Gambar 19. Perbandingan nilai *train accuracy* fungsi optimasi Adam dengan *batch size* 16 dan 32 pada setiap *epoch*

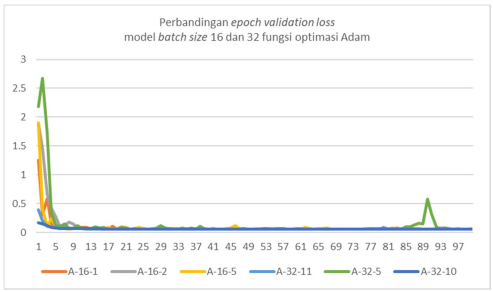


Gambar 20. Perbandingan nilai *validation accuracy* fungsi optimasi Adam dengan *batch size* 16 dan 32 pada setiap *epoch*

Sedangkan ketika dilakukan validasi data prediksi di gambar 20 terlihat bahwa pada data latih yang digunakan, *batch size* 16 memberikan proses pembelajaran yang lebih baik karena model dapat mengambil pola dan mengaturnya secara berkala tanpa menimbulkan ketimpangan gradien antar parameter. Jika kondisi tersebut ditemui, maka akan menghambat proses pembelajaran dari model yang menggunakan fungsi optimasi Adam. Pada grafik gambar 19 model melakukan eksplorasi data dengan melakukan percobaan untuk melihat nilai *loss* yang dapat ditimbulkan sehingga dapat dilakukan penyesuaian di iterasi *epoch* selanjutnya. Sedangkan eksplorasi yang dilakukan oleh *batch size* 32 tidak terlalu banyak sehingga fungsi optimasi Adam kurang bekerja secara optimal.



Gambar 21. Perbandingan nilai *train loss* fungsi optimasi Adam dengan *batch size* 16 dan 32 pada setiap *epoch*



Gambar 22. Perbandingan nilai *validation loss* fungsi optimasi Adam dengan *batch size* 16 dan 32 pada setiap *epoch*

Untuk hasil perolehan *loss function* antara kedua jenis *batch size*, keduanya menghasilkan perolehan yang baik. Hasil yang ditampilkan pada gambar 21 dan 22 diatas menunjukan bahwa model secara konsisten dapat mengurangi tingkat kesalahan prediksi yang dilakukan. Proses pengurangan nilai *loss* dengan menggunakan data latih dan validasi dapat melakukan pengenalan dengan baik, ditunjukan pada grafik yang berjalan secara konsisten antar keduanya. Namun, terdapat kemungkinan kondisi *sharp minima* pada salah satu model *batch size* 32 yaitu model A-32-5. Fenomena *sharp minima* adalah ketika model mampu mengurangi *loss function* secara drastis, tetapi akan terjadi kenaikan ketika iterasi *epoch* dilanjutkan.

Setelah dilakukan perbandingan menggunakan parameter uji *train accuracy, validation accuracy, train loss*, dan *validation loss* selanjutnya akan dilakukan pengujian dengan data subset "test" untuk melihat hasil skor prediksi yang didapatkan dengan data subset baru. Proses pengujian ini dilakukan dengan menggunakan *confussion matrix* untuk menilai skor dari masing-masing kategori pada yaitu *accuracy, precision, spesificity, recall* dan nilai *F1 score*. Hasil pengujian menggunakan *confussion matrix* dibedakan menjadi dua kategori yaitu makro, dan mikro. Nilai uji makro menggambarkan kinerja model secara keseluruhan, sedangkan mikro untuk melihat kinerja model pada masing-masing kelas yang ada. Berikut adalah hasil pengujian *confussion matrix* mikro ditampilkan pada tabel 12.

Tabel 12. Hasil pengujian mikro pada model terpilih

No	*Model name*	*Precision*			*Spesificity*			*Recall*			*F1-score*		
		0	1	2	0	1	2	0	1	2	0	1	2
1.	R-16-9	0.95	1.00	0.83	0.97	1.00	0.95	1.00	0.85	1.00	0.98	0.92	0.91
2.	R-16-8	0.74	1.00	0.83	0.77	1.00	0.97	1.00	0.85	0.50	0.85	0.92	0.62
3.	R-16-7	0.74	0.94	0.83	0.77	0.97	0.97	1.00	0.80	0.50	0.85	0.86	0.62
4.	R-32-8	1.00	0.91	1.00	1.00	0.96	1.00	0.99	0.95	0.95	0.99	0.95	0.95
5.	R-32-10	0.95	1.00	1.00	0.93	1.00	1.00	1.00	0.90	1.00	0.98	0.95	1.00
6.	R-32-12	0.98	1.00	1.00	0.97	1.00	1.00	0.95	1.00	1.00	0.99	0.97	1.00
7.	A-16-1	0.91	1.00	1.00	0.93	1.00	1.00	1.00	0.90	1.00	0.95	0.95	1.00
8.	A-16-2	1.00	0.95	1.00	1.00	0.97	1.00	1.00	1.00	0.90	1.00	0.98	0.95
9.	A-16-5	0.95	1.00	0.82	0.97	1.00	0.95	1.00	0.90	0.90	0.98	0.95	0.86
10.	A-32-11	0.98	1.00	0.91	0.97	1.00	0.98	1.00	0.90	1.00	0.99	0.95	0.95
11.	A-32-5	0.95	1.00	0.90	0.93	1.00	0.98	1.00	0.90	0.90	0.98	0.95	0.90
12.	A-32-10	0.98	1.00	0.91	0.97	1.00	0.98	1.00	0.90	1.00	0.99	0.95	0.95

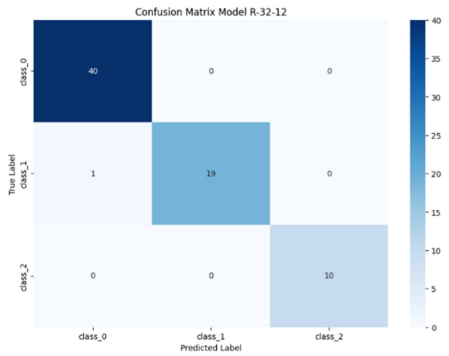
Data hasil pengujian mikro menunjukkan bahwa secara garis besar semua model dapat melakukan identifikasi atau klasifikasi terhadap data yang diberikan secara baik. Terlihat dari nilai benar prediksi (precision) lebih dari 70 persen. Namun, untuk melihat kinerja mikro secara keseluruhan dilakukan dengan membandingkan nilai *F1-score*. Karena nilai ini akan melihat kemampuan prediksi benar model dengan cakupan prediksi model benar yang sesuai dengan kategori asli data.

Diperoleh bahwa model yang memiliki kinerja kurang baik untuk klasifikasi *class 2* yaitu model R-16-8 dan R-16-7. Selanjutnya model yang belum dapat secara optimal melakukan klasifikasi dalam kelas yang sama yaitu model A-16-5, R-16-9, A-32-5. Ketiganya masih terdapat *inconsistencies* ketika dihadapkan dengan data *class 2*. Karena nilai bentukan model yang tampil tergolong baik pada sisa model yang masih ada, akan dilihat nilai pengujian secara makro untuk melihat kinerja model secara keseluruhan. Hasil perolehan pengujian makro yang dilakukan pada model terpilih ditampilkan pada tabel 13.

Tabel 13. Hasil perolehan pengujian makro model terpilih

No	*Model name*	*Accuracy*	*Precision*		*Precision*		*F1-score*		*Support*
			Unadjusted	*Unadjusted*	*Unadjusted*	*Unadjusted*	*Unadjusted*	*Adjusted*	
1	R-16-9	0.94	0.93	0.95	0.95	0.94	0.93	0.94	50
2	R-16-8	0.84	0.86	0.86	0.78	0.84	0.80	0.83	50
3	R-16-7	0.82	0.84	0.84	0.77	0.82	0.78	0.81	50
4	R-32-8	0.97	0.97	0.97	0.96	0.97	0.96	0.97	70
5	R-32-10	0.97	0.98	0.97	0.97	0.97	0.97	0.97	70
6	R-32-12	0.99	0.99	0.99	0.98	0.99	0.99	0.99	70
7	A-16-1	0.96	0.97	0.96	0.97	0.96	0.97	0.96	50
8	A-16-2	0.98	0.98	0.98	0.97	0.98	0.97	0.98	50
9	A-16-5	0.94	0.92	0.94	0.93	0.94	0.93	0.94	50
10	A-32-11	0.97	0.96	0.97	0.97	0.97	0.96	0.97	70
11	A-32-5	0.96	0.95	0.96	0.93	0.96	0.94	0.96	70
12	A-32-10	0.97	0.96	0.97	0.97	0.97	0.96	0.97	70

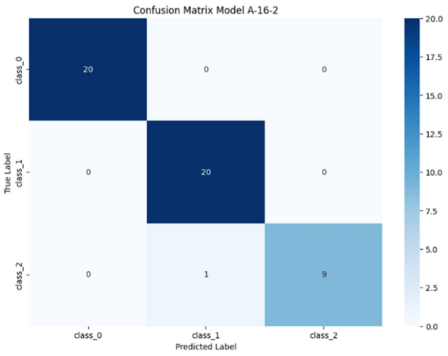
Dari tabel 13 ditampilkan nilai akurasi atau kinerja model secara keseluruhan terhadap kelas yang ada dengan nilai *precision, recall,* dan *f1-score* secara makro. Nilai tersebut akan dihitung tanpa melihat jumlah data uji (Unajusted) yang ada, dan melakukan pembobotan sesuai dengan jumlah data uji (adjusted). Dengan pertimbangan yang telah dilakukan pada hasil pengujian mikro dan makro, dapat disimpulkan bahwa model yang memiliki kinerja secara keseluruhan dengan baik terdapat tiga buah model. Secara berurutan, model tersebut adalah R-32-12, A-16-2 dan R-32-10. Alasan pemilihan ketiga model tersebut dilihat berdasarkan hasil yang didapatkan pada nilai mikro untuk menilai prediksi sesuai dengan kelas yang ditentukan. Berikut adalah tampilan *confussion matrix* dari ketiga jenis model dimulai dengan A-32-12 pada gambar 23.



Gambar 23. Hasil *confussion matrix* model R-32-12

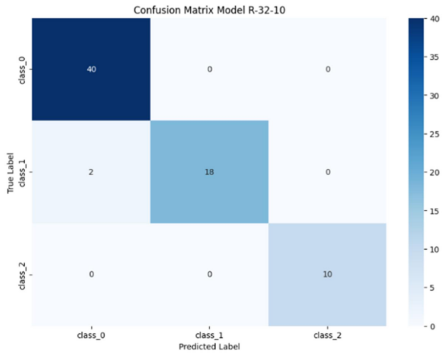
Dari gambar diatas terlihat bahwa model mampu melakukan klasifikasi data dengan benar pada *class 0* dan *class 2* dengan kesalahan klasifikasi pada *class 1* sebanyak 1 buah data dari total uji pada kelas tersebut sebanyak 20. Hal ini menunjukan bahwa model yang dibuat mampu mengenali jenis penyakit dengan baik dan salah

dalam deteksi dalam 1 percobaan dari jumlah total 20 percobaan pada penyakit *class_1*.



Gambar 24. Hasil *confussion matrix* model A-16-2

Untuk model A-16-2 mampu melakukan klasifikasi pada *class 0* dan *class 1* secara keseluruhan, tetapi terdapat kesalahan pada *class 2* sebanyak 1 buah data dari total 10 data uji *class 2* yang ditampilkan pada gambar 24. Hal ini sudah tergolong baik, mengingat data yang terdapat pada jenis penyakit *class_2* hanya berjumlah 10 dengan kesalahan uji 1 kal. Sehingga menghasilkan kinerja yang serupa dengan *class* lainnya yaitu dengan perbandingan 1 kesalahan dari total data.



Gambar 25. Hasil *confussion matrix* model R-32-10

Sedangkan untuk model R-32-10 seperti pada gambar 25, memiliki kesalahan pada *class* 1 sebanyak 2 buah data dari total data uji *class 1* berjumlah 20. Namun, tidak adanya kesalahan yang terjadi ketika melakukan klasifikasi pada *class 0* dan *class* 2 ditampilkan pada gambar 24. Melihat hasil diatas, didapatkan model R-32-12 memiliki kemampuan identifikasi secara keseuluruhan baik yaitu 0.99. Diikuti oleh A-16-2 dengan 0.98 dan R-32-10 dengan 0.97. Hasil tersebut telah dapat menyaingi dari beberapa penelitian sebelumnya dengan studi kasus pada *cacao*. Seperti pada penelitian [35] yang menghasilkan akurasi sebesar 0.98 dan pada penelitian [36] dengan angka 0.99.

Untuk mendukung pemilihan model, juga dilihat nilai rata-rata harmonis antara *precision* dan *recall* atau disebut juga dengan F1-*score*. Ketiganya memiliki nilai yang hampir sama yaitu R-32-12 sebesar 0.99, A-16-2 sebesar 0.98, dan R-32-10 sebesar 0.97. Apabila diperlukan sebegai model alternatif, penggunaan model A-16-2 dan R-32-10 dapat dilakukan sebagai data pembanding.

5. DISKUSI

Penelitian pembuatan model untuk identifikasi penyakit pada buah kakao sebelumnya sudah pernah dilakukan. Seperti yang dilakukan oleh Anissa Fitri, dkk yang meneliti tentang penerapan *convolutional neural networks* untuk mengidentifikasi penyakit pada daun buah kakao khususnya *Vascular Streak Dieback* (VSD). Model yang digunakan adalah berupa 4 jenis model yaitu AlexNet, SqueezeNet, Darknet, dan modifikasi CNN dengan total *dataset* 1200 citra (600 data daun sehat, 600 data daun terinfeksi). Dari keempat model yang digunakan masing-masing mendapatkan hasil berurutan yaitu 97,78%, 97,5%, 98,61%, dan 94,17%. Sehingga diperoleh hasil akurasi maksimal dengan menggunakan model DarkNet-19 dengan hasil akurasi 98,61% [35].

Penelitian serupa juga dilakukan oleh Nurul maulidah, dkk dengan topik identifikasi penyakit *water-sprouts* pada tumbuhan kakao. *Water sprouts* merupakan jamur yang tumbuh pada bagian cabang pada tumbuhan kakao secara cepat menyebabkan buah kakao mengalami kekurangan nutrisi. Untuk mengatasinya Nurul maulidah, dkk membuat sebuah model menggunakan model Mask R-CNN dengan jumlah *dataset* sejumlah 150 citra (120 data training, 30 data testing). Hasil penelitian diperoleh dengan nilai F1-score sebesar 0.907 [5].

Berdasarkan penelitian diatas, penelitian yang dilakukan sudah dapat mencapai nilai akurasi yang sama atau lebih baik dari beberapa arsitektur dan pendekatan yang pernah dilakukan. Dengan menggunakan kombinasi arsitektur *Inception* dan ResNet dilakukan pendekatan *fusion approach* menghasilkan tingkat akurasi sebesar 0.99 dan F1-*score* sebesar 0.99. Model yang sudah terbuat selanjutnya dapat diimplementasikan menjadi sebuah aplikasi *mobile* untuk memudahkan para petani dalam mengenai penyakit-penyakit yang ada pada buah *cacao*. Untuk memperkuat pengenalan, dalam penerapan juga dapat dikombinasikan dengan menggunakan pengenalan model lain seperti identifikasi citra termasuk kedalam kategori objek lain atau buah *cacao*.

Selama proses penelitian ditemukan kendala bahwa dalam melakukan pengujian kecocokan arsitektur memerlukan sumber daya yang besar. Hal ini dilakukan untuk menguji landasan awal yang dapat menjadi titik optimalisasi penelitian. Selain itu, *datasets* yang telah dilakukan *pre-processing* memiliki jumlah tergolong sedikit pada *class_2* yang menyebabkan hasil menjadi bias dan kurang akurat karena informasi yang didapatkan kurang maksimal pada salah satu kelas penyakit. Selain pada data yang digunakan, modul terbaru tensorflow yang

digunakan dengan dukungan CUDA pada kartu grafis Nvidia hanya mendukung sistem operasi linux. Sehingga, proses dilakukan dengan menggunakan alat bantu WSL atau *Windows Subsystem Linux* untuk mempermudah jalannya penelitian karena mayoritas proses yang telah dilakukan ada pada sistem operasi Windows.

Hasil akhir dari penelitian ini diperoleh perbedaan implementasi arsitektur yang telah dilakukan modifikasi dapat menambah ruang pengetahuan dalam topik yang serupa dalam membuat model idenfitikasi agar tetap dapat berkembang dan melakukan eksplorasi lebih lanjut pada penelitian selanjutnya. Pengembangan selanjutnya juga dapat dilakukan dengan mengganti jenis data yang digunakan yang bermula menggunakan data citra RGB (Red Green Blue) menjadi data yang diambil menggunakan teknologi kamera multispektral. Hal tersebut dapat dilakukan untuk membantu memaksimalkan potensi dalam deteksi penyakit yang lebih akurat dan pengembangan arsitektur baru.

5. KESIMPULAN

Model CNN yang dibuat penggabungan arsitektur *InceptionV3* dan *ResNet-50* menggunakan pendekatan *hybrid architecture* atau *architecture fusion*. Proses penggabungan dilakukan dengan mengambil komponen utama pada arsitektur *InceptionV3* dengan komponen dan kekuatan yang ada pada *ResNet-50*. Dalam proses pembuatan arsitektur, digunakan arsitektur *InceptionV3* sebagai *backbone architecture* karena arsitektur *InceptionV3* sangat baik dalam melakukan pengenalan *features* pada *datasets* buah kakao dibantu dengan percepatan dan efisiensi arsitektur *ResNet-50*. Hal ini menjadi alternatif penerapan kombinasi yang dilakukan pada implementasi kedua arsitektur yaitu menerapkannya menggunakan teknik berbeda yaitu *fusion approach*. Selain itu penelitian ini juga membuktikan meskipun berbeda dalam menerapkan kombinasi antar keduanya tetap dapat menghasilkan kinerja yang baik dan optimal.

Untuk mengatasi permasalahan *generalization* dan *overfitting*, digunakan implementasi inisialisasi *Glorot* dan *He* serta *CategoricalFocalCrossEntropy* dengan mengatur pembobotan kelas. Penggunaan pendekatan *fusion architecture* pada arsitektur *InceptionV3* dan *ResNet-50* membantu proses kecepatan pelatihan. Hasil terbaik dapat diperoleh tanpa memerlukan waktu berlebih untuk melakukan pelatihan sesuai dengan total *epoch* yang telah ditentukan.

Fungsi optimasi Adam sangat sensitif terhadap penetapan nilai *hyperparameter* pembuatan model, dibandingkan dengan menggunakan fungsi optimasi RMSProp. Ukuran *batch* dan *learning rate* yang digunakan pada proses pelatihan memengaruhi lama proses pelatihan, jumlah sumber daya yang digunakan untuk melakukan pelatihan, serta hasil perolehan dari sebuah model pada kedua jenis fungsi optimasi. Nilai *learning rate* 0.01 membuat proses iterasi *epoch* pada kedua jenis fungsi optimasi sulit untuk menemukan titik konvergen, sehingga hasil perolehan fluktuatif. Pengujian model dilakukan menggunakan *confusion matrix* pada hasil model terbaik yang terbentuk dan menghasilkan nilai *precision* sebesar 0.99, nilai *recall* sebesar 0.99, nilai *specificity* sebesar 0.99, dan nilai F1-*score* sebesar 0.99. Konfigurasi model yang digunakan adalah nilai *learning rate* 0.0001, fungsi optimasi RMSProp, fungsi inisialisasi (x) *He uniform*, dan fungsi inisialisasi (y) *Glorot normal*.

Model yang sudah didapatkan nantinya dapat diintegrasikan dalam sebuah sistem identifikasi penyakit buah *cacao*. Sistem ini dapat berupa sebuah aplikasi utama berbasis *mobile* untuk melakukan pengenalan dan sebuah aplikasi *dashboard* untuk mengelola dari sisi administator. Sehingga dapat membantu meningkatkan identifikasi sejak awal penyakit pada buah kakao dan mengurangi dampak negatif penyakit yang dapat menyebar ke tanaman lain. Untuk pengembangan selanjutnya dapat digunakan kombinasi model untuk identifikasi jenis objek pada citra, pengenalan penyakit tanaman kakao secara menyeluruh, hingga penggantian jenis data masukan menggunakan teknologi kamera multispektral untuk lebih akurat dalam pengenalan penyakit karena dapat menangkap data yang tidak dapat dilakukan oleh kamera biasa.

DAFTAR PUSTAKA

[1] L. Armengot, L. Ferrari, J. Milz, F. Velásquez, P. Hohmann, dan M. Schneider, "Cacao agroforestry systems do not increase pest and disease incidence compared with monocultures under good cultural management practices," *Crop Prot.*, vol. 130, hal. 105047, 2020, doi: 10.1016/j.cropro.2019.105047.

[2] L. Nesi Bria, "Penentuan Posisi Ekspor Kakao Indonesia Menurut Sembilan Negara Tujuan Di Pasar Internasional," *KAPITA J. Agribisnis Pembang. Pertan.*, vol. 1, no. 2, hal. 58–66, 2022, doi: 10.52562/kapita.v1i2.386.

[3] J. P. Marelli *et al.*, "Chocolate under threat from old and new cacao diseases," *Phytopathology*, vol. 109, no. 8, hal. 1331–1343, 2019, doi: 10.1094/PHYTO-12-18-0477-RVW.

[4] C. Rodriguez, O. Alfaro, P. Paredes, D. Esenarro, dan F. Hilario, "Machine Learning Techniques in the Detection of Cocoa (Theobroma cacao L.) Diseases," *Annalsofrscb.Ro*, vol. 25, no. 3, hal. 7732–7741, 2021, [Daring]. Tersedia pada: http://annalsofrscb.ro/index.php/journal/article/view/2317

[5] N. Maulidah, Indrabayu, dan I. S. Areni, "Water Sprouts Detection of Cacao Tree Using Mask Region-based Convolutional Neural Network," 2020, vol. 4, no. 27, hal. 1–16. doi: 10.1109/ICT49546.2020.09239443.

[6] S. Kumi, D. Kelly, J. Woodstuff, R. K. Lomotey, R. Orji, dan R. Deters, "Cocoa Companion: Deep Learning-Based Smartphone Application for Cocoa Disease Detection," *Procedia Comput. Sci.*, vol. 203, hal. 87–94, 2022, doi: 10.1016/j.procs.2022.07.013.

[7] G. Latif, S. E. Abdelhamid, R. E. Mallouhy, J. Alghazo, dan Z. A. Kazimi, "Deep Learning Utilization in Agriculture: Detection of Rice Plant Diseases Using an Improved CNN Model," *Plants*, vol. 11, no. 17, 2022, doi: 10.3390/plants11172230.

[8] Y. Chen *et al.*, "Classification of lungs infected COVID-19 images based on inception-ResNet," *Comput. Methods Programs Biomed.*, vol. 225, hal. 107053, 2022, doi: 10.1016/j.cmpb.2022.107053.

[9] P. Sharma, Y. P. S. Berwal, dan W. Ghai, "Performance analysis of deep learning CNN models for disease detection in plants using image segmentation," *Inf. Process. Agric.*, vol. 7, no. 4, hal. 566–574, 2020, doi: 10.1016/j.inpa.2019.11.001.

[10] L. Trihardianingsih, A. Sunyoto, dan T. Hidayat, "Classification of Tea Leaf Diseases Based on ResNet-50 and Inception V3," *Sinkron*, vol. 8, no. 3, hal. 1564–1573, 2023, doi: 10.33395/sinkron.v8i3.12604.

[11] R. Wightman, H. Touvron, dan H. Jégou, "ResNet strikes back: An improved training procedure in timm," *Comput. Res. Repos.*, vol. abs/2110.0, hal. 1–22, 2021, [Daring]. Tersedia pada: http://arxiv.org/abs/2110.00476

[12] M. Ahmed dan A. Ahmed, "Palm tree disease detection and classification using residual network and transfer learning of inception ResNet," *PLoS One*, vol. 18, no. 3 March, hal. 1–19, 2023, doi: 10.1371/journal.pone.0282250.

[13] M. T. Ahad, Y. Li, B. Song, dan T. Bhuiyan, "Comparison of CNN-based deep learning architectures for rice diseases classification," *Artif. Intell. Agric.*, vol. 9, hal. 22–35, 2023, doi: 10.1016/j.aiia.2023.07.001.

[14] V. Maeda-Gutiérrez *et al.*, "Comparison of convolutional neural network architectures for classification of tomato plant diseases," *Appl. Sci.*, vol. 10, no. 4, 2020, doi: 10.3390/app10041245.

[15] C. Song *et al.*, "Maize seed appearance quality assessment based on improved Inception-ResNet," *Front. Plant Sci.*, vol. 14, no. August, hal. 1–13, 2023, doi: 10.3389/fpls.2023.1249989.

[16] R. K. Shukla dan A. K. Tiwari, "Masked Face Recognition Using MobileNet V2 with Transfer Learning," *Comput. Syst. Sci. Eng.*, vol. 45, no. 1, hal. 293–309, 2023, doi: 10.32604/csse.2023.027986.

[17] S. R. Shah *et al.*, "Comparing Inception V3, VGG 16, VGG 19, CNN, and ResNet 50: A Case Study on Early Detection of a Rice Disease Syed," *Agronomy*, vol. 13, no. 6, hal. 1–13, 2023, doi: https://doi.org/10.3390/agronomy13061633.

[18] R. F. Mansour dan N. O. Aljehane, "An optimal segmentation with deep learning based inception network model for intracranial hemorrhage diagnosis," *Neural Comput. Appl.*, vol. 33, no. 20, hal. 13831–13843, 2021, doi: 10.1007/s00521-021-06020-8.

[19] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2015, vol. June, hal. 1–9. doi: 10.1109/CVPR.2015.7298594.

[20] K. He, X. Zhang, S. Ren, dan J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016, vol. 2016-Decem, hal. 770–778. doi: 10.1109/CVPR.2016.90.

[21] Z. Sheng, H. Wang, G. Chen, B. Zhou, dan J. Sun, "Convolutional residual network to short-term load forecasting," *Appl. Intell.*, vol. 51, no. 4, hal. 2485–2499, 2021, doi: 10.1007/s10489-020-01932-9.

[22] R. C. Gonzalez dan R. E. Woods, *4TH EDITION Digital image processing*, 4 ed. New York: Pearson, 2018.

[23] H. S. Obaid, S. A. Dheyab, dan S. S. Sabry, "The Impact of Data Pre-Processing Techniques and Dimensionality Reduction on the Accuracy of Machine Learning," *IEEE Access*, hal. 279–283, 2019, doi: 10.1109/IEMECONX.2019.8877011.

[24] J. Liang, Y. Liu, dan V. Vlassov, "The Impact of Background Removal on Performance of Neural Networks for Fashion Image Classification and Segmentation," in *Proceedings - 2023 Congress in Computer Science, Computer Engineering, and Applied Computing, CSCE 2023*, 2023, hal. 1960–1968. doi: 10.1109/CSCE60160.2023.00323.

[25] L. M. Pereira, A. Salazar, dan L. Vergara, "A Comparative Analysis of Early and Late Fusion for the Multimodal Two-Class Problem," *IEEE Access*, vol. 11, hal. 84283–84300, 2023, doi:

10.1109/ACCESS.2023.3296098.

[26] S. Papadopoulos, G. Koukiou, dan V. Anastassopoulos, "Decision Fusion at Pixel Level of Multi-Band Data for Land Cover Classification—A Review," *J. Imaging*, vol. 10, no. 1, 2024, doi: 10.3390/jimaging10010015.

[27] O. Chandraumakantham, N. Gowtham, M. Zakariah, dan A. Almazyad, "Multimodal Emotion Recognition Using Feature Fusion: An LLM-Based Approach," *IEEE Access*, vol. 12, hal. 108052–108071, 2024, doi: 10.1109/ACCESS.2024.3425953.

[28] T. Jiao, C. Guo, X. Feng, Y. Chen, dan J. Song, "A Comprehensive Survey on Deep Learning Multi-Modal Fusion: Methods, Technologies and Applications," *Comput. Mater. Contin.*, vol. 80, no. 1, hal. 1–35, 2024, doi: 10.32604/cmc.2024.053204.

[29] M. A. Ganaie, M. Hu, A. K. Malik, M. Tanveer, dan P. N. Suganthan, "Ensemble deep learning: A review," *Eng. Appl. Artif. Intell.*, vol. 115, no. August, hal. 1–45, 2022, doi: 10.1016/j.engappai.2022.105151.

[30] Y. S. Taspinar, I. Cinar, dan M. Koklu, "Classification by a stacking model using CNN features for COVID-19 infection diagnosis," *J. Xray. Sci. Technol.*, vol. 30, no. 1, hal. 73–88, 2022, doi: 10.3233/XST-211031.

[31] G. Zeng, "On the confusion matrix in credit scoring and its analytical properties," *Commun. Stat. - Theory Methods*, vol. 49, no. 9, hal. 2080–2093, 2020, doi: 10.1080/03610926.2019.1568485.

[32] D. Guest dan P. Keane, "Vascular-streak dieback: A new encounter disease of cacao in Papua New Guinea and Southeast Asia caused by the obligate basidiomycete Oncobasidium theobromae," *Phytopathology*, vol. 97, no. 12, hal. 1654–1657, 2007, doi: 10.1094/PHYTO-97-12-1654.

[33] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, dan Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016, vol. December, hal. 2818–2826. doi: 10.1109/CVPR.2016.308.

[34] P. Bansal, R. Krueger, M. Bierlaire, R. A. Daziano, dan T. H. Rashidi, "Bayesian estimation of mixed multinomial logit models: Advances and simulation-based evaluations," *Transp. Res. Part B Methodol.*, vol. 131, no. December, hal. 124–142, 2020, doi: 10.1016/j.trb.2019.12.001.

[35] A. F. M. Harvyanti, R. I. Baihaki, Dafik, Z. R. Ridlo, dan I. H. Agustin, *Application of Convolutional Neural Network for Identifying Cocoa Leaf Disease*, vol. 2. Atlantis Press International BV, 2023. doi: 10.2991/978-94-6463-174-6_21.

[36] K. J. Ayikpa, D. Mamadou, P. Gouton, dan K. J. Adou, "Classification of Cocoa Pod Maturity Using Similarity Tools on an Image Database: Comparison of Feature Extractors and Color Spaces," *Data*, vol. 8, no. 6, 2023, doi: 10.3390/data8060099.