Pemanfaatan Transfer Learning Pada CNN Untuk Pendeteksian Malaria

Praktik Kerja Lapangan

Benedictus Jullian Pradana
311810007

TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS MA CHUNG
2022

LEMBAR PENGESAHAN

PEMANFAATAN TRANSFER LEARNING PADA CNN UNTUK PENDETENSIAN MALARIA

Oleh:

BENEDICTUS JULLIAN PRADANA

311810**07**

dari

TEKNIK INFORMATIKA

FAKULTAS SAINS DAN TEKNOLOGI

UNIVERSITAS MA CHUNG

Dosen Pembimbing

Windra Swastika, S.Kom., MT., **Ph.D.**

20070039

Dekan Fakultas Sains dan Teknologi

Dr. Kestrilia Rega Prilianti, S.Si., M.Si.

20120035

Kata Pengantar

Puji dan syukur dipanjatkan kepada Tuhan Yang Maha Esa atas karunia-Nya, laporan Praktik Kerja Lapangan (PKL) ini, dapat dibuat dan diselesaikan dengan baik. Laporan dengan judul "Pengaruh Transfer Learning Pada CNN Untuk Pendeteksian Malaria" disusun berdasarkan apa yang telah dilakukan pada saat praktik kerja lapangan selama 4 bulan mulai 10 September 2021 hingga 10 Desember 2021 di Pusat Studi Artificial Intelegence in Medical Image and Entrepreneurship.

Dalam penyusunan laporan praktik kerja lapangan ini, Penulis memberikan ucapan terima kasih kepada berbagai pihak yang sudah membantu dalam proses pembuatan laporan ini, antara lain:

1. Orang tua dan teman-teman yang selalu memberi dukungan dan semangat kepada Penulis sehingga praktik kerja lapangan dapat terselesaikan,
2. Ibu Dr. Kestrilia Rega Prilianti, M.Si. selaku Dekan Fakultas Sains dan Teknologi Universitas Ma Chung,
3. Bapak Hendry Setiawan, S.T., M.Kom. selaku Kepala Program Studi Teknik Informatika Universitas Ma Chung,
4. Bapak Dr.Eng. Romy Budhi, ST., MT. selaku pembimbing akademik,
5. Bapak Windra Swastika, S.Kom., MT., Ph.D. selaku pembimbing praktik kerja lapangan.

Laporan ini merupakan salah satu prasyarat kelulusan sebagai sarjana dan mata kuliah program kerja lapangan. Dalam pengerjaan laporan ini, masih terdapat kekurangan. Oleh sebab itu, kritik dan saran sangat membantu bagi penulis untuk memperbaiki kekurangan yang ada. Demikian semoga laporan PKL ini bisa bermanfaat bagi semua pihak.

Malang, 6 Januari 2022

Benedictus Jullian Pradana

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

Bab I
Pendahuluan

1.1 Latar Belakang

Berdasarkan laporan WHO yang terbaru, pada tahun 2019 terdapat 229 juta kasus malaria di seluruh dunia dan kasus kematian akibat malaria adalah 400.900 jiwa. Golongan yang paling rentan terjangkit malaria adalah anak yang berusia di bawah lima tahun dengan presentase 67% dari total kematian (274.000).Sebaran malaria pada tahun 2019 masih berpusat pada daerah afrika dengan presentase kasus 94% dari total kasus (WHO, World Malaria Report, 2020). Indonesia menempati peringkat ke dua untuk jumlah kasus malaria tertinggi di Asia Tenggara.Kasus malaria di Indonesia sempat menuun pada tahun 2010 – 2014 namun menjadi stagnan pada tahun 2014-2019. (WHO, World Malaria Report, 2020).

Berdasarkan data kemenkes pada tahun 2019 terdapat 250.644 kasus malaria ,mayoritas kasus berada di Papua dengan presentase 86% (216.380) disusul dengan Provinsi Nusa Tenggara Timur berjumlah 12.909 kasus. Sementara itu, terdapat sekitar 300 kabupaten dan kota yang telah memasuki kategori eliminasi, atau sekitar 208,1 juta penduduk (77,7%). Provinsi yang terbebas malaria atau terdapat 0 kasus antara lain adalah Provinsi DKI Jakarta, Provinsi Jawa Timur, dan Provinsi Bali (KemenkesRI, 2019).

Selama dua dekade terakhir, WHO telah membuat rekomendasi tentang bagaimana mencegah malaria yaitu *vector control, preventive chemotherapies*, dan vaksinasi malaria. *Vector control* adalah upaya pencegahan malaria dengan menggunakan jarring *insecticide-treated nets(*ITNs) atau sejenis jarring yang dibuat secara khusus untung menghalau nyamuk secara fisik, *preventive chemotherapies* adalah upaya pencegahan dengan menggunakan obat-obatan, dan yang terakhir adalah vaksinasi, Sejak awal oktober 2021 WHO telah merekomendasikan penggunakan vaksin RTS,S/AS01 secara luas untuk mencegah malaria terutama pada anak-anak (WHO, Malaria, 2021).

Malaria didiagnosa dengan cara pemeriksaan fisik dan pemeriksaan darah. Pemeriksaan darah untuk mendiagnosa malaria meliputi tes diagnostic cepat malaria (RDT malaria) dan pemeriksaan darah penderita di bawah mikroskop.Hal ini bertujuan untuk mendeteksi parasite penyebab malaria yang terdapat pada sel darah (alodokter, 2019).Diagnosa ini dilakukan secara manual oleh dokter dan jika pengalaman dokter yang mendiagnosa kurang mumpuni dapat mengakibatkan kesalahan diagnosa

Machine Learning adalah bagian dari kecerdasan buatan dimana computer dapat memecahkan masalah atau menemukan suatu pola dari data tanpa harus di program secara

eksplisit (IBM, 2020).Salah satu metode dari machine learning adalah transfer learning, metode ini diterapkan dengan cara menggunakan model machine learning yang telah dibuat sebelumnya sebagai *starting point* untuk permasalahan yang serupa. Machine learning dapat digunakan untuk mengklasifkasi sekumpulan citra untuk mendeteksi termasuk apakah citra tersebut.

Penelitian ini dilakukan untuk mendeteksi sel apakah terinfeksi malaria atau tidak dengan menerapkan transfer learning pada dataset citra malaria. Dataset citra diperoleh dari platform Kaggle. Penelitian ini menggunakan tiga model yang telah tersedia dari platform tensorflow hub yaitu Resnet50V2, InceptionV3, dan EfficientNetB0.

Sebelumnya sudah ada penelitian mengenai transfer learning pada dataset citra malaria menggunakan model ResNet50 (Reddy & Juliet, 2019). Pada penelitian tersebut diperoleh *training accuracy* 95.91% dan *validation accuracy* 95.4% sedangkan untuk *training loss* 0.1134 dan *validation loss* 0.1301 dan epoch yang digunakan sebanyak 10 epoch.

1.2 Identifikasi Masalah

Berdasarkan latar belakang yang telah disampaikan. identifikasi yang diangkat pada penelitian ini adalah menguji pengaruh transfer untuk meningkatkan akurasi model dikarenakan performa model dari penelitian sebelumnya dirasa masih kurang dan bisa ditingkatkan lagi.

1.3 Batasan Masalah

Batasan masalah dari pengerjaan penelitian ini adalah sebagai berikut.

1. Penelitian dilakukan dengan dataset citra malaria yang didapat dari platform Kaggle. Jumlah citra yang digunakan adalah 27560 citra.
2. Citra yang digunakan merupakan citra hasil pengamatan mikroskop berupa sel darah yang sehat dan sel darah yang terinfeksi
3. Model dibuat menggunakan platform tensorflow
4. Model yang digunakan untuk transfer learning adalah Resnet50V2, InceptionV3, dan EfficientNetB0
5. Model untuk transfer learning didapat dari platform tensorflow hub

1.4 Rumusan Masalah

Berdasarkan batasan masalah yang telah dikemukakan di atas maka dapat dirumuskan permasalahan penelitian ini sebagai berikut.

1. Bagaimana pengaruh transer learning hasil prediksi model untuk mendeteksi malaria
2. Seberapa tinggi akurasi model dalam mem prediksi malaria.

1.5 Tujuan

Tujuan dari pengerjaan penelitian ini adalah untuk membandingkan transfer learning dari model Resnet50V2, InceptionV3, dan EfficientNetB0 untuk membuat model klasifikasi pada dataset malaria.

1.6 Manfaat

Manfaat dari pengerjaan penelitia ini adalah sebagai berikut.

1. Bagi Universitas Ma Chung khususnya Program Studi Teknik Informatika dapat mempersiapkan lulusan yang berkompeten dan siap kerja dengan memberikan bekal kepada mahasiswa selama proses kegiatan Praktik Kerja Lapangan.
2. Bagi Mahasiswa dapat menerapkan ilmu yang telah diperoleh selama belajar di Universitas Ma Chung dan memberikan bekal pengalaman untuk masuk ke dunia kerja

Bab II
Tinjauan Pustaka

2.1 Malaria

Malaria merupakan penyakit menular mematikan yang disebabkan oleh parasit *plasmodium*. Penyakit ini disebarkan melalui gigitan nyamuk *anopheles* betina yang terinfeksi. Ada lima spesies parasit yang menyebakan malaria pada manusia, dan dua diantaranya memiliki ancaman paling mematikan yaitu *P.falciparum* dan *P.vivax.* (WHO, World Malaria Report, 2020). Malaria dapat dibedakan menjadi dua jenis yaitu biasa dan berat. Malaria biasa adalah penyakit yang tidak menimbulkan komplikasi yang parah dan hanya gejala-gejala utama, karena tidak ada organ vital yang terdampak.Gejala-gejala yang muncul akan berlangsung selama 6-10 jam dan berulang setiap 2 hari sekali, sedangkan untuk malaria berat merupakan komplikasi dari malaria biasa yang tidak segera ditangani dan akhirnya menimbulkan komplikasi (ShylmaNa'imah, 2021). Na'imah juga menjelaskan pada jenis malaria berat terjadi proses yang disebut dengan sekuestrasi, yaitu kondisi ketika darah menggumpal dan membuat sumbatan di pembuluh darah.tersebut. Pada kondisi yang lebih parah, penderita berpotensi mengalami penyakit malaria serebral, yaitu ketika infeksi P. falciparum telah memengaruhi otak. Kondisi ini dapat terjadi kurang dari 2 minggu setelah pertama kali digigit nyamuk, serta diawali dengan demam selama 2-7 hari.

Malaria dapat diobati dengan memberikan obat anti malaria kepada pasien pengidap malaria. Obat-obatan ini perlu disesuaikan dengan jenis parasite penyebab malaria, tingkat keparahan, atau Riwayat area geografis yang pernah ditinggali penderita. Penyakit malaria pada anak-anak bisa ditangani oleh dokter anak konsultan penyakit infeksi tropis (alodokter, 2019).

2.2 Artificial Intelligence

Artificial intelligence(AI) atau kecerdasan buatan adalah program komputer yang diciptakan manusia untuk mensimulasikan dan meniru kecerdasan manusia.Kecerdasan buatan diciptakan untuk membantu manusia dalam menyelesaikan masalah.Secara Dalam aplikasinya, kecerdasan buatan terbagi menjadi tujuh yaitu *Machine Learning, Natural Language Processing,Expert System,Machine Vision,Speech,Planning,* dan *Robotics* (Dicoding, 2020). Salah satu jenis kecerdasan buatan yang telah umum digunakan adalah machine learning. Machine learning merupakan subset dari kecerdasan buatan dimana program dibuat untuk

menemukan pola (*pattern)* yang terdapat pada data yang telah disediakan. Terdapat 4 jenis algoritma yaitu *Supervised Learning*, *Unsupervised Learning, Semi-Supervised Learning* dan *Reinforcement Learning* (Fumo, 2017).

Supervised Learning secara umum digunakan untuk permasalahan *Regression* dan *Classification. Supervise Learning* digunakan ketika data yang digunakan untuk memprediksi output telah diberi label, contoh algoritmanya adalah *K-Nearest Neighbour(KNN), Linear Regression, Decision Tree* dan *Support Vector Machine*. *Unsupervised Learning* digunakan ketika untuk menemukan struktur atau pola tertentu dari data yang belum memiliki label. Secara umum *Unsupervised Learning* digunakan untuk menyelesaikan permasalahan *clustering*, contoh algoritmanya adalah *K-Means Clustering* dan *Fuzzy K-Means.*

Dalam machine learning juga terdapat subset *Deep Learning*. *Deep Learning* yang menggunakan *Artificial Neural Networks(ANN)* hingga membentuk banyak lapisan *Neural Networks* atau *Deep Neural Networks. Deep Learning* terinspirasi dari cara kerja sel neuron pada otak yang saling terkoneksi satu sama lain untuk memproses informasi, contoh algoritma *Deep Learning* adalah *Convolutional Neural Networks(CNN)* dan *Recurent Neural Networks(RNN)*

2.3 Machine Learning

Machine Learning merupakan bagian dari kecerdasan buatan.Secara singkat *Machine Learning* merupakan algoritma untuk menyelesaikan permasalahan berdasarkan data dengan sendirinya tanpa di program secara eskplisit (Samuel, 1959). Secara umum algoritma *Machine Learning* terbagi menjadi 4 tipe yaitu *Supervised Learning, Unsupervised Learning, Semi-Supervised Learning* dan *Reinforcement Learning* (Bonaccorso, 2018).

Supervised Learning secara umum digunakan untuk permasalahan *Regression* dan *Classification. Supervise Learning* digunakan ketika data yang digunakan untuk memprediksi output telah diberi label, contoh algoritmanya adalah *K-Nearest Neighbour(KNN), Linear Regression, Decision Tree* dan *Support Vector Machine*. Contoh aplikasinya antara lain *predictive analysis, linear regression, classification, spam detection, pattern detection, natural language processing (nlp), sentiment analysis.*

Unsupervised Learning digunakan ketika untuk menemukan struktur atau pola tertentu dari data yang belum memiliki label. Secara umum *Unsupervised Learning* digunakan untuk menyelesaikan permasalahan *clustering*, contoh algoritmanya adalah *K-Means Clustering* dan *Fuzzy K-Means.* Contoh aplikasi *Unsupervised Learning* antara lain *object segmentation, similarity detection, automatic labeling, recommendation engines.*

Semi-Supervised Learning merupakakan metode yang digunakan ketika dataset yang digunakan adalah campuran dari data yang memiliki label dan data yang tidak memiliki label. Algoritma yang digunakan adalah perpanjangan dari metode lain (*Supervised Learning* dan *Unsupervised Learning)* yang dapat membuat model berdasarkan data yang belum memiliki label.

Reinforcement Learning berbeda dari tipe machine learning sebelumnya karena model tidak di training berdasarkan label dari dataset melainkan bagaimana *agent* berinteraksi dengan *environment* (Gudimella, et al., 2017). *Agent* merupakan entitas yang berinteraksi dengan *environment,* sedangkan *environment* merupakan situasi yang telah ditentukan. *Agent* akan terus berinteraksi dengan *environment* hingga menemukan perilaku apa yang paling sesuai pada *environment* tersebut. Salah satu algoritma yang umum digunakan untuk *Reinforcment Learning* adalah *Q-Learning.*

2.3.1 K Nearest Neighbour

K-Nearest Neighbours (KNN) merupakan klasifikasi non parametrik yang pertama kali diperkenalkan pada tahun 1951 (Fix & Hodges, 1989) dan kemudian dikembangkan Thomas Cover pada tahun 1967 (Cover & Hart, 1967) yang digunakan untuk permsalahan klasifikasi dan *regression*. KNN bekerja dengan cara memprediksi class yang tepat dari test data dengan menghitung jarak Euclidian antara test data dengan seluruh *training points* (Christopher, 2021).

Tahap pertama dalam menggunakan KNN untuk klasifikasi adalah menentukan data point. Tahap kedua adalah menentukan nilai K (jumlah neighbor dari titik point). Tahap ketiga hitung Euclidian distance dari K. Tahap keempat, antara K neighbours hitung jumlah data point untuk masing-masing kategori. Tahap kelima, masukan data point yang ditentukan pada tahap pertama menjadi kategori dengan jumlah K data point maksimum.

2.3.2 Linear Regression

Linear regerssion merupakan algorimta yang digunakan untuk predictive analysis. Secara umum regression digunakan untuk mengamati dua hal, pertama apakah sekumpulan variabel prediktor (independent variable) saling berkorelasi sehingga dapat memprediksi variabel hasil (dependent variable). Kedua, variabel manakah yang memiliki dampak signifikan sebagai prediktor variabel dalam memprediksi hasil (seberapa besar mempengaruhi variabel hasil). Hasil dari perhitungan regression digunakan untuk menjelaskan hubungan antara satu dependent variable dengan satu atau banyak independent variable (StatisticsSolutions, 2021).

2.4 Deep Learning

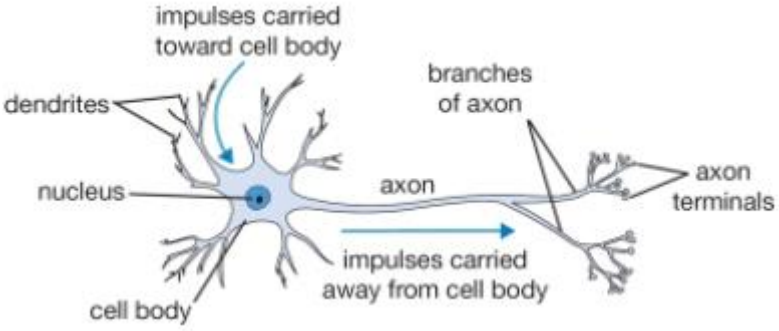
Deep Learning merupakan subset dari machine learning yang menggunakan *Artificial Neural Networks(ANN)* hingga membentuk banyak lapisan *Neural Networks* atau *Deep Neural Networks*. *Deep Learning* terinspirasi dari cara kerja sel neuron pada otak yang saling terkoneksi satu sama lain untuk memproses informasi. *Deep Learning* memungkinkan model komputasi yang terdiri dari *multi-layer processing* untuk mempelajari representasi dari data dengan *multiple-level of abstraction* (LeCun, Bengio, & Hinton, Deep learning, 2015) . LeCun juga menjelaskan bahwa *Deep Learning* menggunakan *Back Propagation* untuk menemukan pola pada struktur data dan bagaimana mesin harus mengganti parameter yang digunakan untuk melakukan komputasi pada representasi yang terdapat di setiap layer dari layer sebelumnya. Contoh algoritma *Deep Learning* adalah *Convolutional Neural Networks(CNN)* dan *Recurent Neural Networks(RNN).*

CNN atau ConvNet merupakan arsitektur untuk deep learning yang tidak perlu secara manual mengekstrak feature dari data. cnn bekerja dengan cara meniru visual korteks pada otak untuk mengenali citra (Kim, 2017). CNN sangat berguna untuk menemukan pola pada citra untuk mengenali objek. CNN juga cukup efektif untuk permasalahan yang menggunakan data non-citra seperti audio dan *time series.*

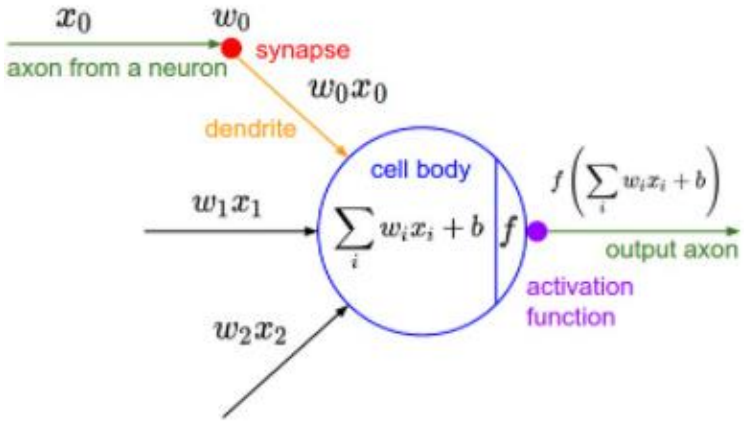
RNN merukapan perkembangan dari ANN yang memungkinan output dari layer sebelumnya digunakan sebagai input ketika memiliki *hidden state*. RNN cocok digunakan untuk menyelesaikan permasalahan time series.

2.5 Artificial Neural Network

Artificial Neural Network (ANN) diesbut juga *Neural Networks(NN)* merupakan algoritma *Deep Learning* yang menirukan cara kerja jaringan sel neuron pada otak dalam memproses informasi untuk menyelesaikan masalah. Setiap ANN tersusun dari neuron yang saling terhubung hingga membentuk *layer*. Neuron pada suatu layer akan menyalurkan informasi pada layer berikutnya untuk melakukan komputasi (Gulli, Kapoor, & Pal, 2019). NN petama kali diperkenalkan pada tahun 1950-an berupa dua layer *perceptron* (Rosenblatt, 1958), Kemudian dikembangkan lebih lanjut dengan dikenalkanya algoritma *back propagation* (Werbos, 1990) untuk membuat *training multi-layer NN* yang lebih efisien (Hinton, Osindero, & Teh, 2006).



Gambar 2. 1 gambar single cell neuron



Gambar 2. 2 Representasi matematis untuk neuron

Pada gambar 2.2 merupakan representasi matematis untuk sel neuron dalam machine learning disebut juga perceptron. *Neural Networks (NN)* didesain berdasarkan cara kerja otak, dengan cara meniru cara kerja neuron yang saling terhubung dan menyalurkan input melalui beberapa *layer* berupa sekumpulan *perceptron* (neuron). Sebelum input disalurkan ke *layer* berikutnya, input ditransformasikan terlebih dahulu menggunakan *activation function*. Berikut adalah formulanya.

$$z = \sum_{i=1}^{n} x_i w_{ij} + b_j \tag{2.1}$$

Dimana:

$x =$ *input*, adalah nilai dari *features*, secara umum berupa sekumpulan bilangan riil dalam array
$w =$ beban (*weight*), merepresentasikan koefisien scalar yang bertugas untuk mengukur seberapa penting nilai dari setiap input.
$z =$ *transfer function*, bertugas mengkombinasikan setiap input menjadi satu output
$b =$ bias

Neural Network tersusun dari empat komponen yaitu *input (x)*, beban atau *weight (w), transfer function (z),* dan bias (b). *Neural Network* memiliki dua fase, *forward propagation* dan *backward propagation* (Lecun, Bottou, Bengio, & Haffner, 1998). *Forward propagation* merupakan proses yang melibatkan perkalian *feature value* dengan *weights*, kemudian di kombinasikan pada transfer function untuk menghasilkan satu output. Output dari *transfer function* akan dihitung kembali menggunakan *Activation function* sebelum dilanjutkan ke neuron pada *layer* berikutnya.

Activation Function berfungsi untuk menambahkan *non-linearity* pada data. *Non-linearity* membantu dalam mengenali *pattern* dari data yang kompleks. *Activation function* yang umum digunakan antara lain *sigmoid function, tanh function, ReLU* dan *Leaky ReLU. Sigmoid Function* menskalakan nilai antara 0 dan 1, biasa digunakan pada *output layer* untuk *binary classification.* Berikut adalah formula untuk sigmoid.

$$f(x) = \sigma(z) = \frac{1}{1 + e^{-z}} \tag{2.2}$$

Tanh merupakan singkatan dari *Hyperbolic Tangent. Tanh function* menskalakan nilai antara -1 dan 1. Berikut adalah formula untuk *Tanh.*

$$f(x) = \tanh(x) = \frac{(e^{x} - e^{-x})}{(e^{x} + e^{-x})} \tag{2.3}$$

ReLU (Rectified Linear Unit) akan mengeluarkan output angka yang sama jika nilai x > 0 dan output 0 jika x < 0. Berikut adalah formula untuk *ReLU.*

$$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases} \quad (2.4)$$

Leaky ReLU mirip dengan *ReLU,* namun ketika nilai x < 0 akan mengembalikan nilai 0.01 dikali dengan x , akan sangat berguna jika terdapat feature value yang bernilai negatif. Berikut adalah formula untuk *Leaky ReLU.*

$$f(x) = \begin{cases} 0.01x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases} \quad (2.5)$$

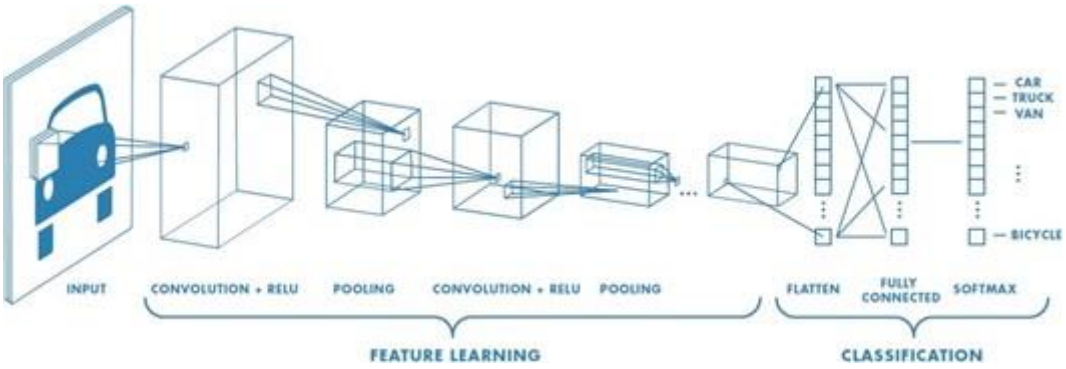
Backward Propagation merupakan proses yang dilakukan untuk menemukan nilai optimal parameter pada model dengan cara memperbaharui parameter berulang kali dengan melakukan perhitungan turunan parsial nilai gradien dari *loss function* dengan nilai *weights*. Fungsi optimasi (*Optimization Function)* digunakan saat *back-propagation* dilakukan. *Chain rule* pada kalkulus berperan penting dalam proses *back-propagation*. Berikut adalah formula turunan parsial dari *loss function* (L) dengan *weights* (w).

$$\frac{\partial L}{\partial w} = \frac{\partial z}{\partial w} \times \frac{\partial a}{\partial z} \times \frac{\partial L}{\partial a} \quad (2.6)$$

Perubahan kecil pada nilai *weights* 'w' mempengaruhi perubahan pada nilai 'z' ($\partial z/\partial w$). Perubahan kecil pada nilai 'z' mempengaruhi perubahan pada nilai aktivasi 'a' ($\partial a/\partial z$). Perubahan kecil pada nilai aktivasi 'a' mempengaruhi perubahan fungsi Loss 'L' ($\partial L/\partial a$)

2.6 Convolutional Neural Network

CNN atau ConvNet merupakan arsitektur untuk deep learning yang tidak perlu secara manual mengekstrak feature dari data. cnn bekerja dengan cara meniru visual korteks pada otak untuk mengenali citra (Kim, 2017). CNN pertama kali diperkenalkan oleh Yan LeCun pada tahun 1989 (LeCun, et al., 1989). CNN pertama kali digunakan untuk mengenali tulisan tangan. Arsitektur CNN dapat dilihat pada gambar 2.3.



Gambar 2. 3 Arsitektur CNN

Sumber (https://au.mathworks.com/discovery/convolutional-neural-network-matlab.html)

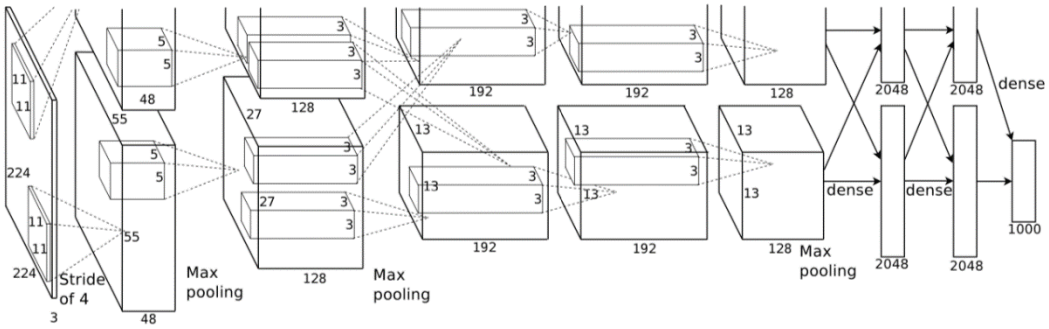
Secara umum CNN tersusun dari *convolution layers, activation functions,pooling layers,* dan *flatten layer* (Wang, et al., 2021). *Input layer* yang beraada pada bagian paling kiri merupakan input citra dari CNN. CNN menggunakan citra RGB sebagai input, sehingga *input layer* menggunakan tiga saluran yaitu untuk warna merah, hijau, dan biru.

Convolutonal layer berperan sebagai fondasi CNN, terdapat kernel (weights) yang telah dipelajari dan mengekstrak features yang membedakan citra yang satu dengan yang lain. Neuron pada *convolutional layer* (*convolutional neuron)* melakukan operasi dot product antara *kernel* dengan output dari neuron layer sebelumnya. *Convolutional neuron* juga merupakan total perhitungan dari output semua neuron sebelumnya dengan bias. Pada *Convolutional layer* terdapat tiga *hyper-parameter* yang digunakan yaitu *padding, kernel_size, dan strides. Padding* untuk menambah area di sekitar *feature map* dengan nilai nol secara simetris hal ini bertujuan agar ukuran output sama dengan input. *Kernel_size* atau disebut juga *filter_size* merupakan ukuran dari *kernel* yang melakukan ekstraksi *feature* pada *feature map*. *Strides* mengindikasikan berapa jauh (dalam hitungan pixel) setiap perpindahan kernel pada *feature map. Activation functions* yang digunakan pada *convolutional layer* adalah *ReLU. ReLU* digunakan untuk mengenalkan *non-linearity* sehingga model CNN dapat mencapai akurasi yang tinggi (He, Zhang, Ren, & Sun, 2016). *Pooling layer* berfungsi untuk mereduksi ukuran spasial dari *feature map* yang telah melalui *convolutional layer* sehingga mengurangi komputasi. *Pooling* yang digunakan terdapat dua tipe yaitu *Max Pooling* dan *Average Pooling. Max Pooling* memilih nilai tertinggi dari kernel sedangkan *Average Pooling* merata-rata seluruh nilai pada kernel. *Flatten layer* berfungsi untuk mereduksi dimensi dari vector input yang sebelumnya tiga dimensi menjadi satu dimensi yang kemudian di salurkan *fully-connected layer* untuk dilakukan klasifikasi. Pada *fully-connected layer, activation functions*

yang digunakan adalah *softmax*, *softmax* akan memastikan bahwa jumlah total dari nilai output CNN adalah satu. *Softmax* berfungsi untuk menskalakan nilai output menjadi propabilitas. Selama satu dekade terakhir terpadat tiga arsitektur cnn populer yang menjadi pioneer untuk arsitektur arsitektur baru berikutnya yaitu AlexNet, VGG, dan ResNet.

2.6.1 AlexNet

AlexNet merupakan arsitektur CNN yang diciptakan oleh Alex Krivhezky yang bekerja sama Ilya Sustkever dan Geoffry Hinton, kemudian diperkenalkan pada tahun 2012 (Krizhevsky, Sutskever, & Hinton, 2012). AlexNet berhasil memenangkan ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) pada tahun 2012. ILSVRC menggunakan 1.2 juta data training, 50 ribu data validasi dan 150 ribu data testing dengan resolusi masing-masing citra 256x256 pixel. Arsitektur AlexNet terdiri dari 8 layer, 5 diantranya adalah convolutional layer dan 3 fully-connected layer. AlexNet menjadi unik pada waktu itu karena menggunakan ReLu NonLinearity, Multi GPU, dan Overlapping PoolingI. Arsitektur AlexNet dapat dilihat pada gambar 2.4

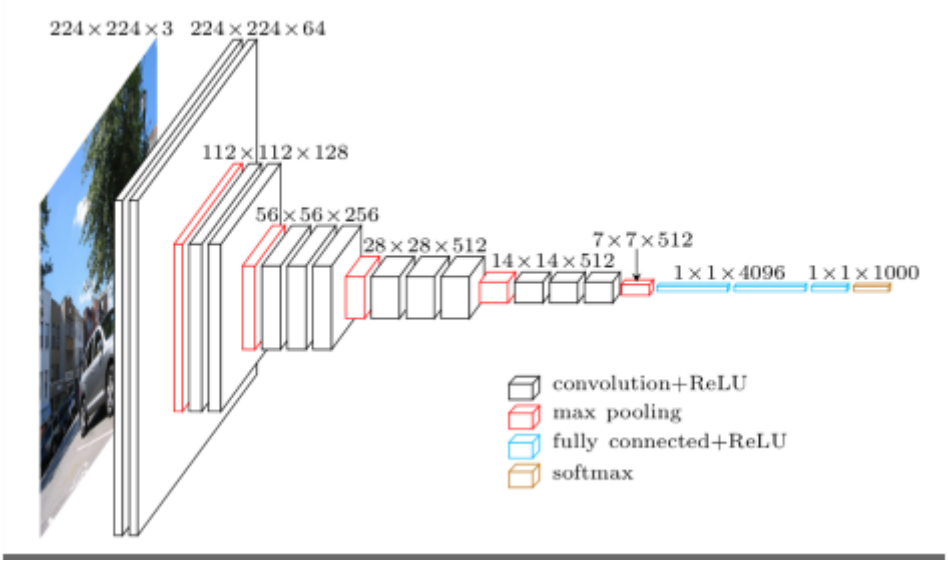


Gambar 2. 4 Arsitektur AlexNet sumber: (Krizhevsky, Sutskever, & Hinton, 2012)

AlexNet memiliki 60 juta parameter dan memiliki isu overfitting. Untuk mengurangi overfitting dua cara diimplementasikan pada AlexNet yaitu augmentasi data dan dropout. AlexNet memenangkan ILSVRC pada tahun 2012 dengan hasil top-5 error rate of 15.3%.

2.6.2 VGG

Visual Geometry Group (VGG) dikenal juga sebagai VGGNet merupakan arsitektur CNN yang diperkenalkan pada tahun 2014 oleh Karen Simonyan dan Andrew Zisserman dari universitas Oxford (Simonyan & Zisserman, 2014).VGGnet menjadi first runner up pada ILSVRC 2014. Contoh varian dari VGG adalah VGG-16 dan VGG-19. Angka 16 dan 19 pada varian VGG mengindikasikan berapa banyak layer yang digunakan. Untuk VGG-16 terdapat 16 layers dalam arsitekturnya. Arsitektur VGG-19 dapat dilihat pada gambar 2.5 dan 2.6



Gambar 2. 5 Arsitektur VGG-16

sumber: (https://www.cs.toronto.edu/~frossard/post/vgg16/)

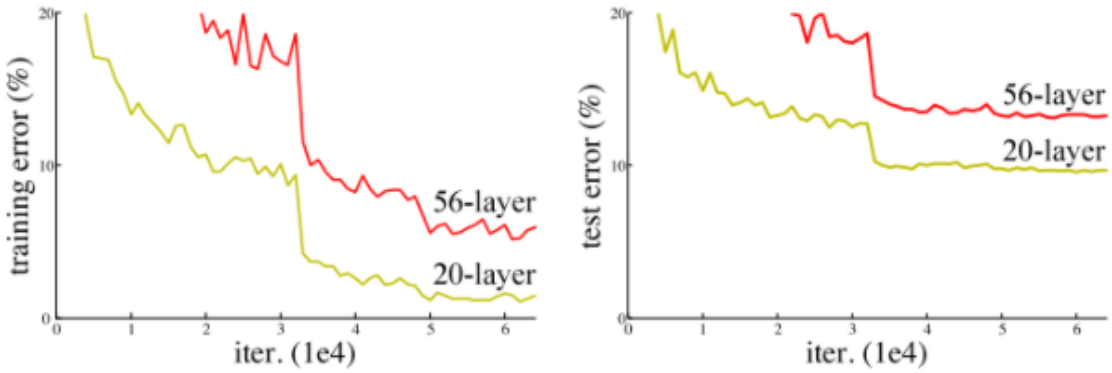
ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Gambar 2. 6 Gambar tabel arsitektur VGG

Input dari VGG-16 adalah citra dengan ukuran 224x224 pixel . Pada kompetisi ILSVRC 2014 VGG berhasil meraih 92.7% top-5 test accuracy dan 7% error rate.

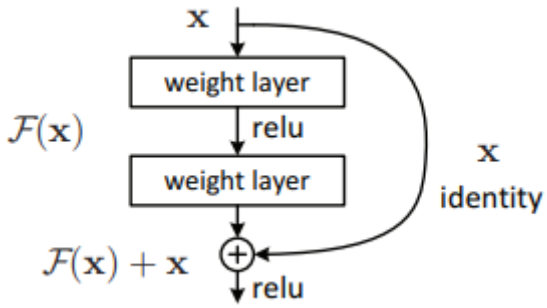
2.6.3 ResNet

Peneliti CNN terdahulu meyakini bahwa semakin dalam (semakin banyak) *convolutional layer*, maka model semakin bagus. Namun setelah *convolutional layer* mencapai jumlah tertentu, performa model akan menurun. Penurunan performa model dapat dilihat pada gambar 2.7.



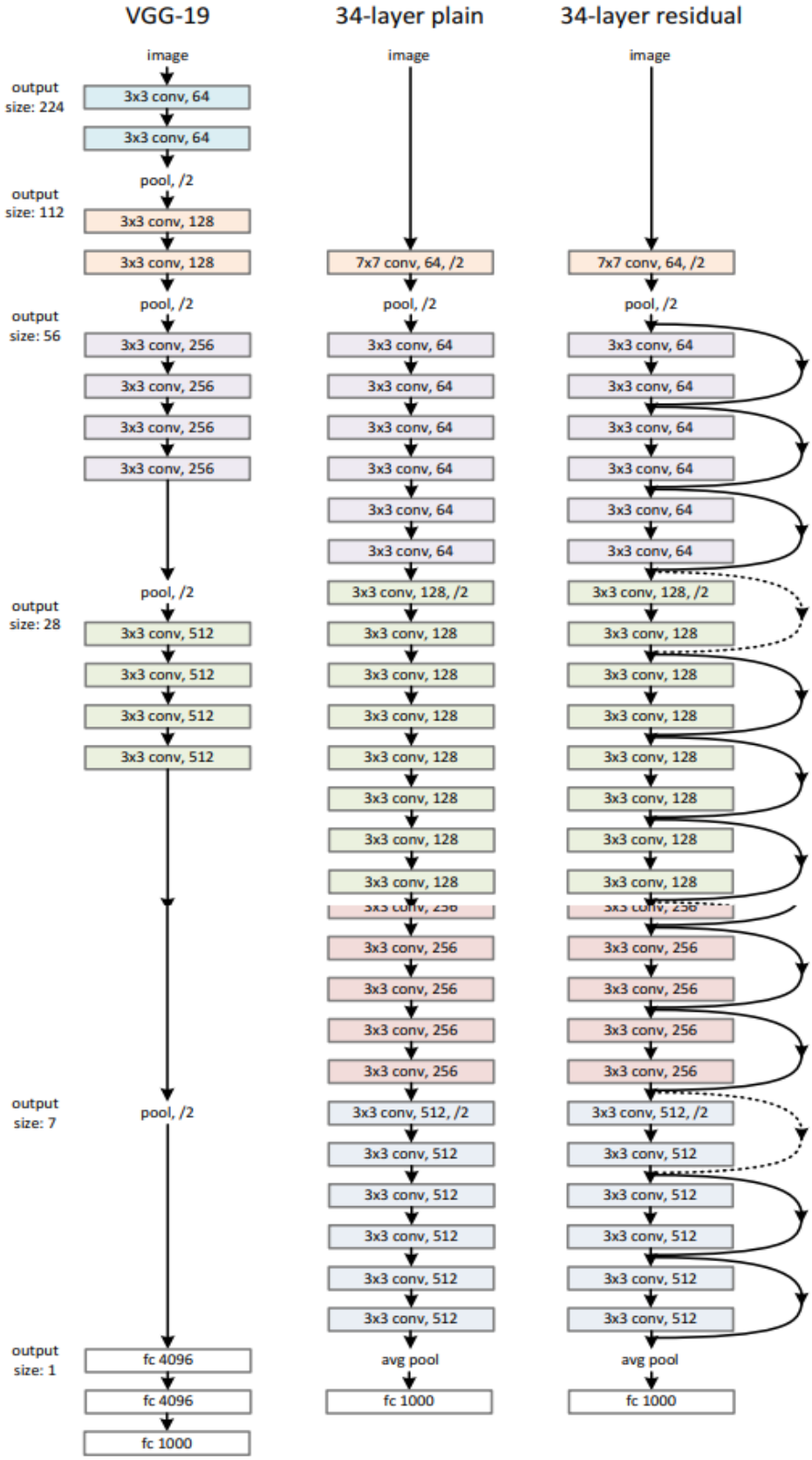
Gambar 2. 7 Penurunan performa pada test data

Hal ini merupakan batasan dari model VGG. Kemudian pada tahun 2015, ResidualNetwork (ResNet) diperkenalkan oleh KaimingHe melalui kompetisi ILSVRC 2015 dan menjadi *state of the art(sota)* cnn architecture pada tahun yang sama (He, Zhang, Ren, & Sun, 2016). Resnet menyelesaikan permasalahan *vanishing gradient (*nilai weights yang sudah tidak dapat di *update* karena model telah mengalami *bottleneck)* yang timbul karena network yang terlalu dalam. ResNet menggunakan residual blok untuk meningkatkan performa dan menyelesaikan *vanishing gradient.* Residual blok menerapkan konsep *skip connection* yaitu membuat jalan pintas alternatif yang dapat dilewati oleh gradien, hal ini memungkinan model untuk mempelajari *identity function* (Viso.AI, 2021). Konsep dari residual blok dapat dilihat pada gambar 2.8



Gambar 2. 8 Residual Blok

Perbandingan antara vgg, plain CNN, dan ResNet dapat dilihat pada gambar 2.9



Gambar 2. 9 Perbandingan arsitektur VGG-19, plain CNN, dan Resnet-34

2.7. Transfer Learning

Transfer learning merupakan teknik *machine learning* yang digunakan untuk meningkatkat akurasi dengan menggunakan model yang telah di training sebelumnya sebagai *starting point* untuk permasalahan yang berbeda namun masih relevan (Brownlee, A Gentle Introduction to Transfer Learning for Deep Learning, 2021). *Transfer learning* memungkinkan ml model untuk menyelesaikan masalah dengan sampel yang sedikit (ZHUANG, et al., 2020). Model yang telah di training sebelumnya dan kemudian digunakan kembali sebagai *starting point* untuk membuat model baru disebut *pre-trained model* (Han, et al., 2021). *Pre-trained* yang digunakan pada penelitian ini adalah ResNet50v2, EfficientNetB0, dan Inceptionv3.

2.7.1 Resnet50V2

ResNet50V2 merupakan varian dari residual neural network atau ResNet (He, Zhang, Ren, & Sun, 2016) dan merupakan. versi upgrade dari ResNet50. ResNet50V2 memiliki akurasi yang lebih tinggi daripada ResNet50. ResNet dibuat oleh team dari Microsoft untuk mengatasi permasalahan *vanishing gradient* saat proses training berlangsung. ResNet mengenalkan konsep yang disebut *ResidualBlock*. S*ummary* dari model yang meggunakan ResNet50V2 untuk transfer learning dapat dilihat pada gambar 2.4.

```
Model: "ResNet50V2_Softmax_aumentation"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 feature_extraction_layer (K  (None, 2048)             23564800
 erasLayer)

 dense (Dense)               (None, 2)                 4098

=================================================================
Total params: 23,568,898
Trainable params: 23,523,458
Non-trainable params: 45,440
_________________________________________________________________
```

Gambar 2. 10 Summary model transfer learning ResNet50v2

Berdasarkan summary terdapat dua layer yaitu *feature extraction layer* dan *dense layer* dengan dua unit neuron sebagai output layer. *Feature extraction layer* merupakan *pre-trained model* yang di training menggunakan citra image-net yang memiliki 1000 kelas yang berbeda. Model diatas memiliki 23.568.898 total parameter.

2.7.2 InceptionV3

InceptionV3 (Szegedy, Vanhoucke, Ioffe, Shlens, & Wojna, 2016) merupakan varian dari model Inception. Inception pertama kali diperkenalkan pada saat ImageNet Recognition Challenge (ILSVRC 2014). Inception V3 juga pernah menjadi *state of the art* ml *architecture* menjelang tahun 2016 (Paperswithcode, 2015). Nama Inception sendiri terinspirasi dari film "Inception" pada tahun 2010 yang disutradarai oleh Christopher Nolan (Szegedy, et al., 2015). Gamar 2.5 adalah summary dari model transfer learning yang menggunakan inveptionv3 sebagai *feature extraction layer.*

```
Model: "inceptionv3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 feature_extraction_layer (K  (None, 2048)             21802784
 erasLayer)

 dense_2 (Dense)             (None, 2)                 4098

=================================================================
Total params: 21,806,882
Trainable params: 21,772,450
Non-trainable params: 34,432
_________________________________________________________________
```

Gambar 2. 11 Summary model inception

Berdasarkan summary terdapat dua layer yaitu *feature extraction layer* dan *dense layer* dengan dua unit neuron sebagai output layer. Model inceptionv3 yang digunakan adalah model dengan benchmark ImageNet. Model diatas memiliki 21.806.882 total parameter, lebih sedikit jika dibandingkan dengan model yang menggunakan ResNet50v2.

2.7.3 EfficientNetB0

EfficientNet dibuat oleh dua insinyur dari Google Brain Team pada tahun 2019 yang Bernama Mingxin Tan dan Quoc V. Le . (Tan & Le, 2019). EfficientNet berhasil mencapai *state of the art* akurasi pada dataset ImageNet dengan akurasi 84.4%. EfficientNet juga memiliki varian EffNet-L2 yang saat ini memiliki *state of the art* akurasi tertinggi pada dataset CIFAR-100 dengan akurasi 96.08% (paperswithcode, 2020). Pada gambar 2.6 adalah summary dari model yang menggunakan EfficientNetB0 sebagai *feature extraction layer.*

```
Model: "EfficientNetB0"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 feature_extraction_layer (K  (None, 1280)             4049564
 erasLayer)

 dense_1 (Dense)             (None, 2)                 2562

=================================================================
Total params: 4,052,126
Trainable params: 4,010,110
Non-trainable params: 42,016
_________________________________________________________________
```

Gambar 2. 12 summary model efficientnetb0

Berdasarkan summary terdapat dua layer yaitu *feature extraction layer* dan *dense layer* dengan dua unit neuron sebagai output layer. Model efficientnetb0 yang digunakan adalah model dengan benchmark ImageNet. Model diatas memiliki 4.052.126 total parameter, model efficientnetb0 memiliki parameter yang paling sedikit jika dibandingkan dari dua model sebelumnya.

2.8 Google Colaboratory

Gambar 2. 13 Logo google colaboratory

Google Colaboratory disinggkat "Colab" merupakan produk dari google research. Colab memungkinkan pengguna untuk mengeksekusi kode python melalui web browser. Colab sangat cocok untuk *machine learning*, *data analysis*, dan edukasi. Colab menyediakan layanan Jupyter Notebook sehingga tidak perlu melakukan setup dan bisa langsung digunakan. Colab juga menyediakan akses gratis GPU namun dengan batasan penggunaan.

2.9 Python

Gambar 2. 14 Logo python

Python merupakan bahasa pemrograman tingkat tinggi yang dapat digunakan untuk banyak kasus(*general purpose*). Python diciptakan oleh Guido van Rosum dan pertama kali di rilis pada 20 februari 1991. Nama python terinspirasi dari acara favorit Guido yang tayang di BBC dengan nama Monty Python's Flying Circus (PythonInstitute, 2021). Pada tahun 2021 berdasarkan stack overflow survey, python digunakan oleh 48.24% developer di seluruh dunia dari 83,052 responden (StackOverflowInsight, 2021).

2.10 Library Python

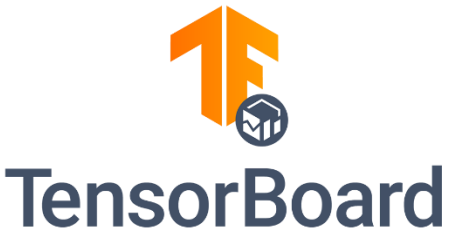
Pada penelitian ini akan digunakan library python seperti tensorflow dan ekosistemnya, numpy pandas, dan matplotlib

Gambar 2. 15 logo tensorflow

Gambar 2.9 merupakan logo tensorflow yang terbaru. Tensorflow merupakan open-source library yang digunakan untuk *machine learning* dan *artificial intelligence.* Tensorflow dibuat oleh Google Brain Team untuk tujuan riset dan produksi yang kemudian dirilis ke publik pada tahun 2015 dengan tensorflow versi 1.0. Tensorflow telah digunakan oleh berbagai perusahan besar seperti AirBnB, CocaCola, AMD, Intel, untuk membuat produk *machine learning* maupun untuk keperluan riset. Saat ini tensorflow memiliki versi yang terbaru yaitu Tensorflow 2.7 .

Gambar 2. 16 Logo tensorfow hub

Tensorflow Hub disingkat “TF Hub” merupakan yang digunakan sebagai repository dan library untuk *machine learning* siap guna, Repository TF Hub menyediakan *pre-trained* model digunakan untuk domain kasus (*problem domain*) teks, audio, citra, hingga video. TF Hub memungkinkan untuk mengimplementasikan *state of the art (sota)* model hanya dengan beberapa baris kode saja. Untuk logonya dapat dilihat pada gambar 2.10



Gambar 2. 17 Logo tensorboard

Tensorboard merupakan platform terintegrasi dengan Tensorflow yang digunakan untuk visualisasi performa dari model. Tensorboard juga dapat digunakan untuk menyimpan *record* dari percobaan saat membuat model. Untuk logonya dapat dilihat pada gambar 2.11

Gambar 2. 18 logo numpy

Numpy(*numerical python)* merupakan library open-source python library yang digunakan untuk perhitungan saintifik. Numpy mengandung multi dimensi array dan struktur data matriks. Numpy sangat popular digunakan untuk keperluan analisis data. Pada gambar 2.12 merupakan logo dari numpy

Gambar 2. 19 logo pandas

Pandas merupakan python library yang digunakan untu manipulasi dan analisis data. Pandas dibuat berdasarkan library numpy. Secara umum pandas menganduk struktur data yang disebut DataFrame yang berguna untuk memanipulasi tabel numerik,Pandas sangat popular di kalangan data saintis dan data analis karena pandas dapat digunakan untuk data cleaning, visualisasi data, analisis statistik, normalisasi data, operasi merges and join, dan masih banyak lagi (pandas - Python Data Analisys Library, 2021). Pada gambar 2.13 merupakan logo dari pandas

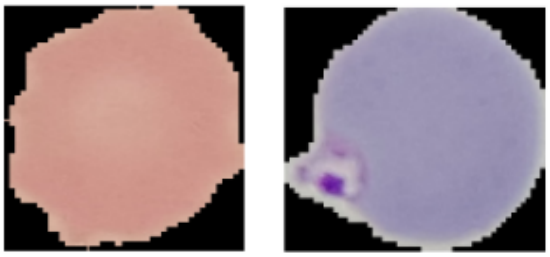
Gambar 2. 20 logo matplotlib

Matplotlib merupakan library python yang digunakan untuk visualisasi data yang interaktif. Matplotlib merupakan ekstensi dari numpy dan biasanya digunakan bersamaan dengan pandas untuk memvisualisasikan model machine learning maupun analisa data. Matplotlib dapat menyajikan *insight* dari data yang telah dianalisa. Pada gambar 2.14 merupakan logo dari matplotlib

Bab III
Perancangan Sistem

3.1 Dataset Malaria

Dataset yang digunakan adalah dataset citra mikroskop sel darah merah yang terdiri dari dua jenis gambar, yaitu sel yang sehat dan sel yang terinfeksi malaria. Total gambar ada sebanyak 13800 untuk gambar sel yang sehat dan 13800 untuk gambar sel terinfeksi malaria. Dataset didapat dari situs Kaggle dengan link (https://www.kaggle.com/iarunava/cell-images-for-detecting-malaria). Sampel dari citra dapat dilihat pada gambar 3.1.



Gambar 3 1 sampel dari citra sel yang parasitized(kanan) dan uninfected(kiri)

Citra sebelah kiri merupakan citra dari sel yang tidak terinfeksi(*uninfected)* oleh parasite plasmodium sedangkan citra sebelah kanan merupakan citra dari sel yang terinfeksi oleh plasmodium (*parasitized)*. Sel yang terinfeksi berubah warnanya yang semula berwarna merah muda menjadi biru pucat disertai bitnik hitam.

3.2 Eksperimen

Tabel 3. 1 Eksperimen

Eksperimen	Epoch	AugmentasiData	LearningRate	CitraInput	Optimizer	FineTune
1	10	Ya	0.001	64x64	Adam	Tidak
2	20	Ya	0.0001	96x96	Adam	Tidak
3	20	Ya	0.001	96x96	Adam	Ya
4	30	Tidak	0.0001	96x96	Adam	Ya
5	30	Ya	0.0001	96x96	Adam	Ya

Pada penelitian ini akan dilakukan empat eksperimen untuk masing masing model transfer learning.

Eksperimen 1:

Pada eksperimen pertama untuk menguji bagaimana performa model tanpa me re-train *feature extraction layer(pre-trained model)*

Eksperimen 2:

Pada eksperimen ke-dua dilkakukan untuk menguji apakah performa model akan lebih baik dari eksperimen 1 dengan menambah jumlah epoch, mengurangi learning rate, dan menambah ukuran citra input.

Eksperimen 3:

Pada eksperimen ke-tiga dilakukan fine tuning, yaitu me-retrain ulang semua layer termasuk layer dari *pretrained model* yang digunakan sebagai *feature extraction layer.* Hal ini dilakukan karena pre-trained model ditraining pada dataset yang tidak memiliki kelas sel terinfeksi (*parasitized*) maupun sel yang sehat (*uninfected).*

Eksperimen 4:

Pada eksperimen ke-empat dilakukan untuk menguji apakah performa model akan lebih baik dari eksperimen ke-tiga dengan menambah jumlah epoch dan mengurangi learning rate.

3.3 Pengujian dan Evaluasi

Pengujian yang dilakukan menggunakan *accuracy* dan *loss*. *Train accracy* dan *validation accuracy* dan *train loss* dan *validation loss.* Untuk menguji performa model, akan dilakukan prediksi pada *test set* sehingga menghasilkan *confusion matrix* dan *f1-score.*

Tabel 3. 2 Confusion Matrix

Kondisi Sesungguhnya	Hasil Prediksi	
	Parasitized cell	*Uninfected*
Parasitized cell	*True Positive*	*False Negative*
Uninfected	*False Positive*	*True Negative*

Hasil prediksi di sini adalah hasil keluaran/output dari model pada data *test set* yang akan dibandingkan dengan kondisi sesungguhnya. Semakin tinggi nilai *accuracy* pada hasil prediksi, maka performa model semakin baik.Sedangkan untuk *loss,* semakin rendah nilai *loss* model semakin baik. Berikut adalah rumus untuk mendapatkan *accuracy* dan *loss*.

$$Accuracy = \frac{(TP+TN)}{(TP+FP+FN+TN)} \quad (3.1)$$

$$Loss = \frac{(FP+FN)}{(TP+FP+FN+TN)} \quad (3.2)$$

Keterangan:

TP = *True Positive* (hasil prediksi benar bahwa hasil berupa positif)

FP = *False Positive* (hasil prediksi salah / negatif, hasil yang benar berupa positif)

TN = *True Negative* (hasil prediksi benar bahwa hasil berupa negatif)

FP = *False Negative* (hasil prediksi salah / positif, hasil yang benar berupa negatif)

F1-Score merupakan salah satu metrics yang digunakan untuk mengevaluasi model machine learning terutama model klasifikasi. F1-score terdiri dari *accuracy, precision,recall,* dan *f1-score. Accuracy* merukapan presentase prediksi benar dari total prediksi.

$$\text{Accuracy} = \frac{\text{correct predictions}}{\text{total predictions}} \quad (3.3)$$

Precision merupakan bagian pertama dari f1-score dan dapat digunakan sebagai *metrics* tersendiri. *Precision* didapat dari perbandingan antara jumlah prediksi *true positives* dengan total prediksi positif

$$Precision = \frac{\#\text{ of True Positives}}{\#\text{ of True Positives} + \#\text{ of False Positives}} \quad (3.4)$$

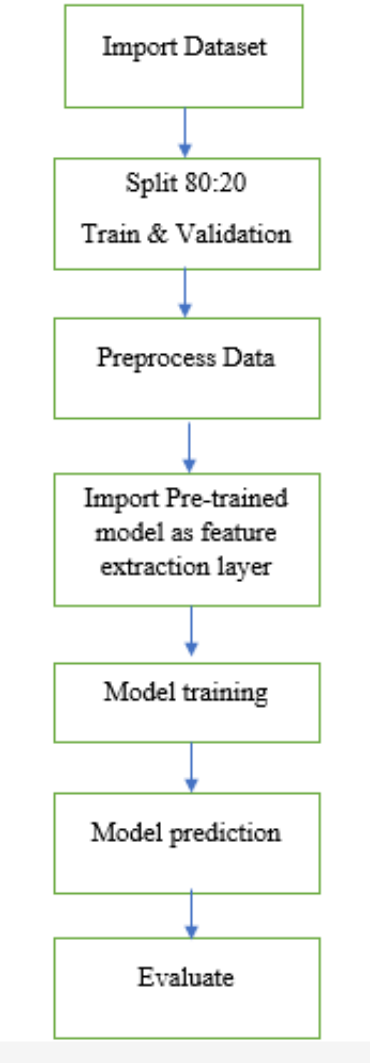
Recall adalah presentasi dari jumlah *true positif* dari total sampel yang digunakan unuk prediksi *true positive*

$$\text{Recall} = \frac{\#\text{ of True Positives}}{\#\text{ of True Positives} + \#\text{ of False Negatives}} \quad (3.5)$$

F1-score merupakan *metrics* yang didapat dengan cara menggabungkan *precision* dan *recall.* F1-score diperoleh dengan menhitung *harmonic mean* dari *precision* dan *recall*

$$\text{F1 score} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \quad (3.6)$$

3.4 Design sistem



Gambar 3. 2 Desain Sistem

Sistem dimulai dengan mengimpor data (mengunduh dataset yang diperlukan) dari polatform Kaggle. Proses *import* dilakukan dengan cara mengunggah file kaggle.json pada runtime colab. File Kaggle.json dapat diunduh pada profil akun Kaggle masing-masing.
Dataset yang telah diimpor akan dibagi menjadi dua bagian yaitu 80% untuk *train* dan 20% untuk *validation*. Data *train* akan digunakan saat proses training dan data validation digunakan pada saat training dan *evaluate*. Untuk *preprocess*, data dinormalisasi dan dilakukan penyesuaian input serta *load* data. Kemudian impor *pre-trained* model untuk transfer learning dari tfhub.dev. Kemudian dilakukan model *training* dan prediksi. Terakhir adalah mengevaluasi performa model. Evaluasi dilakukan dengan membandingkan hasil training dan f1-score.

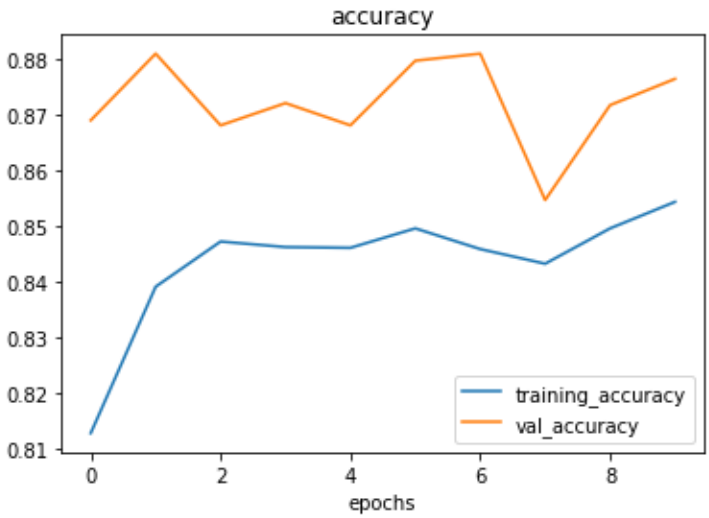
Bab IV
Hasil dan Pembahasan

4.1 Eksperimen 1

Eksperimen 1 dilakukan dengan menggunakan parameter ukuran citra 64 x64, 10 epoch, augmentasi data, tanpa fine tune, dan optimizer adam dengan learning rate default (0.001).

4.1.1 Resnet50V2

Hasil training:



Gambar 4. 1 Grafik akurasi resnet50v2 eksperimen 1

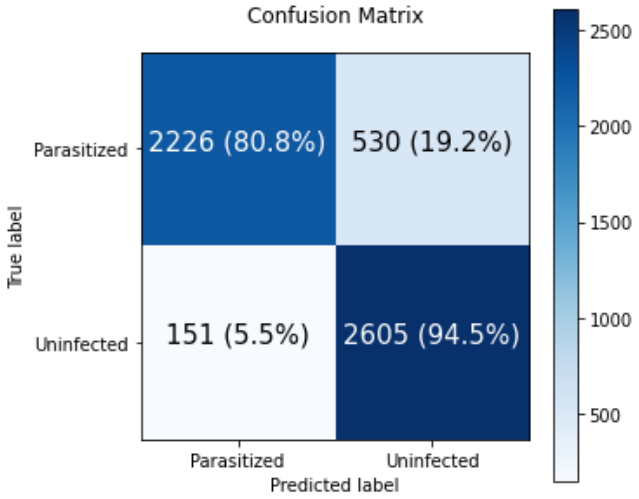
Gambar 4. 2 Grafik loss resnet50v2 eksperimen 1

Pada gambar 4.1 dapat dilihat bahwa training *training accuracy* meningkat n dari 0.81 pada saat training dimulai menjadi 0.85 pada epoch ke-10, namun tidak terjadi peningkatan pada *validation accuracy*. Hal yang sama dapat dilihat pada gambar 4.2, grafik *training loss* menunjukan penurunan dari 0.45 pada saat training di mulai menjadi 0.35 pada epoch ke-10 Hasil akhir dari training dapat dilihat pada tabel 4.1

Tabel 4. 1 Hasil akhir training resnet50v2 eksperimen 1

Train accuracy	*Train loss*	*Validation accuracy*	*Validation loss*
0.8543	0.3571	0.8765	0.3029

Kemudian, dilakukan prediksi pada model dan dari hasil prediksi dibuatlah *confusion matrix* yang dapat dilihat pada gambar 4.3



Gambar 4. 3 Confusion matrix resnet50v2 ekperimen 1

Pada gambar 4.3 dapat dilihat hasil prediksi dari model ResNet50v2, prediksi dilakukan menggunakan 5512 citra dengan perbandingan 50:50. Untuk citra *parasitized* berhasil terprediksi sejumlah 2226 (80.8%) dari 2756 citra dan *uninfected* berhasil terprediksi sejumlah 2605 (94.5%) dari 2756 citra.

Dari hasil prediksi juga dibuat F1-score yang dapat dilihat pada tabel 4.2

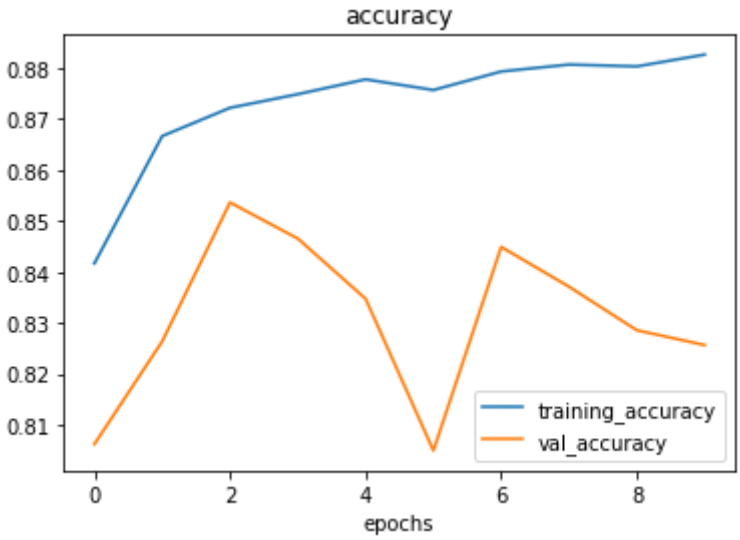
Tabel 4.2 F1-Score model ResNet50v2 eksperimen 1

	Precision	recall	Class F1-score	support
0	0.94	0.81	0.87	2756
1	0.83	0.95	0.88	2756
Accuracy			0.88	5512
Model F1-Score			0.875	5512

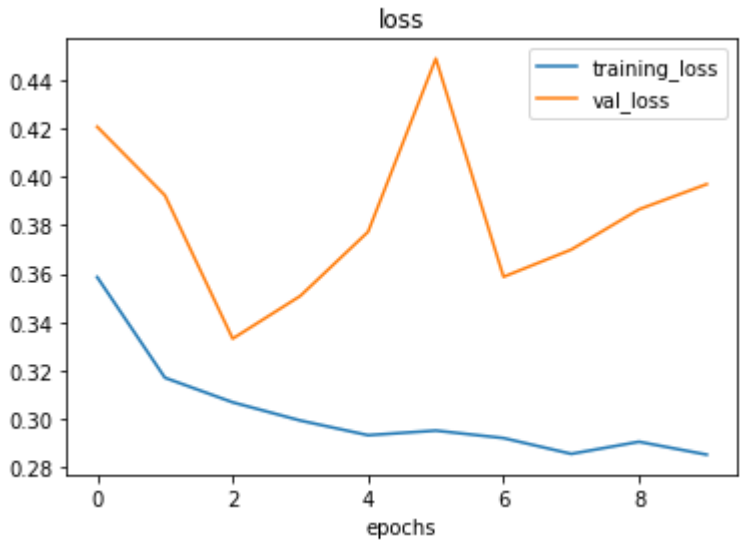
Dari tabel f1-score dapat diketahui nilai akurasi prediksi sebesar 0.88, nilai f1-score pada *class 0(parasitized)* adalah 0.87, dan nilai f1-score pada *class 1(uninfected)* adalah 0.8

4.1.2 EfficientNetB0

Hasil training:



Gambar 4. 4 Grafik akurasi EfficientNetB0 ekperimen 1



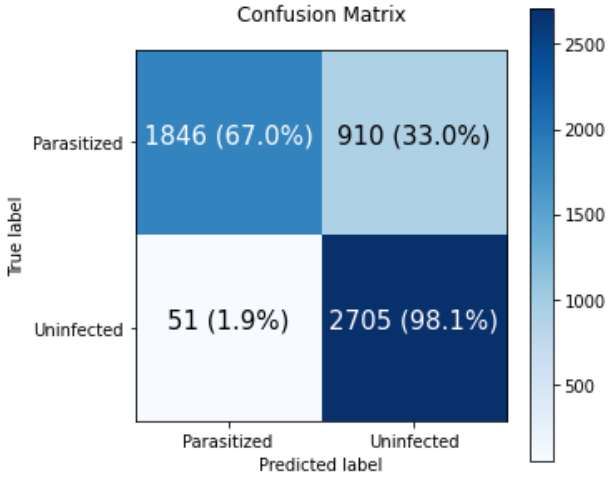
Gambar 4. 5 Grafik loss EfficientNetB0 eksperimen 1

Pada gambar 4.4 dapat diketahui bahwa *training accuracy* meningkat n dari 0.84 pada awal training menjadi 0.88 pada epoch ke-10, pada *validation accuracy* juga terjadi peningkatan dari 0.81 menjadi 0.83 pada akhir training. Hal yang sama dapat dilihat pada gambar 4.5, grafik *training loss* menunjukan penurunan dari 0.36 pada awal training ,menjadi 0.28 pada epoch ke-10 dan *validation loss* yang sebelumnya bernilai 0.42 turun menjadi 0.41.Hasil akhir dari training dapat dilihat pada tabel 4.3

Tabel 4. 3 Hasil akhir training EfficientNetB0 eksperimen 1

Train accuracy	*Train loss*	*Validation accuracy*	*Validation loss*
0.8841	0.2814	0.8173	0.4106

Kemudian, dilakukan prediksi pada model, dari hasil prediksi dibuatlah *confusion matrix* yang dapat dilihat pada gambar 4.6



Gambar 4. 6 Confusion matrix EfficientNetB0 eksperimen 1

Pada gambar 4.6 dapat dilihat hasil prediksi dari model EfficientNetB0, prediksi dilakukan menggunakan 5512 citra dengan perbandingan 50:50. Untuk citra *parasitized* berhasil terprediksi sejumlah 1846 (67.0%) dari 2756 citra dan *uninfected* berhasil terprediksi sejumlah 2705 (98.1%) dari 2756 citra. Dari hasil prediksi juga dibuat F1-score yang dapat dilihat pada tabel 4.2

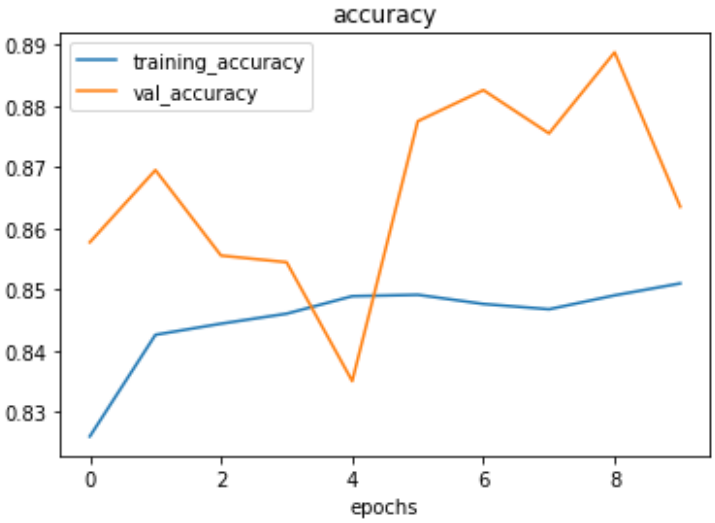
Tabel 4. 4 F1-Score EfficientNetB0 eksperimen 1

	Precision	recall	Class F1-score	support
0	0.97	0.67	0.79	2756
1	0.75	0.98	0.85	2756
Accuracy			0.83	5512
Model F1-Score			0.82	5512

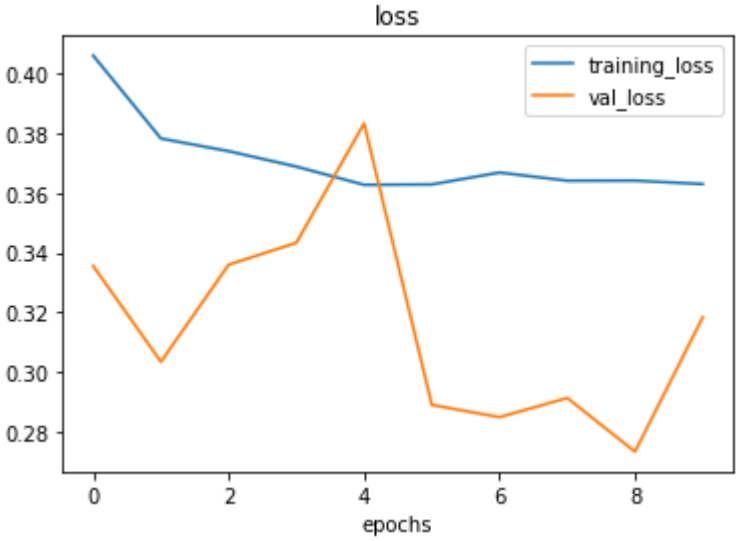
Dari tabel f1-score dapat diketahui nilai akurasi prediksi sebesar 0.83, nilai f1-score pada *class 0(parasitized)* adalah 0.79, dan nilai f1-score pada *class 1(uninfected)* adalah 0.85

4.1.3 InceptionV3

Hasil training:



Gambar 4. 7 Grafik akurasi InceptionV3 eksperimen 1



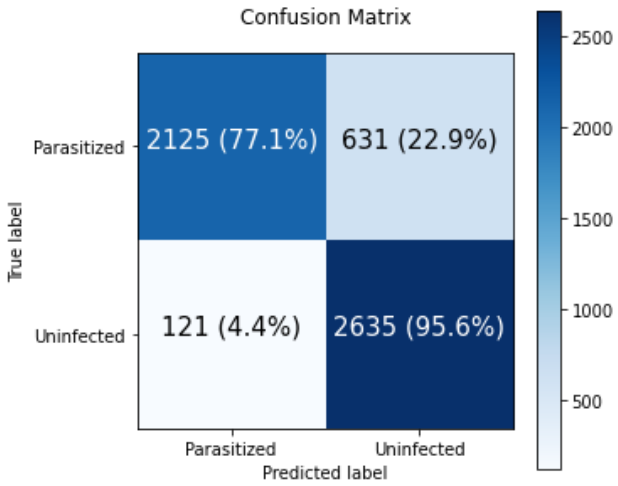
Gambar 4. 8 Grafik loss InceptionV3 eksperimen 1

Pada gambar 4.6 dapat diketahui bahwa *training accuracy* meningkat n dari 0.83 pada awal training menjadi 0.85 pada epoch ke-10,namun pada *validation accuracy* tidak terjadi peningkatan . Pada gambar 4.8, grafik *training loss* menunjukan penurunan dari 0.41 pada awal training ,menjadi 0.36 pada epoch ke-10 dan *validation loss* yang sebelumnya bernilai 0.34 turun menjadi 0.31.Hasil akhir dari training dapat dilihat pada tabel 4.5

Tabel 4. 5 Hasil akhir InceptionV3 eksperimen 1

Train accuracy	*Train loss*	*Validation accuracy*	*Validation loss*
0.8510	0.3630	0.8636	0.3183

Kemudian, dilakukan prediksi pada model dan dari hasil prediksi dibuatlah *confusion matrix* yang dapat dilihat pada gambar 4.9



Gambar 4. 9 Confusion matrix InceptionV3 eksperimen 1

Pada gambar 4.9 dapat dilihat hasil prediksi dari model InceptionV3, prediksi dilakukan menggunakan 5512 citra dengan perbandingan 50:50. Untuk citra *parasitized* berhasil terprediksi sejumlah 2125 (77.1%) dari 2756 citra dan *uninfected* berhasil terprediksi sejumlah 2635 (95.6%) dari 2756 citra. Dari hasil prediksi juga dibuat F1-score yang dapat dilihat pada tabel 4.6

Tabel 4. 6 F1-score inceptionv3 eksperimen 1

	Precision	recall	Class F1-score	support
0	0.95	0.77	0.85	2756
1	0.81	0.96	0.88	2756
Accuracy			0.86	5512
Model F1-Score			0.865	5512

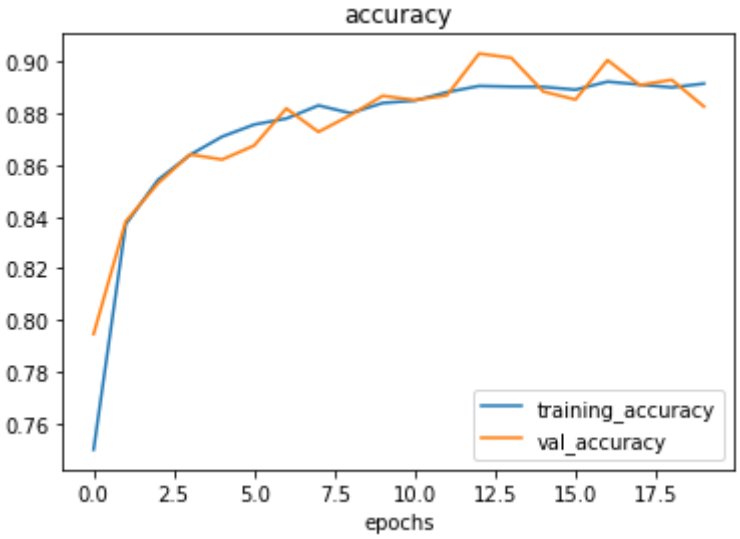
Dari tabel f1-score dapat diketahui nilai akurasi prediksi sebesar 0.83, nilai f1-score pada *class 0(parasitized)* adalah 0.79, dan nilai f1-score pada *class 1(uninfected)* adalah 0

4.2 Eksperimen 2

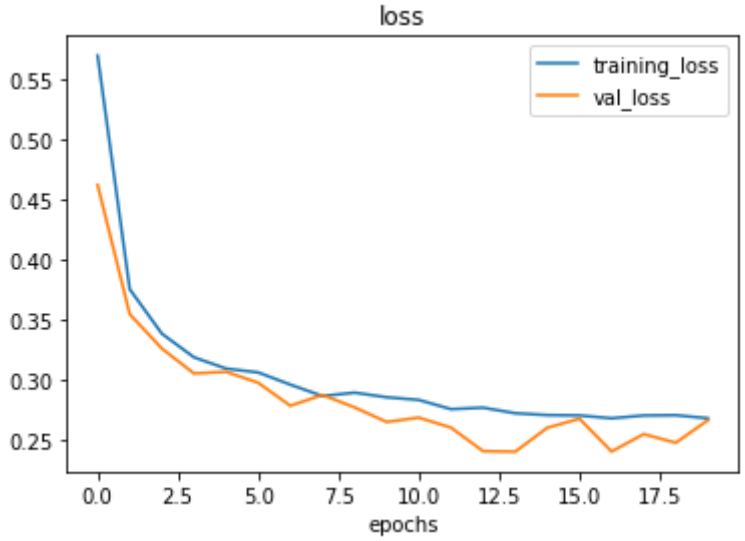
Eksperimen 2 dilakukan dengan menggunakan parameter ukuran citra 96x96, 20 epoch, tanpa fine tune, optimizer Adam dengan learning rate 1e-4. Berikut adalah hasilnya.

4.2.1 ResNet50v2

Hasil training:



Gambar 4. 10 Grafik akurasi ResNet50v2 eksperimen 2



Gambar 4. 11 Grafik loss ResNet50v2 eksperimen 2

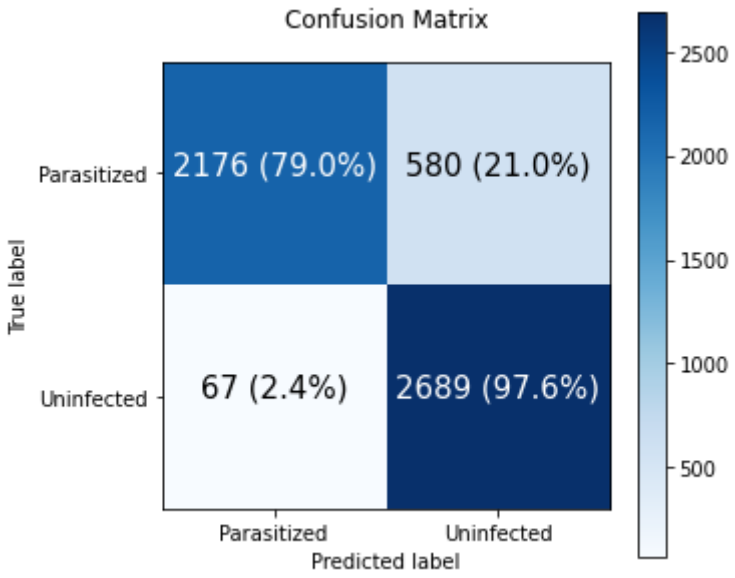
Pada gambar 4.10 dapat diketahui bahwa *training accuracy* meningkat n dari 0.75 pada saat training dimulai menjadi 0.89 pada epoch ke-10, seiringan dengan *validation accuracy* yang juga mengalami peningkatan n yang semula bernilai 0.79 menjadi 0.88 pada epoch ke-10.Hal yang sama dapat dilihat pada gambar 4.11, grafik *training loss* menunjukan penurunan

, yang semula bernilai 0.54 menjadi 0.27 pada epoch ke-10 Hasil akhir dari training dapat dilihat pada tabel 4.7

Tabel 4. 7 Hasil akhir training ResNet50v2 eksperimen 2

Train accuracy	*Train loss*	*Validation accuracy*	*Validation loss*
0.8915	0.2677	0.8826	0.2661

Kemudian, dilakukan prediksi pada model, dari hasil prediksi dibuat *confusion matrix* yang dapat dilihat pada gambar 4.12



Gambar 4. 12 Confusion matrix ResNett50v2 eskperimen 2

Pada gambar 4.12 dapat dilihat hasil prediksi dari model ResNet50v2, prediksi dilakukan menggunakan 5512 citra dengan perbandingan 50:50. Untuk citra *parasitized* berhasil terprediksi sejumlah 2176 (79.0%) dari 2756 citra dan *uninfected* berhasil terprediksi sejumlah 2689 (97.6%) dari 2756 citra. Dari hasil prediksi juga dibuat F1-score yang dapat dilihat pada tabel 4.8

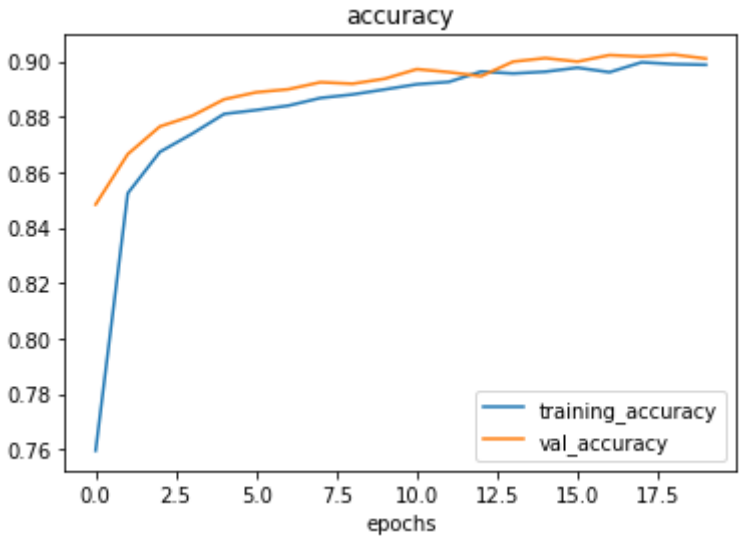
Tabel 4. 8 F1-Score ResNet50v2 eksperimen 2

	Precision	recall	Class F1-score	support
0	0.97	0.79	0.87	2756
1	0.82	0.98	0.89	2756
Accuracy			0.88	5512
Model F1-Score			0.88	5512

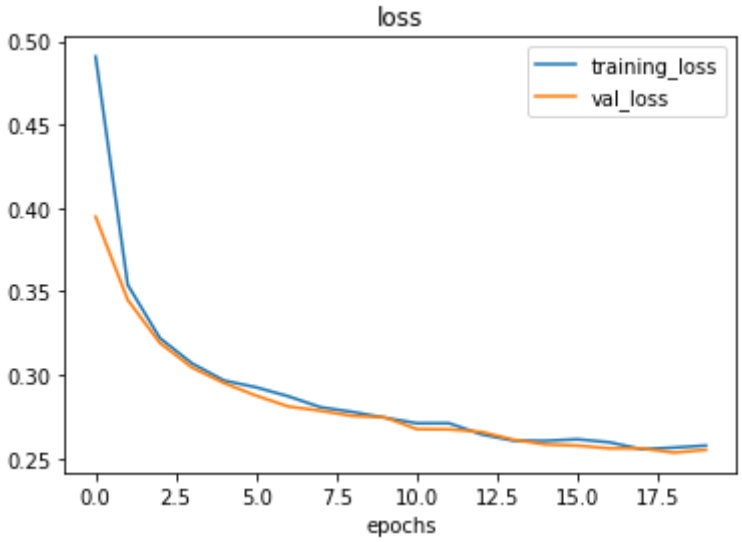
Dari tabel f1-score dapat diketahui nilai akurasi prediksi sebesar 0.88, nilai f1-score pada *class 0(parasitized)* adalah 0.87, dan nilai f1-score pada *class 1(uninfected)* adalah 0.89 dan model f1-score adalah 0.88

4.2.2 EfficientNetB0

Hasil training:



Gambar 4. 13 Grafik akurasi EfficientnetB0 eksperimen 2



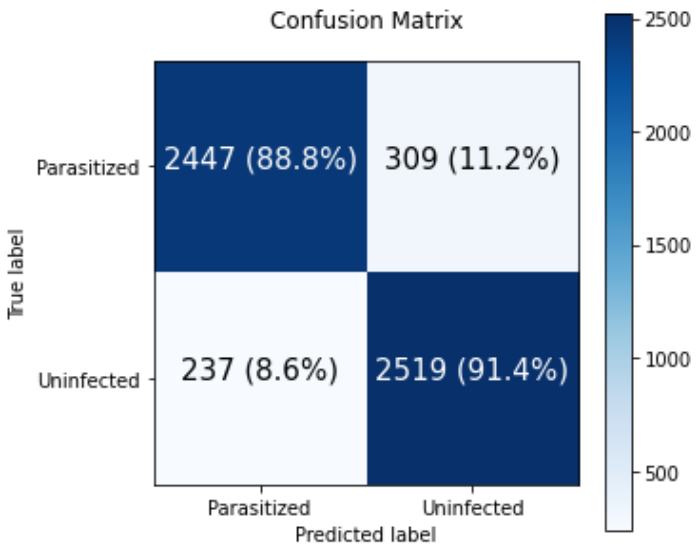
Gambar 4. 14 Grafik loss EfficientNetB0 eksperimen 2

Pada gambar 4.13 dapat diketahui bahwa *training accuracy* meningkat n semula 0.76 pada awal training menjadi 0.89 pada epoch ke-10, seiringan dengan *validation accuracy* yang juga mengalami peningkatan n yang semula bernilai 0.75 menjadi 0.90 pada epoch ke-10. Hal yang sama dapat dilihat pada gambar 4.14, grafik *training loss* menunjukan penurunan , yang semula bernilai 0.50 menjadi 0.25, dan *validation loss* yang semula bernilai 0.40 menjadi 0.25 pada epoch ke-10 Hasil akhir dari training dapat dilihat pada tabel 4.7

Tabel 4. 9 Hasil akhir training EfficientNetB0 eksperimen 2

Train accuracy	*Train loss*	*Validation accuracy*	*Validation loss*
0.8987	0.2577	0.9009	0.2552

Kemudian, dilakukan prediksi pada model, dari hasil prediksi dibuat *confusion matrix* yang dapat dilihat pada gambar 4.15



Gambar 4. 15 Confusion matrix EfficientNetB0 eksperimen 2

Pada gambar 4.15 dapat dilihat hasil prediksi dari model EfficientNetB0, prediksi dilakukan menggunakan 5512 citra dengan perbandingan 50:50. Untuk citra *parasitized* berhasil terprediksi sejumlah 2447 (88.8%) dari 2756 citra dan *uninfected* berhasil terprediksi sejumlah 2519 (91.4%) dari 2756 citra. Dari hasil prediksi juga dibuat F1-score yang dapat dilihat pada tabel 4.8

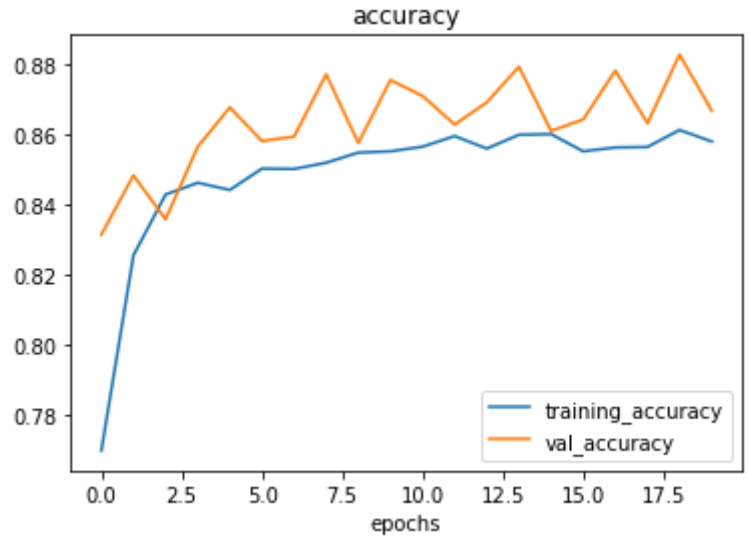
Tabel 4. 10 F1-score EfficientNetB0 eksperimen 2

	Precision	recall	Class F1-score	support
0	0.91	0.89	0.90	2756
1	0.89	0.91	0.90	2756
Accuracy			0.90	5512
Model F1-Score			0.90	5512

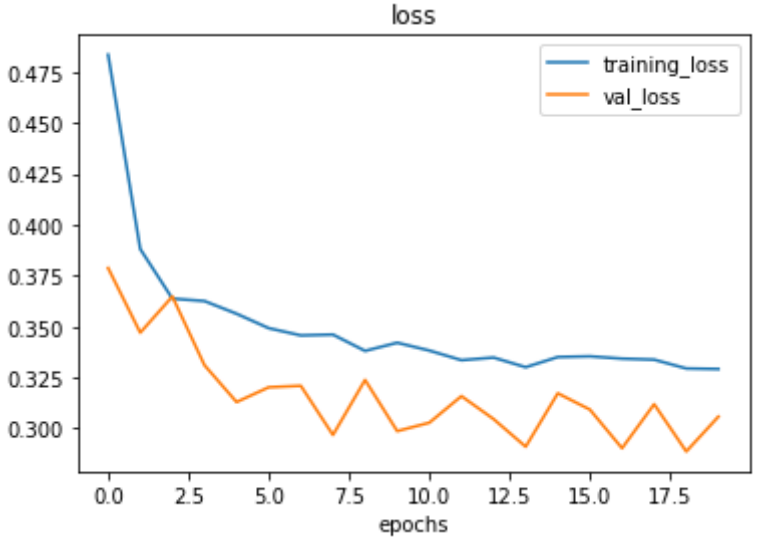
Dari tabel f1-score dapat diketahui nilai akurasi prediksi sebesar 0.90, nilai f1-score pada *class 0(parasitized)* adalah 0.90, dan nilai f1-score pada *class 1(uninfected)* adalah 0.

4.2.3 InceptionV3

Hasil training:



Gambar 4. 16 Grafik akurasi InceptionV3 eksperimen 2



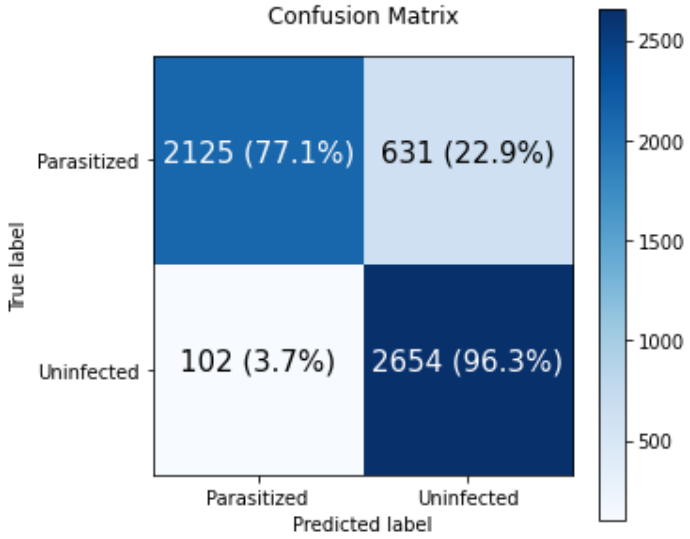
Gambar 4. 17 Grafik loss InceptionV3 eksperimen 2

Pada gambar 4.16 dapat diketahui bahwa *training accuracy* meningkat n semula 0.77 pada awal training menjadi 0.85 pada epoch ke-10, seiringan dengan *validation accuracy* yang juga mengalami peningkatan n yang semula bernilai 0.83 menjadi 0.86 pada epoch ke-10. Hal yang sama dapat dilihat pada gambar 4.17, grafik *training loss* menunjukan penurunan , yang semula bernilai 0.475 menjadi 0.32, dan *validation loss* yang semula bernilai 0.375 menjadi 0.3 pada epoch ke-10 Hasil akhir dari training dapat dilihat pada tabel 4.11

Tabel 4. 11 Hasil akhir training inceptionv3 eksperimen 2

Train accuracy	*Train loss*	*Validation accuracy*	*Validation loss*
0.8582	0.3291	0.8670	0.3058

Kemudian, dilakukan prediksi pada model, dari hasil prediksi dibuat *confusion matrix* yang dapat dilihat pada gambar 4.18



Gambar 4. 18 confusion matrix InceptionV3 eksperimen 2

Pada gambar 4.15 dapat dilihat hasil prediksi dari model EfficientNetB0, prediksi dilakukan menggunakan 5512 citra dengan perbandingan 50:50. Untuk citra *parasitized* berhasil terprediksi sejumlah 2447 (88.8%) dari 2756 citra dan *uninfected* berhasil terprediksi sejumlah 2519 (91.4%) dari 2756 citra. Dari hasil prediksi juga dibuat F1-score yang dapat dilihat pada tabel 4.8

Tabel 4. 12 F1-score Inceptionv3 eksperimen 2

	Precision	recall	Class F1-score	support
0	0.95	0.77	0.85	2756
1	0.81	0.96	0.88	2756
Accuracy			0.87	5512
Model F1-Score			0.865	5512

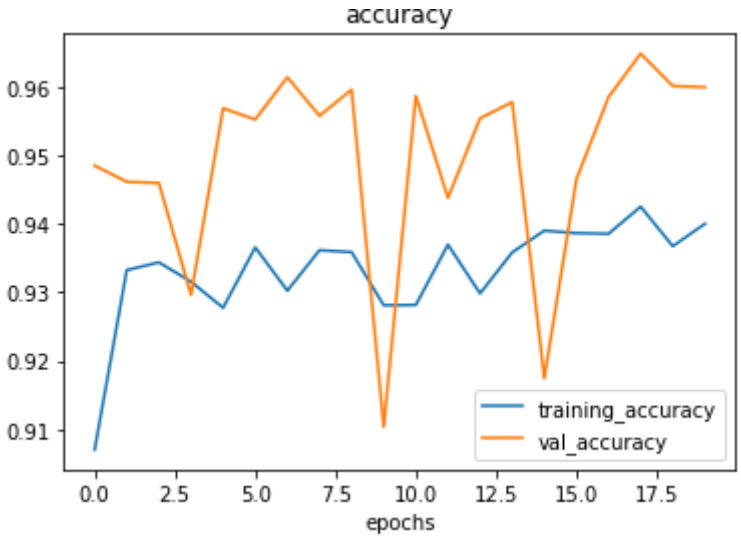
Dari tabel f1-score dapat diketahui nilai akurasi prediksi sebesar 0.87, nilai f1-score pada *class 0(parasitized)* adalah 0.85, dan nilai f1-score pada *class 1(uninfected)* adalah 0.88 , dan model f1-score adalah 0.865

4.3 Eksperimen 3

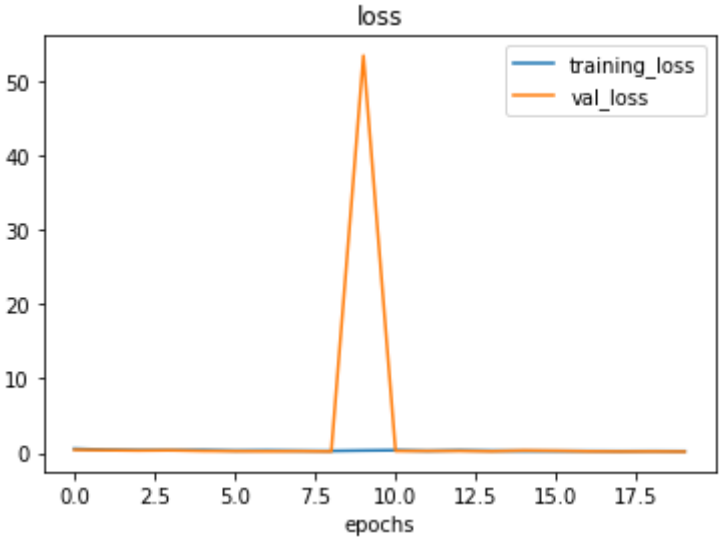
Eksperimen 3 dilakukan dengan menggunakan parameter ukuran citra 96x96, 20 epoch, dengan fine tune, optimizer Adam dengan learning rate default (1e-3). Berikut adalah hasilnya.

4.3.1 ResNet50v2

Hasil training:



Gambar 4. 19 Grafik akurasi Resnet50v2 eksperimen 3



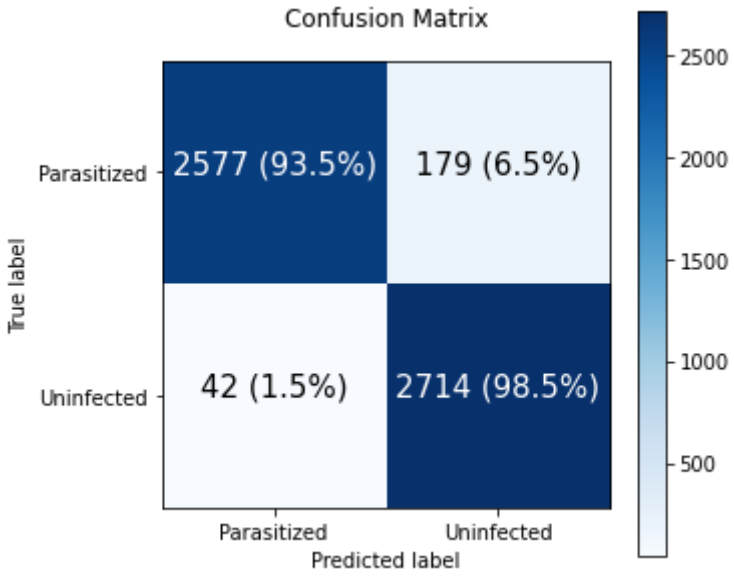
Gambar 4. 20 Grafik loss ResNet50v2 eksperimen 3

Pada gambar 4.19 dapat diketahui bahwa *training accuracy* meningkat dari 0.90 pada saat training dimulai menjadi 0.94 pada epoch ke-10, sedangkan *validation accuracy* yang semula bernilai 0.94 meningkat menjadi 0.95 pada epoch ke-20.Pada gambar 4.20, grafik *training loss* fluktuasi yang cukup besar pada epoch ke-9 kemungkinan diakibatkan oleh learning rate yang kurang sesuai, Hasil akhir dari training dapat dilihat pada tabel 4.13

Tabel 4. 13 Hasil akhir training ResNet50v2 eksperimen 3

Train accuracy	*Train loss*	*Validation accuracy*	*Validation loss*
0.9400	0.2300	0.9599	0.1689

Kemudian, dilakukan prediksi pada model, dari hasil prediksi dibuat *confusion matrix* yang dapat dilihat pada gambar 4.22



Gambar 4. 21 Confusion matrix ResNett50v2 eskperimen 3

Pada gambar 4.21 dapat dilihat hasil prediksi dari model ResNet50v2, prediksi dilakukan menggunakan 5512 citra dengan perbandingan 50:50. Untuk citra *parasitized* berhasil terprediksi sejumlah 2577 (93.5%) dari 2756 citra dan *uninfected* berhasil terprediksi sejumlah 2714 (98.5%) dari 2756 citra. Dari hasil prediksi juga dibuat F1-score yang dapat dilihat pada tabel 4.8

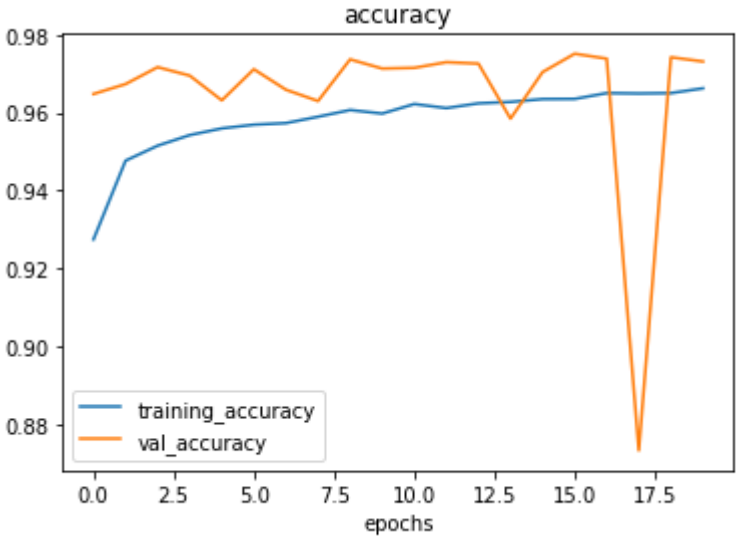
Tabel 4. 14 F1-Score ResNet50v2 eksperimen 3

	Precision	recall	Class F1-score	support
0	0.98	0.94	0.96	2756
1	0.94	0.98	0.96	2756
Accuracy			0.96	5512
Model F1-Score			0.96	5512

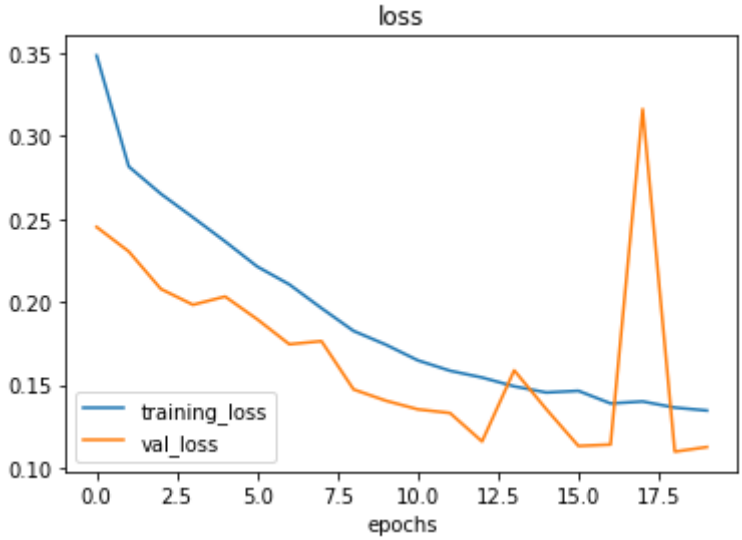
Dari tabel f1-score dapat diketahui nilai akurasi prediksi sebesar 0.96, nilai f1-score pada *class 0(parasitized)* adalah 0.96, dan nilai f1-score pada *class 1(uninfected)* adalah 0.96

4.3.2 EfficientNetB0

Hasil training:



Gambar 4. 21 Grafik akurasi EfficientNetB0 eksperimen 3



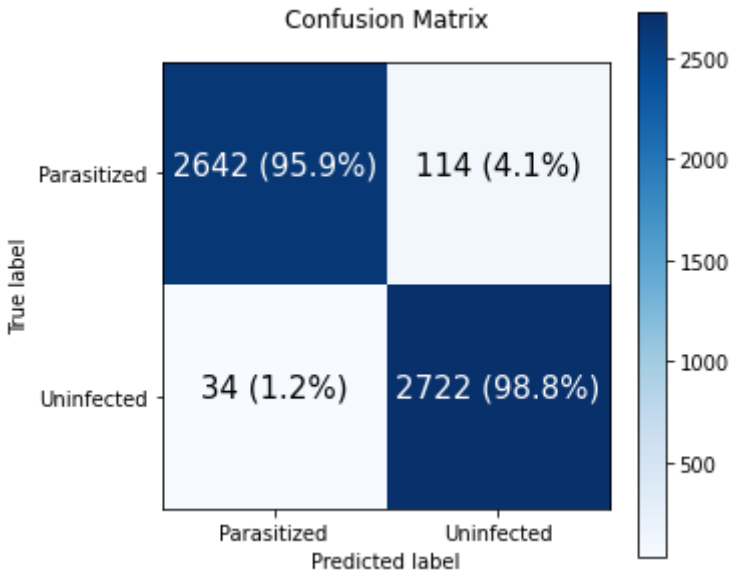
Gambar 4. 22 Grafik loss EfficientNetB0 eksperimen 3

Pada gambar 4.21 dapat diketahui bahwa *training accuracy* meningkat yang semula bernilai 0.92 pada awal training menjadi 0.96 pada epoch ke-20, sedangkan *validation accuracy* tidak mengalami peningkatan dan terjadi fluktuasi pada epoch ke-17. Pada gambar 4.22, grafik *training loss* menunjukan penurunan , yang semula bernilai 0.39 menjadi 0.13, dan *validation loss* yang semula bernilai 0.25 menjadi 0.11 pada epoch ke-20 Hasil akhir dari training dapat dilihat pada tabel 4.15

Tabel 4. 15 Hasil akhir training EfficientNetB0 eksperimen 3

Train accuracy	*Train loss*	*Validation accuracy*	*Validation loss*
0.9663	0.1345	0.9731	0.1125

Kemudian, dilakukan prediksi pada model, dari hasil prediksi dibuat *confusion matrix* yang dapat dilihat pada gambar 4.23



Gambar 4. 23 Confusion matrix EfficientNetB0 eksperimen 3

Pada gambar 4.23 dapat dilihat hasil prediksi dari model EfficientNetB0, prediksi dilakukan menggunakan 5512 citra dengan perbandingan 50:50. Untuk citra *parasitized* berhasil terprediksi sejumlah 2642 (95,9%) dari 2756 citra dan *uninfected* berhasil terprediksi sejumlah 2722(98.8%) dari 2756 citra. Dari hasil prediksi juga dibuat F1-score yang dapat dilihat pada tabel 4.16

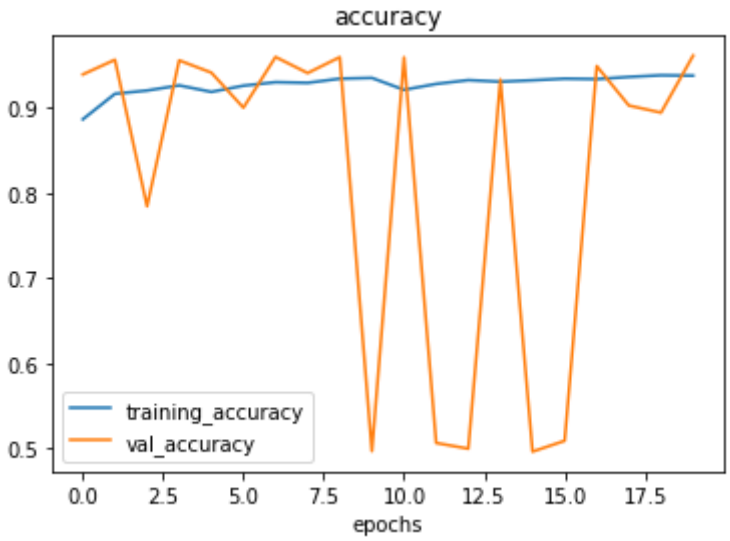
Tabel 4. 16 F1-score EfficientNetB0 eksperimen 3

	Precision	recall	Class F1-score	support
0	0.99	0.96	0.97	2756
1	0.96	0.99	0.97	2756
Accuracy			0.97.5	5512
Model F1-Score			0.97	5512

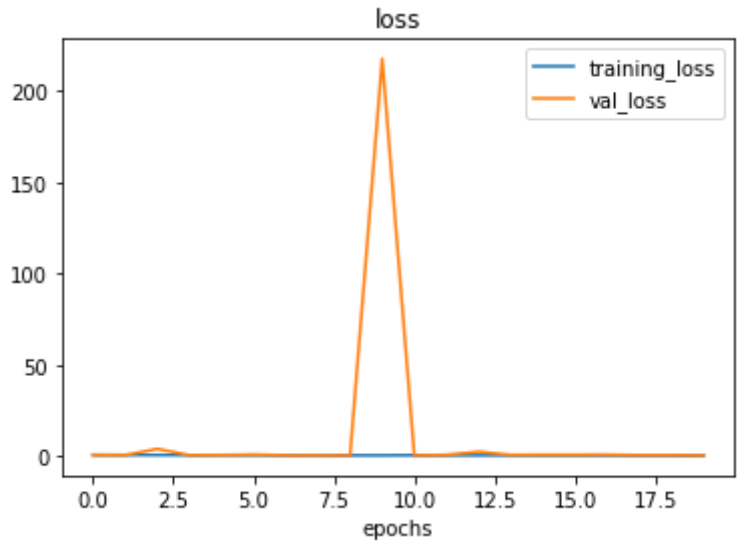
Dari tabel f1-score dapat diketahui nilai akurasi prediksi sebesar 0.97.5, nilai f1-score pada *class 0(parasitized)* adalah 0.97, dan nilai f1-score pada *class 1(uninfected)* adalah 0.97 dan model f1-score adalah 97.

4.3.3 InceptionV3

Hasil training:



Gambar 4. 24 Grafik akurasi InceptionV3 eksperimen 3



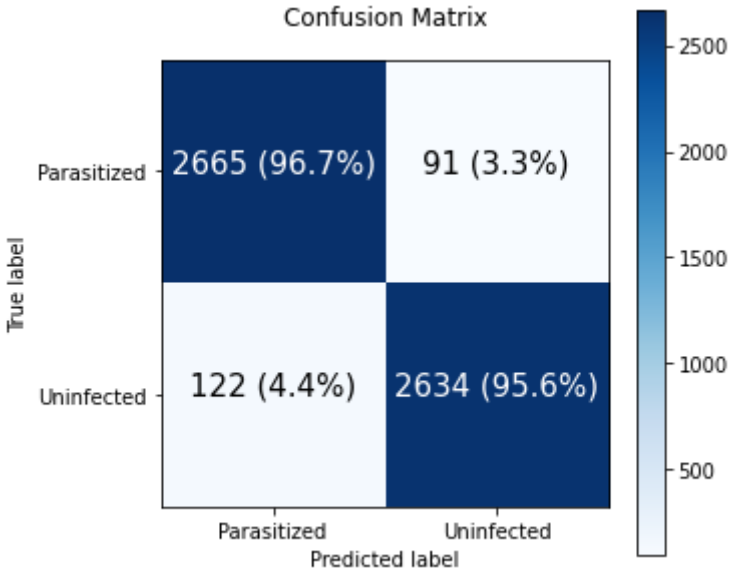
Gambar 4. 25 Grafik loss InceptionV3 eksperimen 3

Pada gambar 4.24 dapat diketahui bahwa *training accuracy* meningkat meskipun tidak signifikan dan tidak ada peningkatan signifikan dari *validation accuracy*.Pada *validation accuracy* fluktuasi yang tinggi pada epoch 9,11,12,14 dan 15. Pada gambar 4.25 terjadi fluktuasi *validation loss* yang sangat tinggi pada epoch ke-9 . Hasil akhir dari training dapat dilihat pada tabel 4.11

Tabel 4. 17 Hasil akhir training inceptionv3 eksperimen 3

Train accuracy	*Train loss*	*Validation accuracy*	*Validation loss*
0.9378	0.2283	0.9614	0.2661

Kemudian, dilakukan prediksi pada model, dari hasil prediksi dibuat *confusion matrix* yang dapat dilihat pada gambar 4.26



Gambar 4. 26 confusion matrix InceptionV3 eksperimen 2

Pada gambar 4.26 dapat dilihat hasil prediksi dari model InceptionV3, prediksi dilakukan menggunakan 5512 citra dengan perbandingan 50:50. Untuk citra *parasitized* berhasil terprediksi sejumlah 2665 (96.7%) dari 2756 citra dan *uninfected* berhasil terprediksi sejumlah 2634 (95.6%) dari 2756 citra. Dari hasil prediksi juga dibuat F1-score yang dapat dilihat pada tabel 4.8

Tabel 4. 18 F1-score Inceptionv3 eksperimen 3

	Precision	recall	Class F1-score	support
0	0.96	0.97	0.96	2756
1	0.97	0.96	0.96	2756
Accuracy			0.96	5512
Model F1-Score			0.96	5512

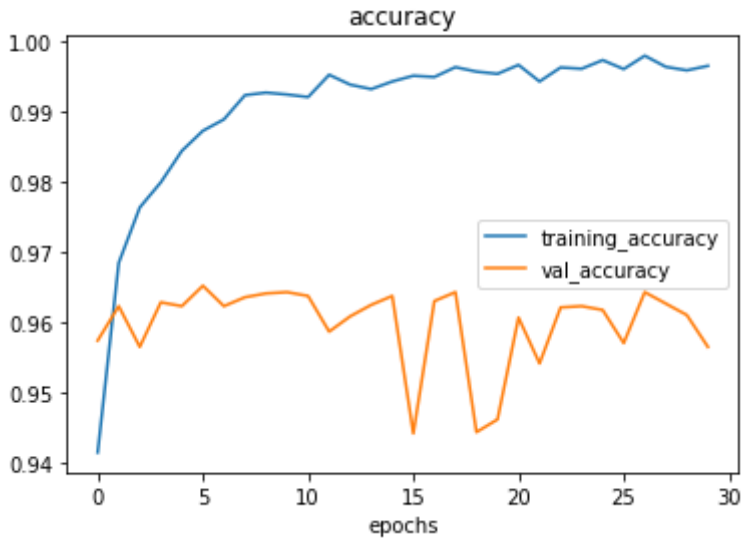
Dari tabel f1-score dapat diketahui nilai akurasi prediksi sebesar 0.87, nilai f1-score pada *class 0(parasitized)* adalah 0.85, dan nilai f1-score pada *class 1(uninfected)* adalah 0.88 , dan model f1-score adalah 0.865

4.4 Eksperimen 4

Eksperimen 4 dilakukan dengan menggunakan parameter ukuran citra 96x96, 30 epoch, fine tune, optimizer Adam dengan learning rate 1e-4 dan tanpa . Berikut adalah hasilnya.

4.4.1 ResNet50v2

Hasil training:



Gambar 4. 27 Grafik akurasi ResNet50v2 eksperimen 4

Gambar 4. 28 Grafik loss ResNet50v2 eksperimen 4

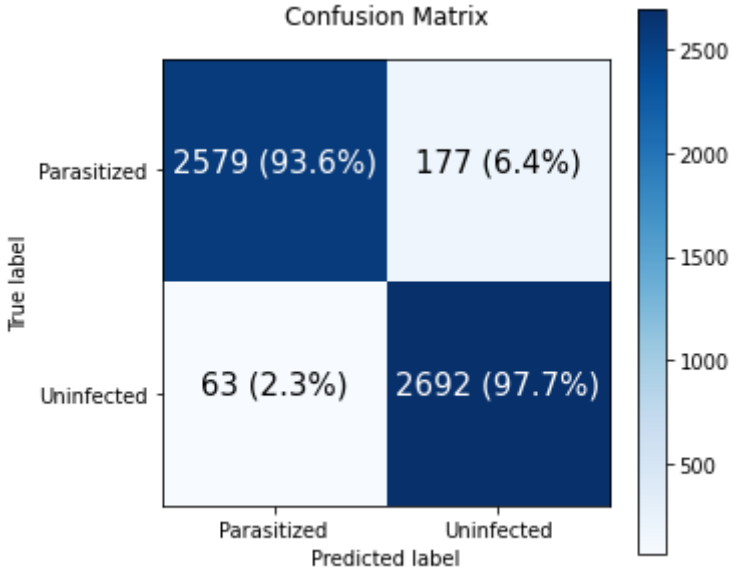
Pada gambar 4.27 dapat diketahui bahwa *training accuracy* meningkat dari 0.94 pada saat training dimulai menjadi 0.99 pada epoch ke-30, namun *validation accuracy* tidak

mengalami peningkatan . Pada gambar 4.28, grafik *training loss* menunjukan penurunan , yang semula bernilai 0.4 menjadi 0.1 pada epoch ke-30, *validation loss* tidak mengalami penurunan . Hasil akhir dari training dapat dilihat pada tabel 4.19

Tabel 4. 19 Hasil akhir training ResNet50v2 eksperimen 4

Train accuracy	*Train loss*	*Validation accuracy*	*Validation loss*
0.9965	0.1117	0.9565	0.3351

Kemudian, dilakukan prediksi pada model, dari hasil prediksi dibuat *confusion matrix* yang dapat dilihat pada gambar 4.29



Gambar 4. 29 Confusion matrix ResNett50v2 eskperimen 4

Pada gambar 4.29 dapat dilihat hasil prediksi dari model ResNet50v2, prediksi dilakukan menggunakan 5512 citra dengan perbandingan 50:50. Untuk citra *parasitized* berhasil terprediksi sejumlah 2579 (93.6%) dari 2756 citra dan *uninfected* berhasil terprediksi sejumlah 2692 (97.7%) dari 2756 citra. Dari hasil prediksi juga dibuat F1-score yang dapat dilihat pada tabel 4..20

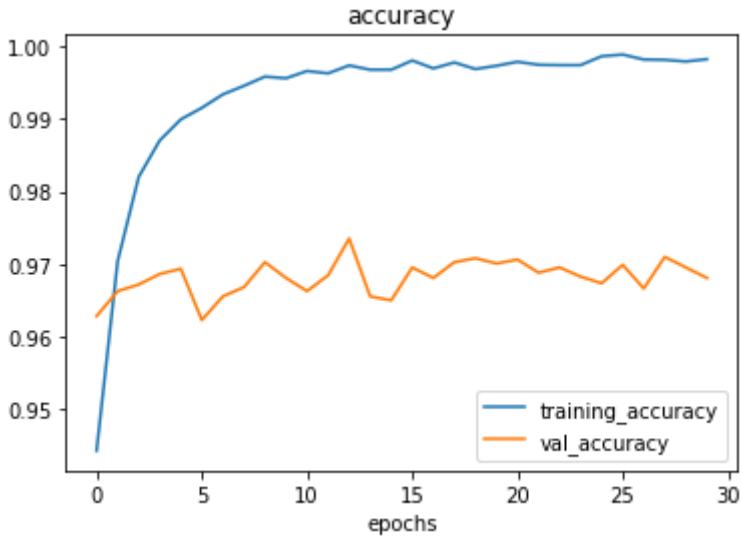
Tabel 4. 20 F1-score ResNet50v2 eksperimen 4

	Precision	recall	Class F1-score	support
0	0.98	0.94	0.96	2756
1	0.94	0.98	0.96	2756
Accuracy			0.96	5512
Model F1-Score			0.96	5512

Dari tabel f1-score dapat diketahui nilai akurasi prediksi sebesar 0.96, nilai f1-score pada *class 0(parasitized)* adalah 0.96, dan nilai f1-score pada *class 1(uninfected)* adalah 0.996 dan model f1-score adalah 0.96

4.4.2 EfficientNetB0

Hasil training:



Gambar 4. 30 Grafiik EfficientNetB0 eksperimen 4

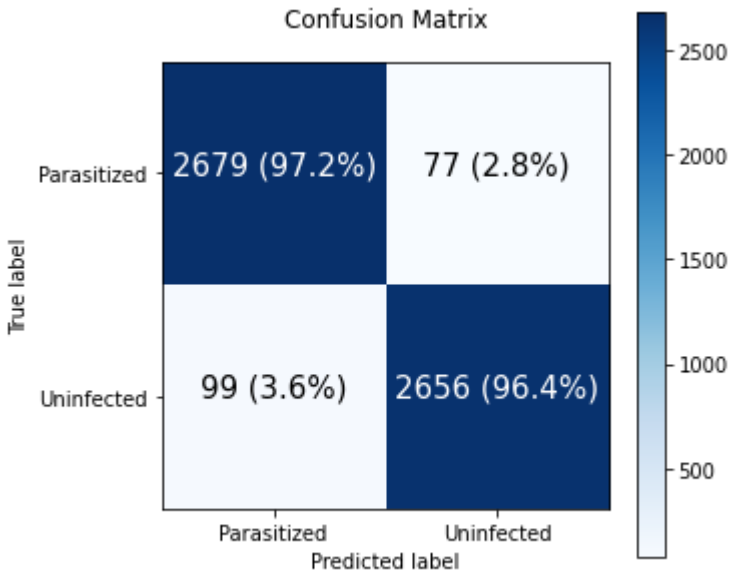
Gambar 4. 31 Grafik loss EfficientNetB0 eksperimen 4

Pada gambar 4.30 dapat diketahui bahwa *training accuracy* meningkat seiring bertambahnya epoch, sedangkan *validation accuracy* tidak mengalami peningkatan . Pada gambar 4.31, grafik *training loss* menunjukan penurunan, seiring bertambahnya epoch, dan *validation loss* justru meningkat. Hasil akhir dari training dapat dilihat pada tabel 4.15

Tabel 4. 21 Hasil akhir training EfficientNetB0 eksperimen 3

Train accuracy	*Train loss*	*Validation accuracy*	*Validation loss*
0.9565	0.1254	0.9681	0.3076

Kemudian, dilakukan prediksi pada model, dari hasil prediksi dibuat *confusion matrix* yang dapat dilihat pada gambar 4.32



Gambar 4. 32 Confusion matrix EfficientNetB0 eksperimen 4

Pada gambar 4.32 dapat dilihat hasil prediksi dari model EfficientNetB0, prediksi dilakukan menggunakan 5512 citra dengan perbandingan 50:50. Untuk citra *parasitized* berhasil terprediksi sejumlah 2679 (97,2%) dari 2756 citra dan *uninfected* berhasil terprediksi sejumlah 2656(98.2%) dari 2756 citra. Dari hasil prediksi juga dibuat F1-score yang dapat dilihat pada tabel 4.22

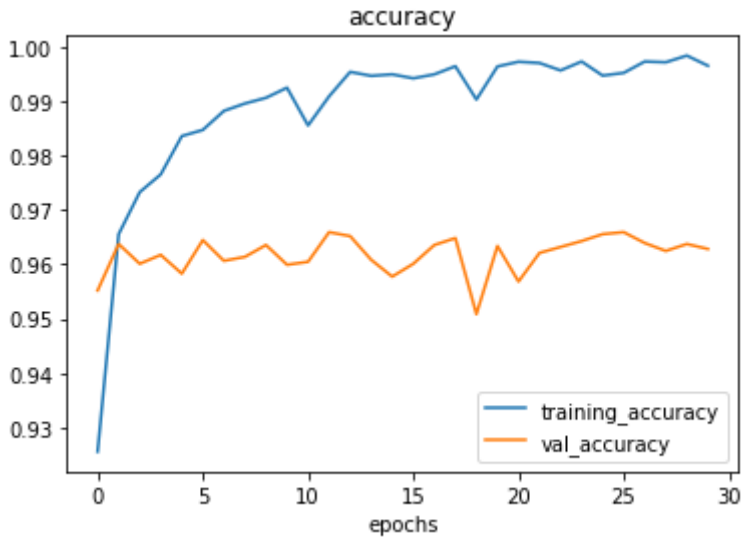
Tabel 4. 22 F1-score EfficientNetB0 eksperimen 4

	Precision	recall	Class F1-score	support
0	0.96	0.96	0.97	2756
1	0.97	0.97	0.97	2756
Accuracy			0.97	5512
Model F1-Score			0.97	5512

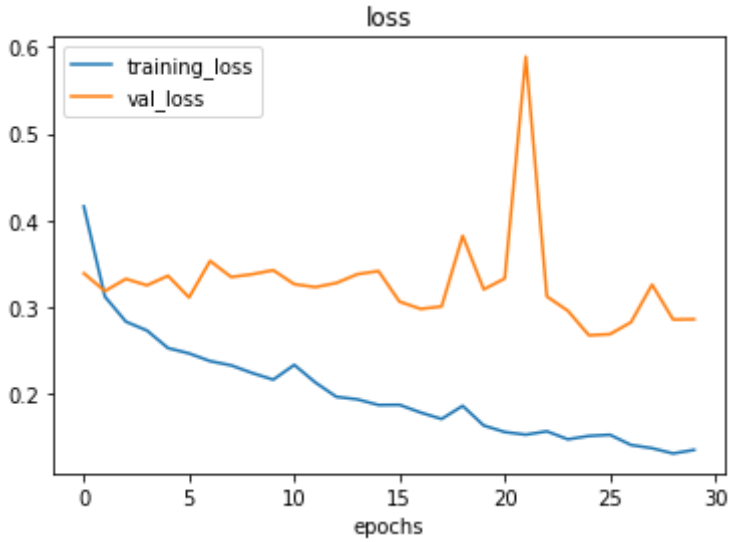
Dari tabel f1-score dapat diketahui nilai akurasi prediksi sebesar 0.97, nilai f1-score pada *class 0(parasitized)* adalah 0.97, dan nilai f1-score pada *class 1(uninfected)* adalah 0.97 dan model f1-score adalah 97.

4.5.3 InceptionV3

Hasil training:



Gambar 4. 33 Grafik akurasi InceptionV3 eksperimen 4



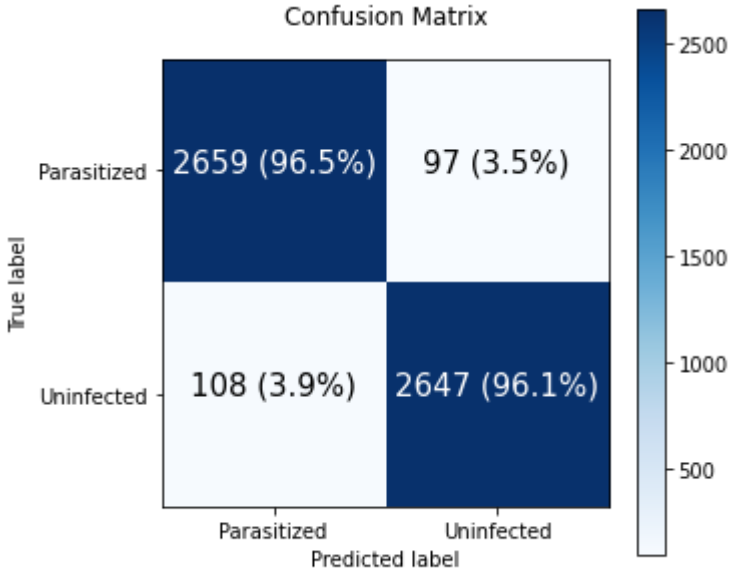
Gambar 4. 34 Grafik loss InceptionV3 eksperimen 4

Pada gambar 4.33 dapat diketahui bahwa *training accuracy* meningkat seiring bertambahnya epoch dan dan tidak ada peningkatan dari *validation accuracy*. Pada gambar 4.34, *training loss* mengalami penurunan seiring bertambahnya epoch, dan *validation loss* tidak mengalami penurunan . Hasil akhir dari training dapat dilihat pada tabel 4.23

Tabel 4. 23 Hasil akhir training inceptionv3 eksperimen 3

Train accuracy	*Train loss*	*Validation accuracy*	*Validation loss*
0.9965	0.1350	0.9628	0.2859

Kemudian, dilakukan prediksi pada model, dari hasil prediksi dibuat *confusion matrix* yang dapat dilihat pada gambar 4.35



Gambar 4. 35 confusion matrix InceptionV3 eksperimen 4

Pada gambar 4.35 dapat dilihat hasil prediksi dari model InceptionV3, prediksi dilakukan menggunakan 5512 citra dengan perbandingan 50:50. Untuk citra *parasitized* berhasil terprediksi sejumlah 2659 (96.5%) dari 2756 citra dan *uninfected* berhasil terprediksi sejumlah 2647 (96.1%) dari 2756 citra. Dari hasil prediksi juga dibuat F1-score yang dapat dilihat pada tabel 4.24

Tabel 4. 24 F1-score Inceptionv3 eksperimen 4

	Precision	recall	Class F1-score	support
0	0.96	0.96	0.96	2756
1	0.96	0.96	0.96	2756
Accuracy			0.96	5512
Model F1-Score			0.96	5512

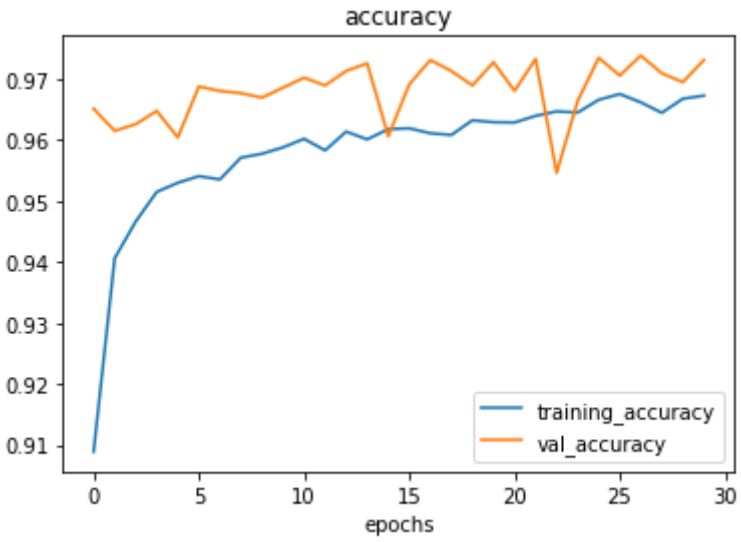
Dari tabel f1-score dapat diketahui nilai akurasi prediksi sebesar 0.96, nilai f1-score pada *class 0(parasitized)* adalah 0.96, dan nilai f1-score pada *class 1(uninfected)* adalah 0.96 dan model f1-score adalah

4.5 Eksperimen 5

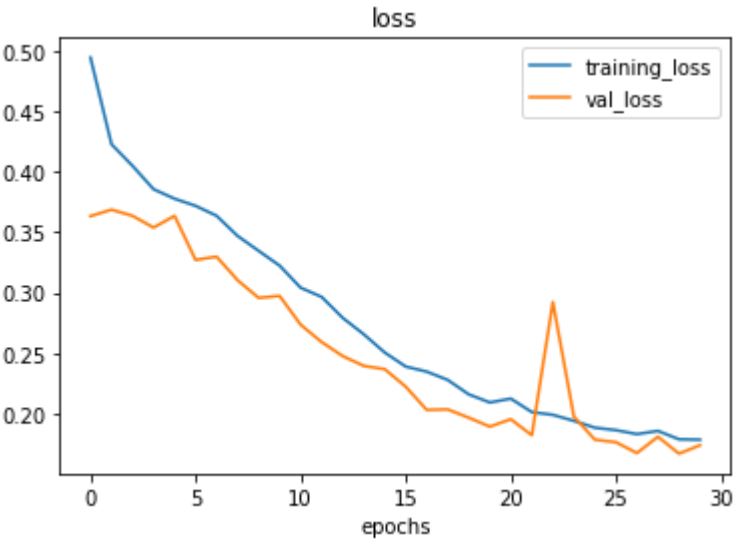
Eksperimen 5 dilakukan dengan menggunakan parameter ukuran citra 96x96, 30 epoch, fine tune, optimizer Adam dengan learning rate 1e-4. Berikut adalah hasilnya.

4.5.1 ResNet50v2

Hasil training:



Gambar 4. 36 Grafik akurasi ResNet50v2 eksperimen 5



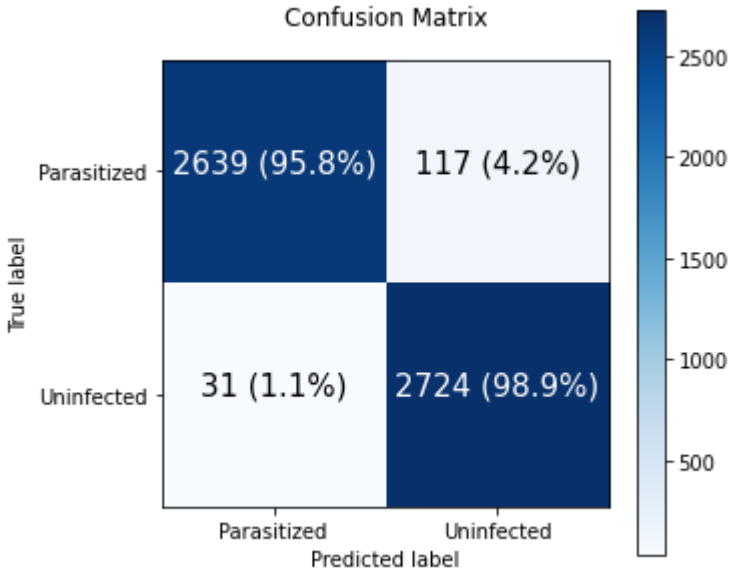
Gambar 4. 37 Grafik loss ResNet50v2 eksperimen 5

Pada gambar 4.27 dapat diketahui bahwa *training accuracy* meningkat dari 0.91 pada saat training dimulai menjadi 0.96 pada epoch ke-30, namun *validation accuracy* tidak mengalami peningkatan n. Pada gambar 4.28, grafik *training loss* menunjukan penurunan , yang semula bernilai 0.50 menjadi 0.2 pada epoch ke-30, *validation loss* juga mengalami penurunan . Hasil akhir dari training dapat dilihat pada tabel 4.19

Tabel 4. 25 Hasil akhir training ResNet50v2 eksperimen 5

Train accuracy	*Train loss*	*Validation accuracy*	*Validation loss*
0.9637	0.2119	0.9710	0.1891

Kemudian, dilakukan prediksi pada model, dari hasil prediksi dibuat *confusion matrix* yang dapat dilihat pada gambar 4.29



Gambar 4. 38 Confusion matrix ResNett50v2 eskperimen 5

Pada gambar 4.29 dapat dilihat hasil prediksi dari model ResNet50v2, prediksi dilakukan menggunakan 5512 citra dengan perbandingan 50:50. Untuk citra *parasitized* berhasil terprediksi sejumlah 2639 (95.8%) dari 2756 citra dan *uninfected* berhasil terprediksi sejumlah 2724 (98.9%) dari 2756 citra. Dari hasil prediksi juga dibuat F1-score yang dapat dilihat pada tabel 4..20

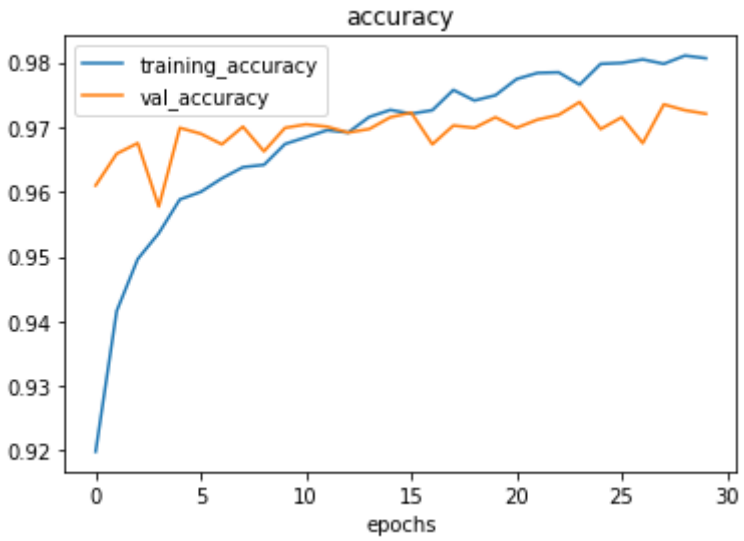
Tabel 4. 26 F1-Score ResNet50v2 eksperimen 4

	Precision	recall	Class F1-score	support
0	0.99	0.96	0.97	2756
1	0.96	0.99	0.97	2756
Accuracy			0.975	5512
Model F1-Score			0.97	5512

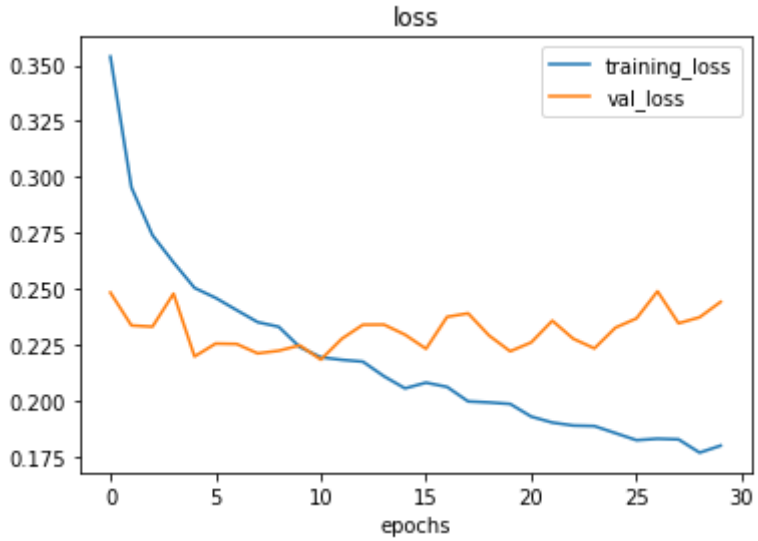
Dari tabel f1-score dapat diketahui nilai akurasi prediksi sebesar 0.975, nilai f1-score pada *class 0(parasitized)* adalah 0.97, dan nilai f1-score pada *class 1(uninfected)* adalah 0.97 dan model f1-score adalah 0.97

4.5.2 EfficientNetB0

Hasil training:



Gambar 4. 39 Grafiik EfficientNetB0 eksperimen 5



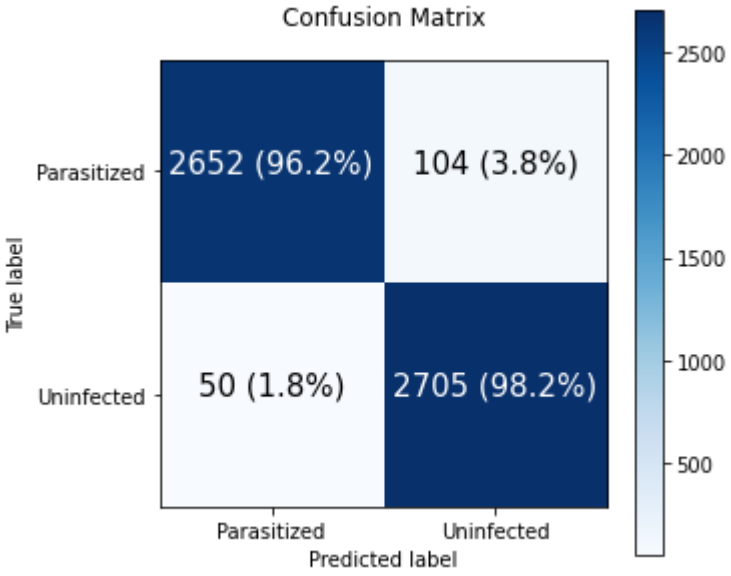
Gambar 4. 40 Grafik loss EfficientNetB0 eksperimen 5

Pada gambar 4.30 dapat diketahui bahwa *training accuracy* meningkat n seiring bertambahnya epoch, sedangkan *validation accuracy* tidak mengalami peningkatan . Pada gambar 4.31, grafik *training loss* menunjukan penurunan , seiring bertambahnya epoch, dan *validation loss* tidak mengalami penurunan. Hasil akhir dari training dapat dilihat pada tabel 4.15

Tabel 4. 27 Hasil akhir training EfficientNetB0 eksperimen 5

Train accuracy	*Train loss*	*Validation accuracy*	*Validation loss*
0.9806	0.1800	0.9721	0.2442

Kemudian, dilakukan prediksi pada model, dari hasil prediksi dibuat *confusion matrix* yang dapat dilihat pada gambar 4.32



Gambar 4. 41 Confusion matrix EfficientNetB0 eksperimen 5

Pada gambar 4.32 dapat dilihat hasil prediksi dari model EfficientNetB0, prediksi dilakukan menggunakan 5512 citra dengan perbandingan 50:50. Untuk citra *parasitized* berhasil terprediksi sejumlah 2652 (96,2%) dari 2756 citra dan *uninfected* berhasil terprediksi sejumlah 2705(98.2%) dari 2756 citra. Dari hasil prediksi juga dibuat F1-score yang dapat dilihat pada tabel 4.22

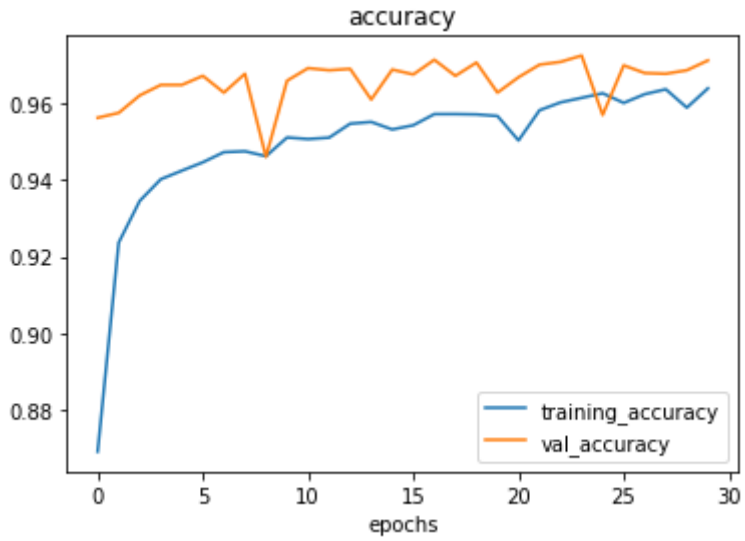
Tabel 4. 28 F1-score EfficientNetB0 eksperimen 5

	Precision	recall	Class F1-score	support
0	0.98	0.96	0.97	2756
1	0.96	0.98	0.97	2756
Accuracy			0.97	5512
Model F1-Score			0.97	5512

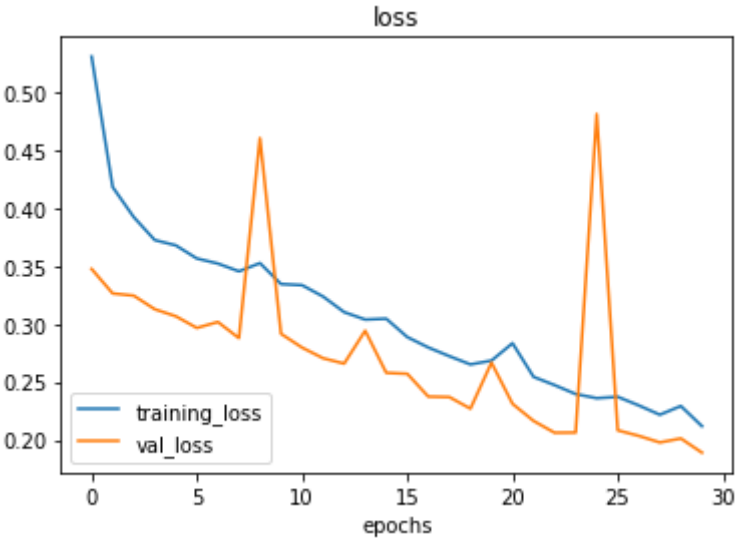
Dari tabel f1-score dapat diketahui nilai akurasi prediksi sebesar 0.97, nilai f1-score pada *class 0(parasitized)* adalah 0.97, dan nilai f1-score pada *class 1(uninfected)* adalah 0.97 dan model f1-score adalah 97.

4.5.3 InceptionV3

Hasil training:



Gambar 4. 42 Grafik akurasi InceptionV3 eksperimen 5



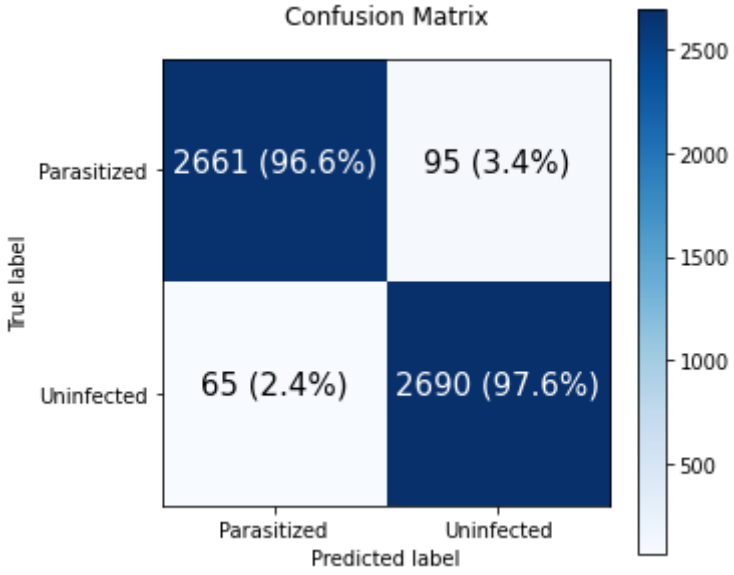
Gambar 4. 43 Grafik loss InceptionV3 eksperimen 5

Pada gambar 4.33 dapat diketahui bahwa *training accuracy* meningkat n seiring bertambahnya epoch dan dan tidak ada peningkatan signifikan dari *validation accuracy*. Pada gambar 4.34, *training loss* mengalami penurunan seiring bertambahnya epoch, hal yang sama juga terjadi pada *validation loss* meskipun sempat terjadi fluktuasi yang sangat tinggi pada epoch ke-7 dan 24 . Hasil akhir dari training dapat dilihat pada tabel 4.23

Tabel 4. 29 Hasil akhir training inceptionv3 eksperimen 5

Train accuracy	*Train loss*	*Validation accuracy*	*Validation loss*
0.9637	0.2119	0.9710	0.1891

Kemudian, dilakukan prediksi pada model, dari hasil prediksi dibuat *confusion matrix* yang dapat dilihat pada gambar 4.35



Gambar 4. 44 confusion matrix InceptionV3 eksperimen 5

Pada gambar 4.35 dapat dilihat hasil prediksi dari model InceptionV3, prediksi dilakukan menggunakan 5512 citra dengan perbandingan 50:50. Untuk citra *parasitized* berhasil terprediksi sejumlah 2661 (96.6%) dari 2756 citra dan *uninfected* berhasil terprediksi sejumlah 2690 (97.6%) dari 2756 citra. Dari hasil prediksi juga dibuat F1-score yang dapat dilihat pada tabel 4.24

Tabel 4. 30 F1-score Inceptionv3 eksperimen 5

	Precision	recall	Class F1-score	support
0	0.98	0.97	0.97	2756
1	0.97	0.98	0.97	2756
Accuracy			0.97	5512
Model F1-Score			0.97	5512

Dari tabel f1-score dapat diketahui nilai akurasi prediksi sebesar 0.97, nilai f1-score pada *class 0(parasitized)* adalah 0.97, dan nilai f1-score pada *class 1(uninfected)* adalah 0.97 dan model f1-score adala

4.6 Perbandingan Model

Untuk membandingkan model, parameter yang digunakan adalah hasil akhir training model (*training loss*, *validation loss*, *training accuracy*, dan *validation loss)* dan performa model dalam melakukan prediksi.

4.6.1 Perbandingan hasil training

Fill hijau merupakan nilai rata-rata dan fill warna biru merupakan model dengan metrics terbaik

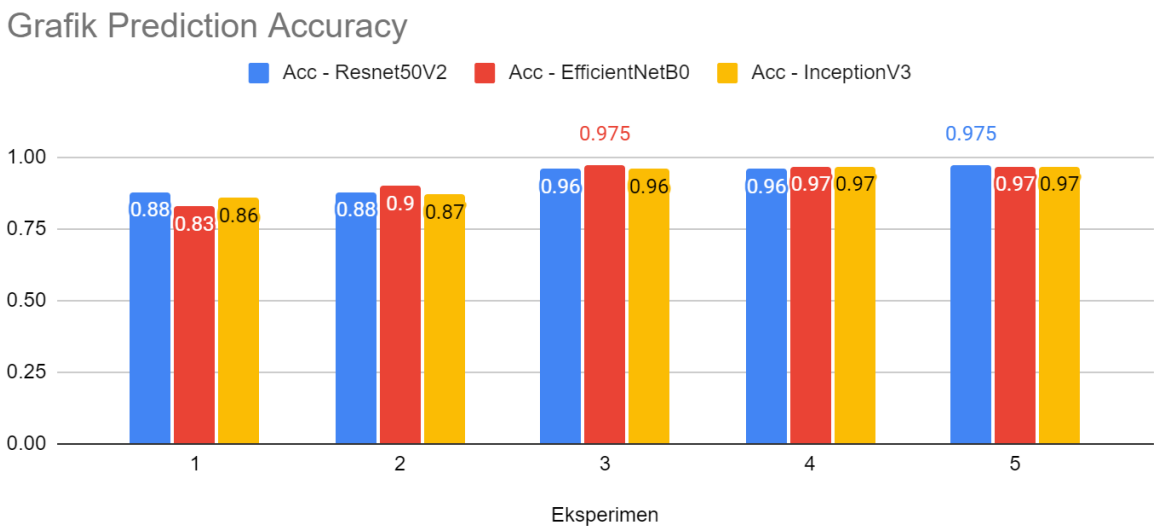
Tabel 4. 31 Perbandingan hasil training

Model	Eksperimen	*Train Accuracy*	*Training Loss*	*Validation Accuracy*	*Validation Loss*
ResNet50V2	1	0.8543	0.3571	0.8765	0.3029
	2	0.8915	0.2677	0.8826	0.2661
	3	0.9400	0.2300	0.9599	0.1689
	4	0.9965	0.1117	0.9565	0.3351
	5	0.9637	0.2119	0.9710	0.1891
Avg		0.9292	0.23568	0.9293	0.25242
EfficientNetB0	1	0.8841	0.2814	0.8173	0.4106
	2	0.8987	0.2577	0.9009	0.2552
	3	0.9663	0.1345	0.9731	0.1125
	4	0.9982	0.1254	0.9681	0.3076
	5	0.9806	0.1800	0.9721	0.2442
Avg		0.94558	0.1958	0.9263	0.26602
InceptionV3	1	0.8510	0.3630	0.8636	0.3183
	2	0.8582	0.3291	0.8670	0.3058
	3	0.9378	0.2283	0.9614	0.2661
	4	0.9965	0.1350	0.9628	0.2859
	5	0.9637	0.2119	0.9710	0.1891
Avg		0.92144	0.25346	0.92516	0.27304

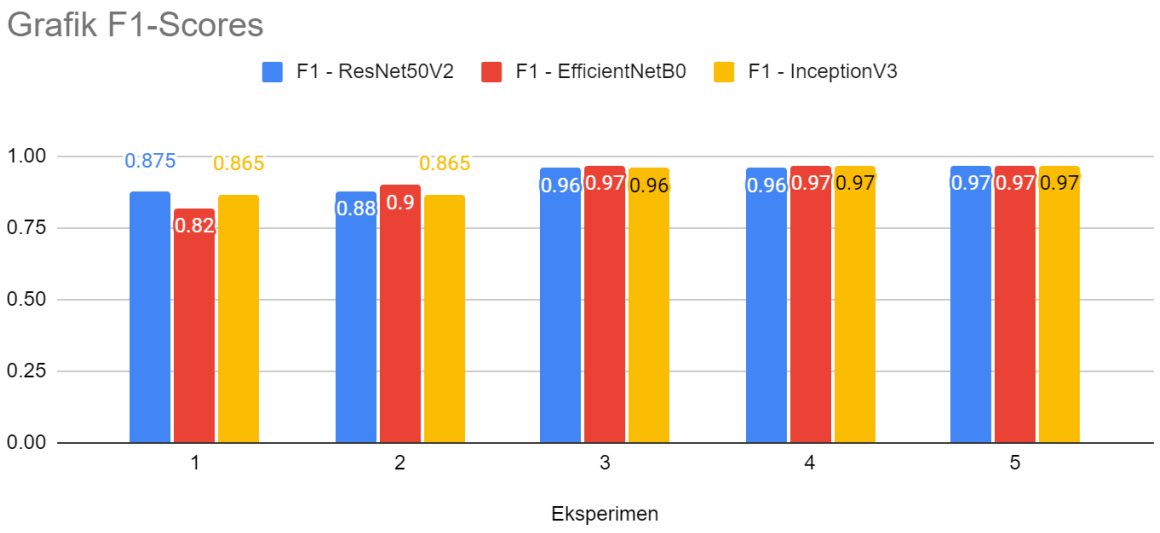
4.6.2 Perbandingan hasil prediksi model

Tabel 4. 32 Perbandingan hasil prediksi

Model	Eksperimen	*Parasitized*	*Uninfected*	*Prediction Accuray*	*F1-Score*
ResNet50V2	1	0.81	0.95	0.88	0.875
	2	0.79	0.98	0.88	0.88
	3	0.94	0.98	0.96	0.96
	4	0.94	0.98	0.96	0.96
	5	0.96	0.99	0.975	0.97
Avg		0.888	0.976	0.931	0.929
EfficientNetB0	1	0.67	0.98	0.83	0.82
	2	0.89	0.91	0.90	0.90
	3	0.96	0.99	0.975	0.97
	4	0.97	0.96	0.97	0.97
	5	0.96	0.98	0.97	0.97
Avg		0.89	0.964	0.929	0.926
InceptionV3	1	0.77	0.96	0.86	0.865
	2	0.77	0.96	0.87	0.865
	3	0.97	0.96	0.96	0.96
	4	0.96	0.96	0.97	0.97
	5	0.97	0.98	0.97	0.97
Avg		0.888	0.964	0.926	0.926


Grafik Prediction Accuracy
Acc - Resnet50V2
Acc - EfficientNetB0
Acc - InceptionV3
1.00
0.75
0.50
0.25
0.00
0.975
0.975
0.88
0.83
0.86
0.88
0.9
0.87
0.96
0.96
0.96
0.97
0.97
0.97
0.97
1
2
3
4
5
Eksperimen

Gambar 4. 45 Grafik Perbandingan *Prediction Accuracy*



Gambar 4. 46 Grafik perbandingan F1-Score

4.7 Diskusi

Pada eksperimen pertama dilakukan untuk menguji performa model awal transfer learning Dari eksperimen pertama diperoleh hasil training model untuk *train accuracy* sebesar 0.854 ,0.88, dan 0.851 (ResNet50v2, EfficientNetB0, dan InceptionV3) dan *val accuracy* sebesar 0.876, 0.817, dan 0.83. Eksperimen kedua dilakukan untuk meningkatkan performa model dari eksperimen sebelumnya. Dilakukan penambahan epoch, pengurangan nilai lr(*learning rate),* dan penamabahan ukuran pada citra input . Hasil dari eksperimen kedua terjadi peningkatan pada *train accuracy* dan *validation accuracy* untuk semua model.

Pada eksperimen ketiga dilakukan penambahan epoch , dilakukan *fine-tune* dan lr *optimizer* yang diatur ke default. Hasil dari eksperimen ketiga terjadi peningkatan pada *train accuracy* dan *val accuracy* , meskipun sempat terjadi fluktuasi yang cukup tinggi pada *val loss* ResNet50v2 dan InceptionV3 . Range fluktuasi berkisar 0-50 untuk ResNet50v2 menjelang epoch ke-10 dan untuk InceptionV3 berkisar 0-200 menjelang epoch ke – 10.

Pada eksperimen ke-empat sama dengan eksperimen sebelumnya namun dilakukan tanpa augmentasi dan pengurangan learning rate . Hasil dari eksperimen ke-empat berhasil menyelesaikan fluktuasi pada *val loss*, namun tidak ada peningkatan pada *val accuracy* dan model secara keseluruhan mengalami sedikit *overfitting* . Pada eksperimen kelima sama

dengan eksperimen sebelumnya namun ditambahkan augmentasi data. Hasil dari eksperimen kelima, model ResNet50v2 dan EfficientNetB0

memiliki hasil *training accuracy* dan *val accuracy* yang relative tinggi dan memiliki gap yang dekat namun untuk model InceptionV3 masih terjadi sedikit overfitting.

Bab V
Kesimpulan dan Saran

5.1 Simpulan

Pada saat model training, model dapat mencapai akurasi yang relatif tinggi pada saat training dimulai (epoch ke-1). Jika tanpa fine tune rentang akurasi berkisar 0.76 – 0.81 untuk ResNet50v2, 0.76 – 0.80 untuk EfficientNetB0, dan 0.77 – 0.82 untuk. Jika dengan fine tune. Pada eksperimen ke-3 terjadi fluktuasi yang tinggi pada model resnet dan inception, hal ini bisa saja terjadi karena learning rate optimizer yang kurang tepat. Dari semua eksperimen yang dilakukan Model dengan metrics terbaik adalah EfficientnetB0 pada eksperimen 3 dan ResNet50v2 eksperimen 5 berdasarkan prediction accuracy dan f1-score.

5.2 Saran

Untuk penelitian berikutnya, peneliti dapat melakukan resize pada citra dataset terlebih dahulu untuk meningkatkan performa model. Citra dapat di resize agar sesuai dengan default input dari *pre-trained* model, 224x224 untuk ResNet50v2 dan EfficientNetB0. Sedangkan untuk InceptionV3 citra dapat di resize menjadi 299x299. Selain me-resize citra peneliti dapat melakukan hyper-parameter tuning seperti menambahkan batch-size dan mengurangi learning rat

DAFTAR PUSTAKA

alodokter. (2019). *Malaria - Gejala,penyebab, dan mengobati*. Retrieved November 7, 2021, from https://www.alodokter.com/malaria

Bonaccorso, G. (2018). *Machine Learning ALgorithm* (Ke2 ed.). Birmingham: Packt Publishing.

Brownlee, J. (2020). *What is machine learning?* Retrieved 11 12, 2021, from https://machinelearningmastery.com/what-is-machine-learning/

Brownlee, J. (2021, Desember 1). *A Gentle Introduction to Transfer Learning for Deep Learning*. Retrieved from machinelearningmastery.com: https://machinelearningmastery.com/transfer-learning-for-deep-learning/

Christopher, A. (2021, Februari 2). *K-Nearest Neighbour, A complete explatantion of K-NN*. Retrieved from medium.com: https://medium.com/swlh/k-nearest-neighbor-ca2593d7a3c4

Cover, T. M., & Hart, P. E. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory. 13 (1)*, 21-27.

Dicoding. (2020). *Apa itu machine learning? Beserta Pengertian dan Cara Kerjanya*. Retrieved November 12, 2021, from https://www.dicoding.com/blog/machine-learning-adalah/

Fix, E., & Hodges, J. L. (1989). Discriminatory analysis. Nonparametric discrimination: Consistency properties. *International Statistical Review/Revue Internationale de Statistique , 57(3)*, 238-247.

Fumo, D. (2017). *Types of Machine Learning You Should Know*. Retrieved 11 13, 2021, from https://towardsdatascience.com/types-of-machine-learning-algorithms-you-should-know-953a08248861

Gudimella, A., Story, R., Shaker, M., Kong, R., Brown, M., Shnayder, V., & Campos, M. (2017). Deep Reinforcement Learning for Dexterous Manipulation with Concept Networks. *arXiv:1709.06977[cs.AI]*.

Gulli, A., Kapoor, A., & Pal, S. (2019). *Deep Learning With Tensorflow 2 and Keras.* Birmingham: Packt Publishing.

Han, X., Zhang, Z., Ding, N., Gu, Y., Liu, X., Huo, Y., & Qiu, J. (2021). Pre-trained models: Past, present and future. *AI Open.*

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770-778.

Hinton, G. E., Osindero, S., & Teh, Y.-W. (2006). A Fast Learning Algorithm for Deep Belief Nets. *Neural Computation*, 1527–1554.

IBM. (2020). *What is machine learning?* Retrieved November 7, 2021, from https://www.ibm.com/cloud/learn/machine-learning

KemenkesRI. (2019). *Profil Kesehatan Indonesia.* Kementerian Kesehatan Republik Indonesia.

Kim, P. (2017). Convolutional Neural Network. In P. Kim, *MATLAB Deep Learning* (pp. 121-147). Berkeley, CA: Apress.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. *Advances in neural information processing systems, 25*, 1097 - 1105.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature 521, no. 7553*, 436-444.

LeCun, Y., Boser, B., Denker, J. S., Hendersen, D., Howard, R. E., & Hubbard, W. (1989). Handwritten Digit Recognition with a Back-Propagation Network. *Advances in neural information processing systems*.

Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE 86(11)*, 2278-2324.

pandas - Python Data Analisys Library. (2021, Desember 1). Retrieved from https://pandas.pydata.org/: https://pandas.pydata.org/

Paperswithcode. (2015, Desember). *ImageNet Benchmark (Image Classification)*. Retrieved from Papers With Code: https://paperswithcode.com/sota/image-classification-on-imagenet

paperswithcode. (2020). *CIFAR-100 Benchmark (Image Classification)*. Retrieved from paperswithcode: https://paperswithcode.com/sota/image-classification-on-cifar-100

PythonInstitute. (2021, Desember 1). *About Python*. Retrieved from https://pythoninstitute.org: https://pythoninstitute.org/what-is-python/

Reddy, A. S., & Juliet, D. S. (2019). Transfer learning with ResNet-50 for malaria cell-image classification. *International Conference on Communication and Signal Processing (ICCSP)* (pp. 0945-0949). Chennai, India: IEEE.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 386–408.

Samuel, A. (1959). Some Studies in Machine Learning Using the Game of Checkers. *IBM Journal of Research and Development*, 210 - 229.

ShylmaNa'imah. (2021). *Penyakit Malaria: Gejala,Penyebab, Hingga Pengobatan*. Retrieved November 7, 2021, from https://hellosehat.com/infeksi/infeksi-serangga/penyakit-malaria/

Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Neural Networks for large-scale image recognition. *arXiv preprint*, arXiv:1409.1556.

StackOverflowInsight. (2021, Desember 1). *Stack Overflow Developer Survey 2021*. Retrieved from insights.stackoverflow.com: https://insights.stackoverflow.com/survey/2021#most-popular-technologies-language

StatisticsSolutions. (2021). *What is Linear Regression*. Retrieved from statisticssolutions: https://www.statisticssolutions.com/free-resources/directory-of-statistical-analyses/what-is-linear-regression/

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., . . . Rabinovich, A. (2015). Going deeper with convolutions. *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1-9). Boston, MA: IEEE.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the Inception Architecture for Computer Vision. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2818-2826.

Tan, M., & Le, Q. V. (2019). Efficientnet: Rethinking model scaling for convolutional neural networks. *International Conference on Machine Learning*, 6105-6114.

Tyagi, N. (2021). *6 Major Branches of Artificial Intelligence*. Retrieved November 12, 2021, from https://www.analyticssteps.com/blogs/6-major-branches-artificial-intelligence-ai

Viso.AI. (2021). *Deep Residual Networks*. Retrieved from viso.ai: https://viso.ai/deep-learning/resnet-residual-neural-network/

Wang, Z. J., Turko, R., Shaikh, O., Park, H., Das, N., Hohman, F., . . . Chau, D. H. (2021). CNN Explainer: Learning Convolutional Neural Networks with Interactive Visualization. *IEEE Transactions on Visualization and Computer Graphics, 27(2)*, 1396 - 1406.

Werbos, P. (1990). Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 1550 - 1560.

WHO. (2020). *World Malaria Report.* World Health Organization.

WHO. (2021). *Malaria*. Retrieved November 7, 2021, from https://www.who.int/news-room/fact-sheets/detail/malaria

ZHUANG, F., QI, Z., DUAN, K., XI, D., ZHU, Y., XIONG, H., & HE, Q. (2020). A comprehensive survey on transfer learning. *Proceedings of the IEEE, 109(1)*, 43-76.