RANCANG BANGUN APLIKASI UNTUK MENDETEKSI PARASIT MALARIA PADA CITRA APUSAN DARAH TIPIS DENGAN MENERAPKAN TRANSFER LEARNING PADA CONVOLUTIONAL NEURAL NETWORK

TUGAS AKHIR

BENEDICTUS JULLIAN PRADANA

311810007

TEKNIK INFORMATIKA

FAKULTAS SAINS DAN TEKNOLOGI

UNIVERSITAS MA CHUNG

2022

LEMBAR PENGESAHAN

TUGAS AKHIR

RANCANG BANGUN APLIKASI UNTUK MENDETEKSI PARASIT MALARIA PADA CITRA APUSAN DARAH TIPIS DENGAN MENERAPKAN TRANSFER LEARNING PADA CONVOLUTIONAL NEURAL NETWORK

Oleh:

BENEDICTUS JULLIAN PRADANA

NIM. 311810007

dari:

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS SAINS dan TEKNOLOGI

UNIVERSITAS MA CHUNG

Telah dinyatakan lulus dalam melaksanakan Tugas Akhir sebagai syarat kelulusan dan berhak mendapatkan gelar Sarjana S.Kom.

Dosen Pembimbing 1,

Windra Swastika, S.Kom., MT., Ph.D.
NIP. 20070039

Dosen Pembimbing 2,

Ir.Oesman Hendra Kelana, M.Cs.
NIP. 20110022

Dekan Fakultas Sains dan Teknologi,

Dr. Kestrilia Rega Prilianti, M.Si.
NIP. 20120035

PERNYATAAN KEASLIAN TUGAS AKHIR

Dengan ini saya menyatakan bahwa isi sebagian maupun keseluruhan Tugas Akhir saya dengan "Rancang Bangun Aplikasi untuk Mendeteksi Malaria Pada Citra Apusan Darah Tipis Dengan Menerapkan Transfer Learning Pada Convolutional Neural Network" adalah benar benar hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diizinkan dan bukan merupakan karya pihak lain yang saya akui sebagai karya sendiri.

Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap pada daftar pustaka. Apabila ternyata pernyataan ini tidak benar, saya bersedia menerima sanksi sesuai peraturan yang berlaku.

Malang, 28 Juli 2022

Benedictus Jullian Pradana
NIM. 311810007

RANCANG BANGUN APLIKASI UNTUK MENDETEKSI PARASIT MALARIA PADA CITRA APUSAN DARAH TIPIS DENGAN MENERAPKAN TRANSFER LEARNING PADA CONVOLUTIONAL NEURAL NETWORK

Benedictus J. Pradana, Windra Swastika, Oesman H. Kelana
Universitas Ma Chung

Abstrak

Malaria merupakan penyakit menular yang disebabkan oleh parasit Plasmodium. Pada tahun 2019 terdapat 229 juta kasus malaria dengan korban jiwa mencapai 400.900 jiwa. Kasus Malaria meningkat pada tahun 2020 menjadi 241 juta jiwa dengan korban jiwa mencapai 627.000 jiwa. Diagnosis Malaria yang dilakukan dengan pengamatan sampel darah pasien membutuhkan tenaga ahli dan jika tidak dilakukan dengan benar dapat terjadi kesalahan diagnosis.

Deep Learning dapat digunakan untuk membantu diagnosis Malaria dengan melakukan klasifikasi pada citra apusan darah tipis. Pada penelitian ini digunakan teknik *Transfer Learning* pada *Convolutional Neural Network* untuk mempercepat proses pelatihan model dan mendapat akurasi yang tinggi. Arsitektur yang digunakan untuk Transfer Learning adalah EfficientNetB0.

Model hasil latih ditanamkan pada aplikasi web berbasis python yang kemudian di*deploy* pada platform Google App Engine. Hal ini dilakukan agar dapat digunakan oleh tenaga ahli untuk membantu diagnosis. Model hasil latih memiliki *training accuracy* sebesar 0,9664, *training loss* sebesar 0,0937, *validation accuracy* sebesar 0,9734, dan *validation loss* sebesar 0,0816. Hasil prediksi pada data tes memiliki akurasi 96,8% dan nilai F1-score 0,968.

Kata kunci: Aplikasi Web, *Convolutional Neural Network, Deep Learning*, Docker Container, Google App Engine, Penyakit Malaria, *Transfer Learning*

DESIGN AND DEVELOP APPLICATION TO DETECT MALARIA DISEASE ON THIN BLOOD SMEAR IMAGES BY USING TRANSFER LEARNING ON CONVOLUTIONAL NEURAL NETWORK

Benedictus J. Pradana, Windra Swastika, Oesman H. Kelana
Universitas Ma Chung

Abstract

Malaria is an infectious disease caused by Plasmodium parasite. In 2019 there are 229 million Malaria cases with 400.900 deaths. In 2020 the number of cases increased to 241 million of cases with 627.000 deaths. Malaria diagnosis is done by sampling the patient blood cell and smear it with giemsa stain then observe the sample under microscope. This process is done manually by medical personnel, if the diagnosis isn't done correctly it can lead to misdiagnosis.

Deep learning approach can be used to help Malaria diagnosis by applying classification on blood smear images. This research use Transfer Learning method on Convolutional Neural Network architecture to fasten the process of model development and to obtain model with high accuracy. The model that is used for Transfer Learning is EfficientNetB0.

The model that has been developed is embedded on python-based web application, containerize using docker container, and deploy it on Google App Engine so it can be used by medical personnel to diagnose patient with Malaria. The model training result in 0,9664 training accuracy, 0,0937 training loss, 0,9734 validation accuracy, and 0,0816 validation loss. The evaluation using test data result in 96,8% accuracy and 0,968 F1-score.

Keywords: Convolutional Neural Network, Deep Learning, Docker Container, Google App Engine, Malaria, Transfer Learning, Web Application

KATA PENGANTAR

Puji syukur kepada Tuhan Yang Maha Esa atas karunia-Nya sehingga penulis dapat melaksanakan proses Tugas Akhir dengan judul "Rancang Bangun Aplikasi Untuk Mendeteksi Penyakit Malaria Pada Citra Apusan Darah Tipis Dengan Menerapkan Transfer Learning Pada Convolutional Neural Network" dengan baik. Pada kesempatan ini penulis ingin mengucapkan terima kasih kepada semua pihak yang sudah membantu penulis dalam proses pembuatan tugas akhir ini. Ucapan terima kasih ini juga saya ucapkan kepada:

1. Ibu Dr. Kestrilia Rega Prilianti, M.Si., Selaku Dekan Fakultas Sains dan Teknologi Universitas Ma Chung
2. Bapak Hendry Setiawan, ST., M. Kom., Selaku Kepala Program Studi Teknik Informatika Universitas Ma Chung
3. Bapak Windra Swastika, S.Kom., MT., Ph.D. Selaku dosen pembimbing 1 dalam pengerjaan proyek tugas akhir,
4. Bapak Ir.Oesman Hendra Kelana, M.Cs. Selaku dosen pembimbing 2 dalam pengerjaan proyek tugas akhir,
5. Orang tua terkasih yang selalu mendukung penulis dan memberikan semangat untuk menyelesaikan tugas akhir ini,
6. Dan teman-teman yang sudah membantu dalam pengerjaan tugas akhir ini. Terutama Guntur, Yusano, William juga teman-teman HMI dan KRPA.

Akhir kata, semoga skripsi ini dapat bermanfaat.

Malang, 28 Juli 2022

Benedictus Jullian Pradana
311810007

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 (WHO, 2020). Kasus Malaria mengingkat menjadi 241 juta kasus dan korban kematian 627.000 jiwa (WHO, 2021). Golongan yang paling rentan terjangkit Malaria adalah anak yang berusia di bawah lima tahun dengan persentase 67% dari total kematian (274.000). Sebaran Malaria pada tahun 2019 masih berpusat pada daerah Afrika dengan persentase kasus 94% dari total kasus (WHO, 2020). Indonesia menempati peringkat kedua untuk jumlah kasus Malaria tertinggi di Asia Tenggara. Kasus Malaria di Indonesia sempat menurun pada tahun 2010–2014 namun menjadi stagnan pada tahun 2014-2019 (WHO, 2020).

Berdasarkan data Kemenkes pada tahun 2019 terdapat 250.644 kasus Malaria, mayoritas kasus berada di Papua dengan persentase 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 cara mencegah Malaria yaitu *vector control, preventive chemotherapies,* dan vaksinasi malaria. *Vector control* adalah upaya pencegahan Malaria dengan menggunakan *insecticide-treated* nets (ITNs) atau sejenis jaring yang dibuat secara khusus untuk 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 penggunaan vaksin RTS, S/AS01 secara luas untuk mencegah Malaria terutama pada anak-anak (WHO, 2022).

Malaria didiagnosis dengan cara pemeriksaan fisik dan pemeriksaan darah. Pemeriksaan darah untuk mendiagnosis Malaria meliputi tes diagnostik cepat

Malaria (RDT Malaria) dan pemeriksaan darah penderita di bawah mikroskop dengan metode *blood smear*. Hal ini bertujuan untuk mendeteksi parasit penyebab Malaria yang terdapat pada sel darah (Alodokter, 2020). Diagnosis ini dilakukan secara manual oleh dokter dan jika pengalaman dokter yang mendiagnosis kurang mumpuni dapat mengakibatkan kesalahan diagnosis.

Diagnosis juga menjadi lebih sulit pada negara di mana Malaria sudah tidak menjadi endemi. Penyedia layanan kesehatan bisa saja kurang cermat dalam mengamati hasil pemeriksaan darah. Penyedia layanan kesehatan juga bisa mengesampingkan kemungkinan terjangkitnya Malaria pada pasien dan tidak melakukan diagnostik lanjutan. Kesalahan diagnosis dapat diminimalisir jika petugas kesehatan memiliki sistem yang dapat membantu mereka dalam melakukan diagnosis. Salah satu sistem yang dapat digunakan untuk membantu petugas kesehatan dalam diagnosis adalah *Machine Learning.*

Machine Learning (ML) adalah bagian dari kecerdasan buatan di mana komputer dapat memecahkan masalah atau menemukan suatu pola dari data tanpa harus diprogram secara eksplisit (IBM, 2020). *Machine Learning* dapat digunakan untuk mengklasifikasi sekumpulan citra untuk mendeteksi kelas dari citra tersebut. Dalam kasus Malaria ML dapat digunakan untuk mendeteksi sampel darah yang didapat dari pasien terinfeksi oleh parasite *Plasmodium* atau tidak.

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. *Transfer Learning* digunakan untuk mempercepat proses pembuatan solusi dari permasalahan yang memanfaatkan ML. Sebelumnya terdapat penelitian untuk mendeteksi darah yang terinfeksi Malaria dengan memanfaatkan Transfer Learning dan model yang digunakan adalah 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*. Delgado-Ortet, et al (2020) dalam penelitianya mengklasifikasi penyakit Malaria dengan menggunakan *custom* CNN dan memperoleh akurasi 93,72% pada data validasi. Masud, et al. (2020) dalam penelitianya mengklasifikasi penyakit Malaria menggunakan *custom* CNN waktu

training 97 menit menghasilkan akurasi 97% pada data validasi dan kemudian model ditanamkan pada aplikasi android. Montalbo & Alon (2021) pada penelitianya melakukan klasifikasi pada penyakit Malaria dengan menerapkan *Transfer Learning* menggunakan EfficientNetB0 dan memperoleh akurasi 94,7% pada data validasi.

Pada penelitian sebelumnya peneliti telah melakukan uji coba untuk membandingkan tiga arsitektur *Transfer Learning.* Arsitektur yang dibandingkan adalah ResNet50V2, InceptionV3 dan EfficientNetB0 (Pradana, 2021). Hasil dari penelitian tersebut diperoleh arsitektur terbaik yaitu EfficientNetB0. Namun, pada penelitian sebelumnya, prediksi dilakukan pada data validasi dan hasilnya masih berupa model, sehingga belum dapat digunakan oleh *end user.* Pada penelitian ini akan dilakukan rancang bangun aplikasi dengan menggunakan model berdasarkan penelitian sebelumnya sehingga aplikasi dapat digunakan oleh *end user*. *End user* yang ditargetkan adalah dokter spesialis penyakit dalam dan ahli patologi.

1.2 Identifikasi Masalah

Banyak kasus malaria masih terjadi di Dunia dan di Indonesia, proses diagnosis Malaria yang dilakukan dengan cara manual memiliki kemungkinan untuk terjadinya kesalahan diagnosis. Pada penelitian sebelumnya peneliti telah mengembangkan model *Transfer Learning* untuk mendeteksi parasit Malaria namun model tersebut masih belum bisa digunakan langsung oleh *end-user* (dokter & petugas kesehatan lain) dan prediksi pada model masih menggunakan data validasi.

1.3 Batasan Masalah

Batasan masalah dari pengerjaan penelitian ini adalah sebagai berikut.

1. Penelitian dilakukan dengan dataset citra sel darah yang didapat dari platform Kaggle. Jumlah citra yang digunakan adalah 27560 citra.
2. Dataset yang digunakan merupakan citra hasil pengamatan mikroskop berupa sel darah yang sehat dan sel darah yang terinfeksi Malaria
3. Model dibuat menggunakan platform Tensorflow
4. Model yang digunakan untuk Transfer Learning adalah EfficientNetB0

5. Aplikasi yang dibuat berbasis web
6. Web servis yang digunakan adalah Google Cloud Platform (GCP)

1.4 Rumusan Masalah

Berdasarkan identifikasi masalah yang telah dikemukakan di atas maka dapat dirumuskan permasalahan penelitian ini adalah bagaimana membuat model dengan akurasi yang tidak bias (prediksi model dilakukan pada data tes atau data yang belum pernah dilihat oleh model), dan bagaimana *end user* dapat menggunakan model yang telah dibuat untuk mendeteksi Malaria.

1.5 Tujuan

Tujuan dari pengerjaan penelitian ini adalah untuk membuat model CNN dengan akurasi yang tinggi dalam mendeteksi Malaria dan aplikasi yang dapat digunakan untuk mendeteksi penyakit Malaria pada citra apusan darah tipis.

1.6 Manfaat

Manfaat dari pengerjaan penelitian 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 Tugas Akhir.
2. Bagi Mahasiswa dapat menerapkan ilmu yang telah diperoleh selama belajar di Universitas Ma Chung dan memberikan bekal pengalaman untuk masuk ke dunia kerja

1.7 Luaran Penelitian

Luaran dalam penelitian ini adalah aplikasi untuk mendeteksi penyakit Malaria serta dilanjutkan dengan publikasi ilmiah ke jurnal

1.8 Sistematika Penulisan

Sistematika penulisan ini adalah sebagai berikut:

Bab I: Pendahuluan

Bab pendahuluan terdiri dari latar belakang, identifikasi masalah, batasan masalah, rumusan masalah, tujuan, luaran, manfaat, dan sistematika penulisan.

Bab II: Tinjauan Pustaka

Bab tinjauan pustaka terdiri dari penguraian teori tentang Malaria, *Artificial Intelligence, Machine Learning, Deep learning, Convolution Neural Network, Transfer Learning,* Google Colaboratory, EfficientNetB0, Python, *Library Python,* Google Cloud Platform, Docker, *Application Testing,* dan Penelitian terdahulu.

Bab III: Analisis dan Perancangan Sistem

Bab analisis dan perancangan sistem terdiri dari penguraian detail mengenai analisis kebutuhan pengguna dan peneliti, pengumpulan data, desain dan perancangan sistem, dan metode yang akan digunakan dalam penelitian.

Bab IV: Hasil dan Pembahasan

Bab hasil dan pembahasan berisi penguraian dari hasil percobaan *training* model menggunakan SGD, Adam, dan RMSprop, perbandingan hasil percobaan, aplikasi web, dan pengujian aplikasi.

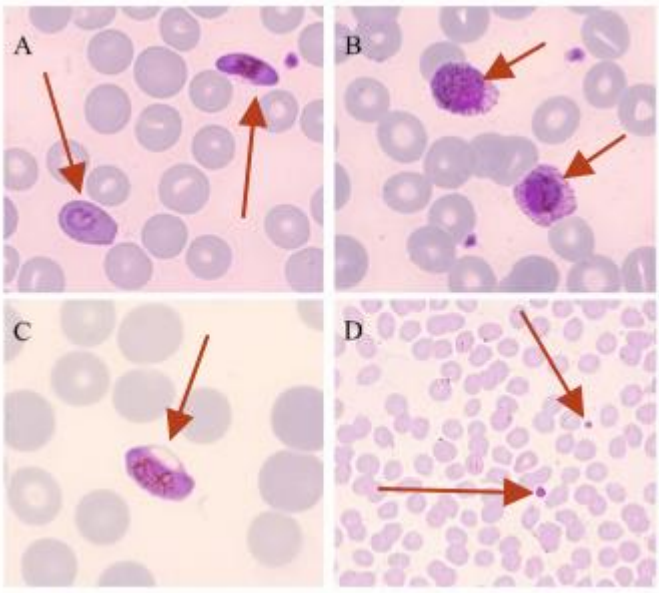
Bab V: Kesimpulan dan Saran

Bab kesimpulan dan saran berisi kesimpulan dari penelitian yang telah dilakukan dan saran yang diberikan agar model dan aplikasi yang dibuat bisa lebih baik dalam mendeteksi penyakit Malaria.

Bab II
Tinjauan Pustaka

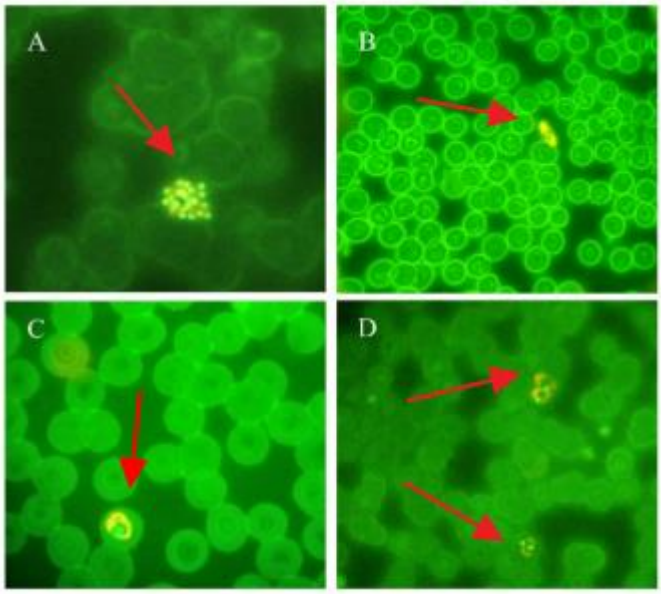
2.1 Malaria

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 menyebabkan Malaria pada manusia, dan dua di antaranya memiliki ancaman paling mematikan yaitu *P.falciparum* dan *P.vivax* (WHO, 2020). Malaria dapat didiagnosis dengan cara hasil anamnesis, pemeriksaan fisik, dan pemeriksaan laboratorium. Untuk hasil yang lebih akurat, diagnosis Malaria dilakukan dengan pemeriksaan darah menggunakan mikroskop (pemeriksaan darah tepi atau *blood smear)* ataupun dengan cara tes antigen (*rapid diagnostic test)* (Evani, 2020). Pengamatan menggunakan mikroskop masih menjadi teknik yang sering digunakan untuk mendeteksi Malaria dan merupakan metode *gold-standard* (Calderaro, et al., 2021) *Blood smear* dilakukan dengan meneteskan *Giemsa* (lihat gambar 2.1)



Gambar 2.1 *Blood smear* menggunakan *Giemsa* (Calderaro, et al., 2021)

Pada gambar 2.1 parasit Plasmodium terlihat berwarna ungu setelah terkena cairan *Giemsa* (perhatikan anak panah). Selain menggunakan *Giemsa*, *blood smear* juga dapat dilakukan dengan meneteskan *acridin orange* (lihat gambar 2.2)



Gambar 2. 2 *Blood smear* menggunakan *acridin orange* (Calderaro, et al., 2021)

Pada gambar 2.2 dapat dilihat sampel darah yang telah diteteskan *acridin orange*. Parasit Plasmodium terlihat berwarna kuning terang (perhatikan anak panah) dan sampel darah menjadi berwarna hijau. Terdapat dua teknik *blood smear, thick blood smear* dan *thin blood smear*. Pada sampel darah *thick blood smear* digunakan untuk meningkatkan sensitivitas jika level parasit pada sampel rendah. *Thin blood smear* sensitivitasnya lebih rendah daripada *thick blood smear,* meski begitu *thin blood smear* lebih sering digunakan karena parasit lebih mudah terlihat dan dihitung.

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 (Na'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 parasit 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, 2020).

2.2 Artificial Intelligence

Artificial intelligence (AI) atau kecerdasan buatan adalah program komputer yang diciptakan manusia untuk menyimulasikan 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* (Takdirillah, 2020). Salah satu jenis kecerdasan buatan yang telah umum digunakan adalah ML. Machine Learning merupakan sub set dari kecerdasan buatan di mana 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 ML juga terdapat sub set *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 *Recurrent Neural Networks (RNN)*

2.3 Machine Learning

Machine Learning (ML) merupakan bagian dari kecerdasan buatan. Secara singkat *Machine Learning* merupakan algoritma untuk menyelesaikan permasalahan berdasarkan data dengan sendirinya tanpa di program secara eksplisit (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 algoritma nya 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 merupakan metode yang digunakan ketika data set 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 ML sebelumnya karena model tidak di *training* berdasarkan label dari dataset melainkan terdapat *agent* yang 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 *Reinforcement Learning* adalah *Q-Learning.*

2.3.1 K Nearest Neighbour

K-Nearest Neighbours (KNN) merupakan klasifikasi non para metrik yang pertama kali diperkenalkan pada tahun 1951 (Fix & Hodges, 1989) dan kemudian dikembangkan Thomas Cover pada tahun 1967 (Cover & Hart, 1967)yang digunakan untuk permasalahan klasifikasi dan *regression*. KNN bekerja dengan cara memprediksi *class* yang tepat dari test data dengan menghitung jarak *Euclidean* 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 Euclidean *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 regression merupakan algoritma yang digunakan untuk predictive analysis. Secara umum regression digunakan untuk mengamati dua hal, pertama sekumpulan variabel prediktor (Independen variabel) yang berkorelasi sehingga dapat memprediksi variabel hasil (dependen variabel). Kedua, mencari variabel 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 dependen variabel dengan satu atau banyak independen variabel (Statistic Solution, 2022)

2.4 Deep Learning

Deep Learning merupakan sub set dari *Machine Learning* yang menggunakan *Artificial Neural Networks (ANN)* hingga membentuk banyak lapisan

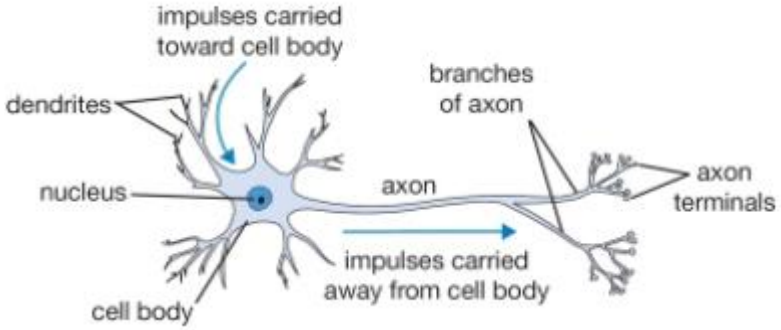
Neural Networks atau *Deep Neural Networks. Deep Learning* terinspirasi dari cara kerja sel neuron pada otak yang saling terkoneksi satu sama lain untuk memproses informasi. *Deep Learning* memungkinkan model komputasi yang terdiri dari *multi-layer processing* untuk mempelajari representasi dari data dengan *multiple-level of abstraction* (LeCun, et al., 2015). LeCun juga menjelaskan bahwa *Deep Learning* menggunakan *Backpropagation* untuk menemukan pola pada struktur data dan cara 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 *Recurrent Neural Networks (RNN).*

CNN atau ConvNet merupakan arsitektur untuk *deep learning* yang tidak perlu secara manual mengekstrak fitur 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.*

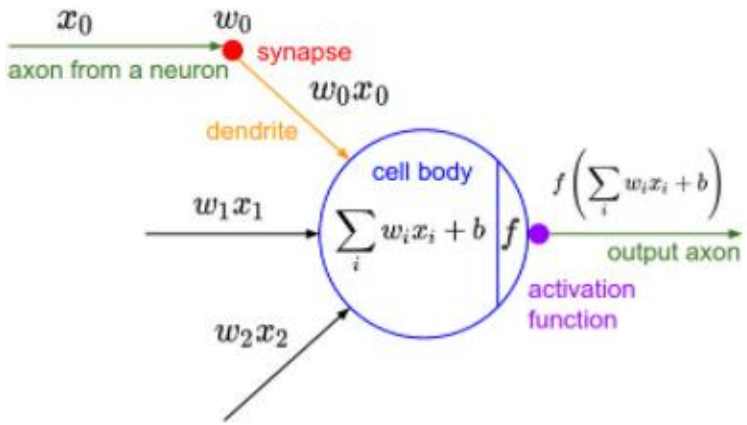
RNN merupakan perkembangan dari ANN yang memungkinkan output dari layer sebelumnya digunakan sebagai *input* ketika memiliki *hidden state*. RNN cocok digunakan untuk menyelesaikan permasalahan time series.

2.5 Artificial Neural Network

Artificial Neural Network (ANN) disebut 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, et al., 2019). NN pertama kali diperkenalkan pada tahun 1950-an berupa dua layer *perceptron* (Rosenblatt, 1958), Kemudian dikembangkan lebih lanjut dengan dikenalnya algoritma *back propagation* (Werbos, 1990) untuk membuat *training multi-layer NN* yang lebih efisien (Hinton, et al., 2006).



Gambar 2. 3 gambar single cell neuron (Stanford CS231n, 2020)



Gambar 2. 4 Representasi matematis untuk neuron (Stanford CS231n, 2020)

Pada gambar 2.4 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.

$$z = \sum_{i=1}^{n} x_i w_{ij} + b_j \tag{2-1}$$

Di mana:

$x = input$, adalah nilai dari *features*, secara umum berupa sekumpulan bilangan riil dalam array

$w =$ beban (*weight*), merepresentasikan koefisien skalar yang bertugas untuk mengukur seberapa penting nilai dari setiap input.

$z = transfer\ function$, bertugas mengombinasikan 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, et al., 1998).*Forward propagation* merupakan proses yang melibatkan perkalian *feature value* dengan *weights*, kemudian dikombinasikan 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* mengubah 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 scale* 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.*

$$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases} \tag{2-4}$$

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.*

$$f(x) = \begin{cases} 0.01x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases} \quad (2\text{-}5)$$

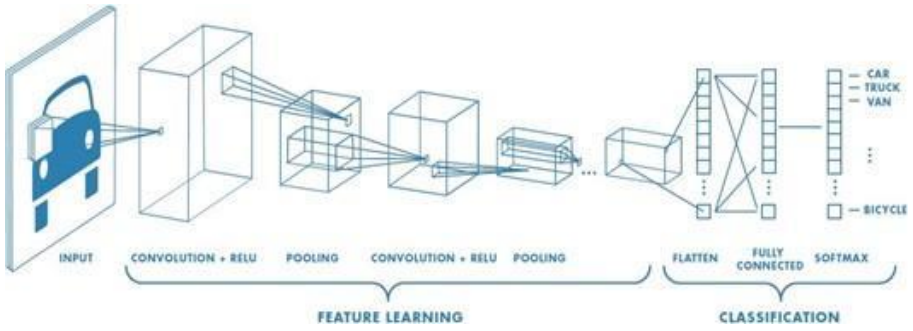
Backward Propagation merupakan proses yang dilakukan untuk menemukan nilai optimal parameter pada model dengan cara memperbaharui parameter berulang kali dengan melakukan perhitungan turunan parsial nilai gradien dari *loss function* dengan nilai *weights*. Fungsi optimasi (*Optimization Function)* digunakan 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).

$$\frac{\partial L}{\partial w} = \frac{\partial z}{\partial w} \times \frac{\partial a}{\partial z} \times \frac{\partial L}{\partial a} \quad (2\text{-}6)$$

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

Convolutional Neural Network (CNN) atau *ConvNet* merupakan arsitektur untuk *deep learning* yang tidak perlu secara manual mengekstrak fitur 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. CNN cukup populer dalam bidang *Computer Vision*, baik digunakan untuk mengenali citra maupun *Object Detection.* Arsitektur CNN dapat dilihat pada gambar 2.5.



Gambar 2. 5 Arsitektur CNN (MathWorks, 2022)

Secara umum CNN tersusun dari *convolution layers, activation functions, pooling layers,* dan *flatten layer* (Wang, et al., 2021). *Input layer* yang berada 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.

Convolutional 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, et al., 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 vektor input yang sebelumnya tiga dimensi menjadi satu dimensi yang kemudian disalurkan *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 probabilitas. Selama satu dekade terakhir terpadat tiga arsitektur cnn populer yang menjadi pioneer untuk arsitektur arsitektur baru berikutnya yaitu AlexNet, VGG, dan ResNet.

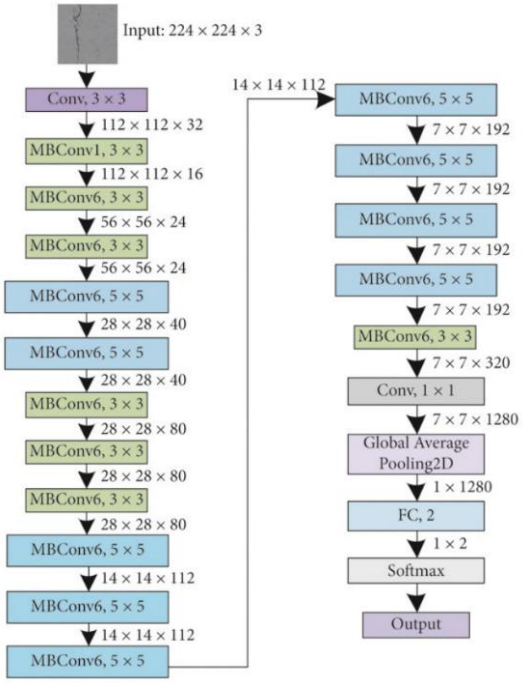
2.7. Transfer Learning

Transfer Learning merupakan teknik ML yang digunakan untuk meningkatkan akurasi dengan menggunakan model yang telah di *training* sebelumnya sebagai *starting point* untuk permasalahan yang berbeda namun masih relevan (Brownlee, 2017).*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). *Early layer* (layer awal) pada *deep learning* khususnya CNN berfungsi untuk mengidentifikasi bentuk sederhana, *later layer* berfungsi untuk mengidentifikasi pola visual yang kompleks, dan *layer* terakhir melakukan prediksi (*output layer).* Sebagian besar *layer* (*layer awal)* pada *pre-trained* model sangat berguna untuk dataset yang berbeda karena kebanyakan permasalahan dalam *computer vision* khususnya klasifikasi gambar adalah melibatkan pencarian pola sederhana (khusunya bentuk). *Transfer Learning* bekerja dengan menggunakan semua *layer* pada *pretrained-model* dan mengganti *layer* terakhir untuk prediksi sesuai dengan dataset yang digunakan.

2.8 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, 2019). ImageNet merupakan dataset yang kompleks karena mengandung 14 juta citra dengan 1000 kelas yang berbeda. Pada gambar 2.6 adalah arsitektur dari efficientnetb0.



Gambar 2. 6 Arsitektur model EfficientNetB0 (Su & Wang, 2020)

Blok MBConv pada arsitektur di atas dapat disebut juga dengan *inverted residual block. Inverted residual block* menggunakan pendekatan *narrow, wide, narrow* pada arsitekturnya. MBConv memiliki tiga *convolutional layer*, layer pertama memiliki filter 1x1, layer kedua memiliki filter 3x3, dan layer ketiga memiliki filter 1x1. MBConv digunakan untuk mengurangi parameter yang diperlukan untuk *training*. Karena parameter training yang relative sedikit, EfficientNet disebut "Efficient".

2.9 Google Colaboratory

Google Collaboratory disingkat "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.

2.10 Python

Python merupakan bahasa pemrograman tingkat tinggi yang dapat digunakan untuk banyak kasus(*general purpose*). Python diciptakan oleh Guido van Rosum dan pertama kali dirilis pada 20 februari 1991. Nama python terinspirasi dari acara favorit Guido yang tayang di BBC dengan nama Monty Python's Flying Circus (PythonInstitute, 2022). Pada tahun 2021 berdasarkan stack overflow survey, python digunakan oleh 48.24% developer di seluruh dunia dari 83,052 responden (Stack Overflow, 2021).

2.11 Library Python

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

2.11.1 Tensorflow

Tensorflow merupakan open-source library yang digunakan untuk ML dan AI. 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 ML maupun untuk keperluan riset. Saat ini tensorflow memiliki versi yang terbaru yaitu Tensorflow 2.7 .

2.11.2 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

2.11.3 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 ML maupun analisa data. Matplotlib dapat menyajikan *insight* dari data yang telah dianalisa.

2.11.4 Streamlit

Streamlit merupakan *open-source python framework* yang memungkinkan *data scientist* untuk membuat *dashboard* interaktif dan aplikasi web ML tanpa harus memiliki pengalaman *front-end* (Li, 2021). Streamlit memungkinkan untuk membuat aplikasi dengan cara yang sama untuk menulis program python dan membuat website interaktif untuk *deploy* ML model dengan mudah (Rajan, 2021). Sebelum streamlit dapat digunakan, pastikan *python interpreter* sudah ter-*install* terlebih dahulu lalu gunakan perintah `pip install streamlit` pada terminal dan setelah terinstal gunakan perintah `import streamlit as st`. Streamlit memudahkan pembuatan tampilan *web app* dengan menyediakan *widget* yang mudah untuk diatur. Streamlit bekerja dengan cara menjalankan *script* dari atas ke bawah seperti halnya ketika program python dijalankan. Setiap kali terjadi perubahan pada *script*, streamlit akan menunjukan notifikasi jika aplikasi harus di *run* ulang atau tidak. Setiap kali user berinteraksi dengan aplikasi, streamlit akan menjalankan ulang script dari awal. Streamlit juga memungkinkan penggunaan *cache* untuk membuat pekerjaan berat seperti memuat dataset yang berukuran besar menjadi lebih cepat. Keunggulan dari streamlit adalah mudah untuk dipelajari karena cukup sederhana dan mudah diimplementasikan hanya dengan beberapa baris kode saja. Streamlit juga dapat membuat proses A/B test menjadi lebih cepat dan yang terakhir, streamlit ringan untuk digunakan dan gratis.

2.12 Google Cloud Platform

Google Cloud Platform (GCP) merupakan layanan *cloud computing service* yang ditawarkan oleh Google. GCP menawarkan layanan *computing*, penyimpanan dan pengembangan aplikasi yang dijalankan dengan arsitektur *hardware* yang sama digunakan oleh Google. Google cloud menawarkan layanan seperti *Compute*

Engine, Storage, Networking, Big Data, AI *and* ML *(AI Platform)* dan *Internet of Things (IoT)* (J.Bigelow, 2020). Produk inti dari GCP termasuk:

1. *Google Compute Engine,* merupakan *infrastructure-as-a-service (IaaS)* yang menawarkan user dengan *virtual machine* untuk *workload hosting.*
2. *Google App Engine,* merupakan *platform-as-a-service(PaaS)* yang menawarkan akses ke Google *scalable hosting.*
3. *Google Cloud Storage,* merupakan *cloud storage platform* yang didesain untuk menyimpan *unstructured data set* yang berukuran besar.
4. *Google Container Engine,* merupakan sistem manajemen untuk *Docker Container* yang dapat berjalan pada *Google Public Cloud.*

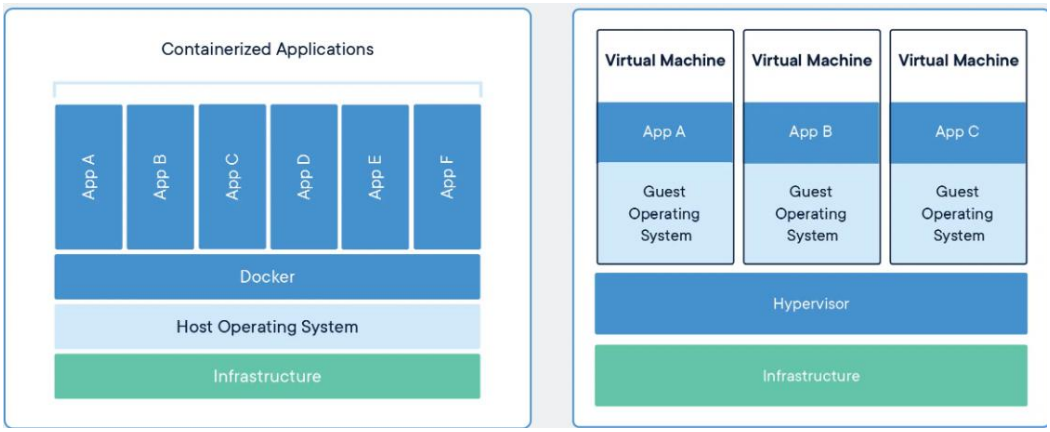
GCP juga menawarkan pengembangan aplikasi dan layanan integrasi seperti Google Cloud Pub/Sub yang digunakan untuk mengatur layanan pesan secara *real-time* dan dapat digunakan untuk saling mengirim pesan antar aplikasi. GCP juga memiliki layanan dengan level yang lebih tinggi yang berhubungan dengan *Big Data* dan ML (J.Bigelow, 2020). Untuk layanan yang berhubungan dengan AI dan ML disebut Google Cloud AI Platform. AI Platform merupakan layanan GCP yang secara kusus ditargetkan untuk *building*, *deploying*, dan mengelola model ML pada ekosistem *cloud.* AI Platform didesain untuk mempermudah *data scientist* dan *data engineers* dalam melakukan ML *workflows* dan akses yang mudah untuk menggunakan AI yang dikembangkan oleh Google (Green, 2020).

GCP juga memiliki layanan Google App Engine (GAE) yang menyediakan pengembang aplikasi web dan industri untuk membuat *scalable* aplikasi web dan *mobile back end* pada platform *serverless* (Lutkevich, 2022). Aplikasi yang berjalan pada GAE dijalankan menggunakan infrastruktur yang dibuat oleh Google dan dapat dijalankan pada banyak *server* untuk mengelola lalu lintas pada aplikasi web. Untuk menjalankan aplikasi pada GAE, aplikasi harus dibuat dengan bahasa pemrograman go, python, node js, java, php, ruby, atau *custom runtime* untuk bahasa pemrograman lain.

2.13 Docker

Docker merupakan open platform yang berfungsi untuk membuat, deploy, dan menjalankan aplikasi. Docker memungkinkan pengembang aplikasi untuk memisahkan aplikasi dari infrastruktur yang dibutuhkan sehingga aplikasi dapat

segera publikasikan (Docker Inc, 2022). Docker menyediakan yang memungkinkan aplikasi dijalankan dalam *isolated environment* yang disebut *container*. Cara kerja *container* mirip dengan *virtual machine* (VM) namun berbeda fungsi, jika *container* membuat virtualisasi dari *Operating System (OS)* sedangkan VM membuat virtualisasi perangkat keras sehingga dapat menjalankan OS yang berbeda-beda (Docker Inc, 2022).



Gambar 2.7 Perbedaan *container* (kiri) dan VM (kanan) (Docker Inc, 2022)

Pada gambar 2.7 menunjukan perbedaan *container* dan VM. *Container* merupakan abstraksi pada lapisan app yang memuat *source code* dari aplikasi dan *dependencies* secara bersamaan. Multi *container* dapat dijalankan pada mesin yang sama.

2.14 Application Testing

Pengujian aplikasi atau biasa disebut *software testing/application testing* merupakan proses pemeriksaan yang dilakukan untuk mengetahui performa dan kualitas dari aplikasi yang sedang diuji. Secara spesifik *software testing* dilakukan untuk menemukan bug yang terdapat pada aplikasi (Dhuha, 2018). *Software testing* juga dapat digunakan untuk mendapatkan pandangan yang objektif dan independen terhadap aplikasi untuk memahami tingkat resiko pada saat *deployment.* Dengan kata lain *software testing* dilakukan untuk memastikan bahwa suatu aplikasi telah memenuhi syarat dan kebutuhan teknis yang mendasari pengembangan aplikasi dan aplikasi bekerja sesuai ekspektasi. Secara umum terdapat enam tahapan dalam melakukan *software testing* yaitu:

1. *Unit/Component Testing, unit testing* merupakan proses di mana setiap bagian basic dari *source code* seperti fungsi dan prosedur diuji. *Component testing* dilakukan setiap kali terdapat kode unit baru seperti fungsi yang baru saja dibuat.
2. *Integration Testing,* tahap ini dilakukan untuk menguji interaksi antara setiap unit.
3. *System Testing*, tahap ini dilakukan setelah aplikasi selesai dibuat, hal ini dilakukan untuk memastikan bahwa aplikasi dapat berfungsi sesuai dengan yang diharapkan
4. *Acceptance Testing,* tahap ini hamper sama dengan *system testing* hanya saja dilakukan oleh *end-user* terpilih. Hal ini dilakukan untuk memastikan fungsi dari aplikasi dapat diterima oleh *end-user atau tidak*
5. *Regression Testing*, merupakan *testing* yang dilakukan Ketika aplikasi mendapatkan *update* dan untuk memastikan aplikasi versi *update* telah menyelesaikan permasalahan yang terdapat pada versi sebelumnya.

Software testing terbagi menjadi tiga metode yaitu *white box, black box*, dan *gray box* (Hamilton, 2022). *White box* merupakan proses pengujian yang dilakukan untuk pengecekan struktur internal dari aplikasi, hal ini berjutuan untuk mengetahui bahwa program dari aplikasi berjalan dengan baik dan untuk memastikan bahwa operasi internal seperti fungsi, perulangan, dan kemanan berfungsi dengan baik (Dhuha, 2018). *White box testing* cocok digunakan untuk aplikasi yang berukuran kecil karena diperlukan waktu dan tenaga yang dukup besar. *Black box testing* dilakukan untuk menguji aplikasi besar, pengujian ini dilakukan hanya dengan mengamati proses berjalanya aplikasi dari input hingga output dan pengujian ini hanya dilakukan pada *interface* saja tanpa mengecek *source code* dari aplikasi. Keunggulan dari *black box* adalah dapat menemukan bug dengan cepat dan secara efisien namun memiliki kekurangan karena tidak dapat dipastikan bahwa fungsi dari masing masing unit benar benar lulus uji.

Software testing dapat disesuaikan dengan jenis aplikasi yang dibuat, *software testing* dibagi menjadi tiga, *Web Application Testing, Mobile Application Testing,* dan *Desktop Application Testing* (Hamilton, 2022). Pada penelitian ini akan dilakukan *Mobile Application Testing,* karena aplikasi yang dikembangkan

berbasis mobile khususnya android. Dilansir dari softwaretestinghelp.com (softwaretestinghelp, 2022).

Jenis-jenis *mobile app testing* antara lain:

1. *Usability testing*, dilakukan untuk memastikan bahwa aplikasi mudah digunakan dan menyediakan *user experience* yang memuaskan terhadap pengguna
2. *Compatibility testing*, menguji aplikasi dengan menggunakan perangkat mobile yang berbeda beserta versi OS-nya
3. *Interface testing*, menguji tampilan seperti menu, tombol, pengaturan dan navigasi
4. *Service testing*, menguji penggunaan aplikasi saat offline maupun online
5. *Low-level resource testing*, menguji penggunaan memori saat aplikasi dijalankan, dan *database* lokal
6. *Performance testing*, menguji performa aplikasi ketika mengganti koneksi dari 2G,3G,4G,5G maupun WiFi, penggunaan baterai dll.
7. *Operational testing*, menguji *backups and recovery* jika daya baterai tiba-tiba habis, *data loss* saat *update* aplikasi dari appstore
8. *Installation tests*, validasi aplikasi dengan menginstal / *uninstall*
9. *Security testing*, menguji aplikasi bahwa sistem melindungi data

2.15 Penelitian terdahulu

Beberapa penelitian terdahulu yang memiliki hubungan dengan *Transfer Learning* dan pendeteksian Malaria akan menjadi referensi pada penelitian yang akan dilakukan.

Pradana (2021) melakukan penelitian untuk membandingkan tiga jenis *pretrained* model untuk mendeteksi Malaria. *Pretrained* model yang digunakan antara lain InceptionV3, EfficientNetB0, dan Resnet50V2. Penelitian dilakukan untuk mengetahui performa *pretrained* model dalam melakukan klasifikasi citra apusan darah tipis parasit Malaria. Dari penelitian yang dilakukan terdapat dua model yang memiliki nilai akurasi prediksi tertinggi yaitu model ResNet50v2 pada eksperimen kelima dan EfficientNetB0 pada eksperimen ketiga, masing-masing model memiliki nilai akurasi prediksi sebesar 0,975. Berikut adalah hasil percobaan dari penelitian tersebut

Tabel 2. 1 Hasil percobaan Pradana (2021)

Model	Percobaan	*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

Reddy & Juliet (2019) melakukan penelitian untuk mendeteksi Malaria pada citra apusan darah tipis. Penelitian dilakukan dengan menerapkan *Transfer Learning*. Mesin yang digunakan untuk *training* adalah computer dengan 8 GB RAM dan tanpa GPU. *Pretrained* model yang digunakan pada penelitian tersebut adalah ResNet50. Dataset yang digunakan berisi citra yang berjumlah 27558 dan memiliki dua kelas *uninfected* dan *parasitized* dengan rasio 50:50. Sebelum dilakukan *training* pada model, dilakukan *split* pada dataset dengan rasio 70:15:15 untuk *training set, validation set,* dan *test set.* Setiap citra pada dataset diubah menjadi 224x224 untuk menyesuaikan dengan ukuran input pada ResNet50.

Parameter yang digunakan untuk *training* adalah *batch size* = 100, *epoch* = 10, dan *optimizer* yang igunakan adalah SGD. Hasil yang didapat adalah *training accuracy = 95,91%*, *validation accuracy = 95,4%*, *training loss* = 0,1134, dan *validation loss* = 0,1301

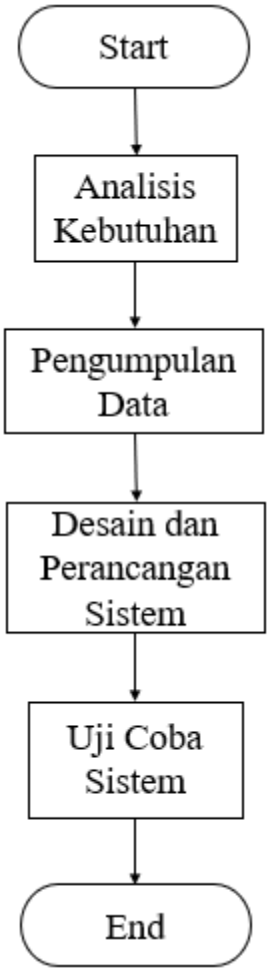
Delgado-Ortet, et al., (2020) melakukan penelitian untuk mendeteksi Malaria pada citra apusan darah tipis. Penelitian dilakukan dengan menerapkan *Segmentation Neural Network* dan *custom* CNN (membuat arsitektur sendiri). Dataset yang digunakan berisi citra yang berjumlah 27558 dan memiliki dua kelas *uninfected* dan *parasitized* dengan rasio 50:50. Peneliti juga menggunakan dataset tambahan. Dataset tambahan berupa 517 citra apusan darah tipis yang terbagi menjadi dua, 186 citra untuk *uninfected* dan 331 citra untuk *parasitized*. Sebelum dilakukan training pada SNN, diterapkan image segmentation pada citra terlebih dahulu. Hasil dari SNN adalah *global accuracy* 93,72% dan *precision* 87,04%. Hasil dari CNN adalah akurasi 95% pada data validasi dan 75% pada data tes

Masud, et al,. (2020) melakukan penelitian untuk mendeteksi Malaria pada citra apusan darah tipis. Penelitian dilakukan dengan menerapkan *custom* CNN. Dataset yang digunakan berisi citra yang berjumlah 27558 dan memiliki dua kelas *uninfected* dan *parasitized* dengan rasio 50:50. Dataset kemudian dibagi menjadi dua dengan rasio 80:20 untuk *training* dan *validation*. Citra pada dataset kemudian diubah ukuranya menjadi 224x224. Model yang dibuat kemudian ditanamkan pada aplikasi android dengan dikonversikan terlebih dahulu menggunakan Tensorflow Lite. Peneliti juga menerapkan konsep *cyclical learning rate* (CLR) dan *triangular schedule*. *Learning rate* (lr) dalam CLR akan menurun setiap *epoch* sampai batas yang telah ditentukan, namun dengan dibuatnya *triangular schedule*, lr akan naik hingga nilai maksimum pada setengah siklus kemudian menurun kembali hingga nilai minimum pada siklus penuh. CNN yang dibuat memiliki empat *convolutional block* dan dua *fully connected layer*. Setiap blok terdiri dari *convolution*, *max pooling*, *batch normalization,* dan *dropout layer*. Model dilatih dengan *epoch* sejumlah 50 dan waktu *training* yang dibutuhkan adalah 97 menit. Model hasil penelitian memiliki akurasi sebesar 97,3% pada data validasi dan nilai F1-score 0,97

Montalbo & Alon (2021) melakukan penelitian untuk mendeteksi Malaria pada citra apusan darah tipis. Penelitian dilakukan dengan menerapkan *Transfer Learning*. Dataset yang digunakan berisi citra yang berjumlah 27558 dan memiliki dua kelas *uninfected* dan *parasitized* dengan rasio 50:50. Dataset kemudian dibagi menjadi dua dengan rasio 80:20 untuk *training* dan *validation*. Citra pada dataset kemudian diubah ukuranya menjadi 224x224. *Pretrained* model yang digunakan untuk *Transfer Learning* adalah EfficientNetB0. Paremeter yang diguanakan adalah *epoch* = 10, *optimizer* ADAM, dan *learning rate 1e-4*. Model hasil penelitian memiliki akurasi 94,68% dan 94,61 untuk F1-score.

Bab III
Analisis dan Perancangan Sistem

Penelitian ini dilakukan untuk mengembangkan aplikasi Android untuk dapat mendeteksi penyakit Malaria dengan menerapkan Transfer Learning pada CNN. Proses penelitian ini dibagi menjadi beberapa tahap yang dapat dilihat pada gambar 3.1



Gambar 3. 1 Tahapan proses penelitian

Penelitian ini dimulai dari analisis kebutuhan yaitu mempersiapkan apa saja yang dibutuhkan untuk proses penelitian. Setelah itu dilanjutkan ke tahap pengumpulan data, untuk dataset Malaria yang digunakan dalam penelitian ini sudah tersedia pada platform Kaggle. Setelah dataset diperoleh, dilanjutkan dengan desain dan perancangan sistem. Pada tahap ini juga akan dilakukan pengembangan

untuk tampilan aplikasi untuk uji coba dari Android yang masih berupa MVP (*Most Viable Product)*. Setelah sistem berhasil dibuat, akan dilanjutkan dengan tahapan terakhir yaitu uji coba.

3.1 Analisis Kebutuhan

Perancangan sistem pada penelitian ini diawali dengan analisis terlebih dahulu, sehingga penelitian dapat dilakukan dengan benar dan sesuai target. Analisis yang dilakukan meliputi analisis perangkat yang digunakan dalam penelitian baik dari sisi peneliti dan pengguna.

3.1.1 Kebutuhan Pengguna

Target pengguna pada penelitian ini adalah para dokter yang memiliki spesialisasi dalam mengurus penyakit dalam seperti Malaria. Aplikasi ini digunakan oleh dokter yang ingin mendiagnosis pasien yang diduga terjangkit Malaria. Aplikasi ini dapat memudahkan para dokter dalam proses mendiagnosis dan mengurangi kemungkinan terjadinya salah diagnosis karena faktor *human error*. Aplikasi ini juga mudah untuk digunakan, hanya perlu menekan tombol untuk mengambil gambar yang tersimpan pada *smartphone* lalu tekan tombol prediksi, hal ini tentu saja relatif lebih mudah dilakukan dan memakan tenaga lebih sedikit daripada melakukan diagnosis dengan pengamatan mikroskop yang membutuhkan konsentrasi tinggi.

3.1.2 Kebutuhan Peneliti

Berikut merupakan perangkat keras dan perangkat lunak yang digunakan sebagai penunjang penelitian.

1. Perangkat Keras
   a. *Smartphone* POCO X3 NFC
      i.CPU: Qualcomm Snapdragon 732 Octa-Core
      ii.RAM: 6 GB
      iii.ROM: 64 GB
      iv.Sistem Operasi: Android 10, One UI 2.0

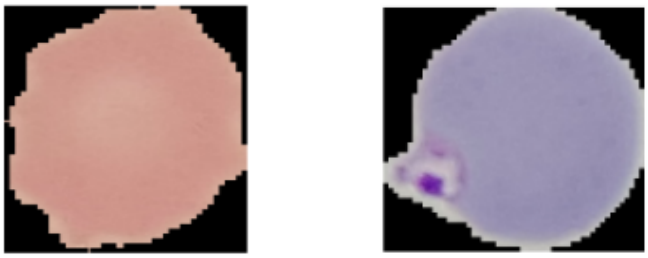
   b. Laptop ASUS X555QG
      i.Prosesor: AMD A12-9700P ~ 2.5GHz
      ii.RAM: 16 GB
      iii.SSD: 512 GB

iv.GPU: AMD Radeon R7
v.Sistem Operasi: Windows 10 Professonal 64 Bit

2. Perangkat Lunak
   a. Visual Studio Code
   b. Python 3
   c. Tensorflow 2.x
   d. Numpy
   e. Pandas
   f. Matplotlib
   g. StreamLit
   h. GCP App Engine

3.2 Pengumpulan Data

Data yang akan digunakan sudah tersedia pada platform Kaggle. Data yang digunakan adalah data set 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 3.2.

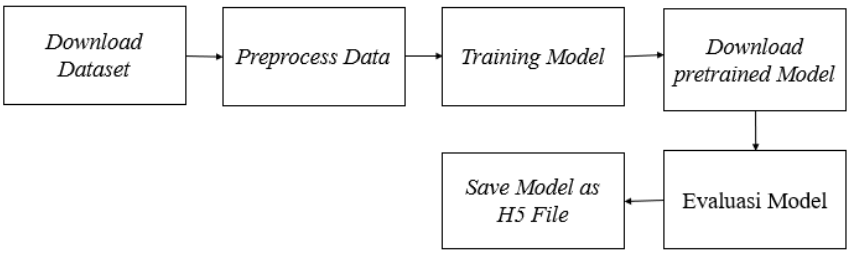


Gambar 3. 2 Sampel dari citra sel yang *uninfected* dan *parasitized*

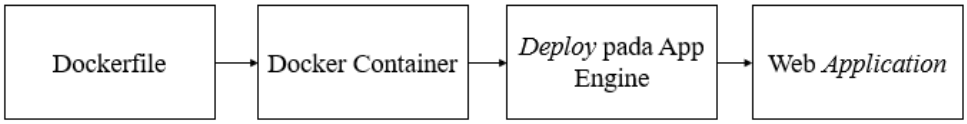
Citra sebelah kiri merupakan citra dari sel yang tidak terinfeksi(*uninfected)* oleh parasit Plasmodium sedangkan citra sebelah kanan merupakan citra dari sel yang terinfeksi oleh Plasmodium (*parasitized)*. Sel yang terinfeksi berubah warnanya yang semula berwarna merah muda menjadi biru pucat disertai bintik hitam. Citra diambil dengan mengamati sel darah yang terdapat parasit malaria menggunakan mikroskop dengan perkiraan perbesaran 100x kemudian citra dibagi menjadi beberapa segmen

3.3 Desain dan Perancangan Sistem

Pada tahap ini akan dilakukan desain dan perancangan sistem dari penelitian. Gambar 3.3 menunjukan tahapan dari *preprocessing* dan pelatihan model. Gambar 3.4 menunjukkan tahapan dari uji coba pada Android.



Gambar 3. 3 Tahap *Preprocessing* dan Pelatihan Model



Gambar 3. 4 *Deployment* pada *App Engine*

Gambar 3. 5 Tahap uji coba pada aplikasi Android

3.3.1 Preprocessing

Pada tahap ini, dimulai dengan mengunduh dataset yang diperlukan dari platform Kaggle. Proses mengunduh dilakukan dengan cara mengunggah file kaggle.json pada runtime colab. File Kaggle.json dapat diunduh pada profil akun Kaggle masing-masing.

Dataset yang telah diunduh akan dibagi menjadi dua bagian yaitu 70% untuk *train*, 15% untuk *validation*, dan 15% untuk *test (*evaluasi). Data *train* akan digunakan saat proses training dan data *validation* digunakan bersamaan pada saat proses training. Setelah itu dilakukan *preprocess data*, data dinormalisasi dan dilakukan penyesuaian ukuran citra untuk input. Kemudian mengunduh *pre-trained* model untuk Transfer Learning dari keras. Kemudian dilakukan model *training*.

3.3.2 Model *Training*

Pada tahap ini akan dilakukan *training* pada model dengan menerapkan *Transfer Learning. Pretrained* model yang digunakan adalah EfficientNetB0. Model EfficientNetB0 dilatih dengan benchmark ImageNet. Berikut adalah *summary* dari model yang akan digunakan.

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_layer (InputLayer)    [(None, 128, 128, 3)]     0

 data_augmentation (Sequenti  (None, 128, 128, 3)      0
 al)

 efficientnetb0 (Functional)  (None, None, None, 1280)  4049571

 global_average_pooling_laye  (None, 1280)             0
 r (GlobalAveragePooling2D)

 output_layer (Dense)        (None, 2)                 2562

=================================================================
Total params: 4,052,133
Trainable params: 2,562
Non-trainable params: 4,049,571
_________________________________________________________________
```

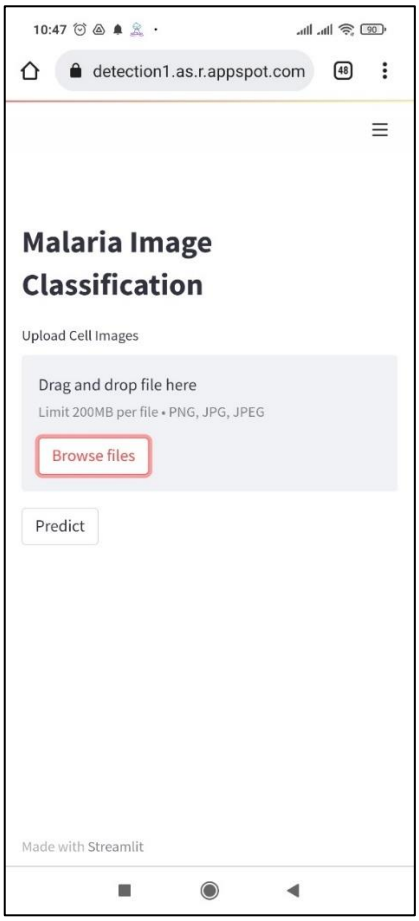
Gambar 3. 6 *Summary* model

Model pada gambar 3.6 memiliki 4.052.126 total parameter. 4.049.571 parameter berasal dari model *pre-trained* EfficientNetB0. Fungsi aktivasi yang digunakan sesuai dengan model dari Transfer Learning yaitu ReLU, dan untuk fungsi aktivasi pada output *layer* yang digunakan adalah *softmax* dengan jumlah neuron 2 unit. Parameter merupakan jumlah weight yang terdapat pada model.

Pada penelitian ini akan dilakukan 18 percobaan dengan variasi pada *optimization function, learning rate,* dan *dropout* untuk mendapatkan model dengan akurasi yang tinggi. *Optimization function* yang akan dicoba adalah Adam, RMSProp, dan SGD. *Learning rate* yang akan dicoba adalah *default learning rate* (1e-3),1e-4, dan *fine tuning* dengan *learning rate* 1e-5. Dari hasil percobaan yang dilakukan kemudian akan dipilih satu model terbaik dan disimpan dalam format hdf5 (*Hierarchical Data Format* atau file dengan ekstensi .h5)

3.3.3 Uji Coba Aplikasi Web

Pada tahap ini akan dilakukan uji coba terhadap model terbaik , uji coba digunakan pada smartphone. Sebelum dilakukan uji coba, model ditanamkan pada aplikasi web dan kemudian aplikasi akan diunggah pada App Engine, sehingga pengguna dapat melakukan prediksi secara online. Tampilan *mockup* antar muka dapat dilihat pada gambar 3.5. Pada tampilan halaman utama (*home)* dari aplikasi terdapat tempat untuk mengunggah gambar, tombol untuk prediksi, dan hasil prediksi. Uji coba akan dilakukan dengan dua cara. Cara pertama adalah dengan menguji fitur yang terdapat pada aplikasi, fitur unggah foto dan fitur prediksi. Cara kedua dilakukan dengan melakukan survei pada sepuluh pengguna *smartphone*.



Gambar 3. 7 *Interface* aplikasi web

3.3.4 *Deployment* Pada Google App Engine

Sebelum aplikasi diuji dilakukan *deployment* terlebih dahulu. Hal yang diperlukan untuk *deployment* adalah, konfigurasi dockerfile yang dibutuhkan untuk membuat *container*

3.4 Pengujian dan Evaluasi

Untuk pengujian akan dilakukan pengujian pada model hasil *training* dan pengujian pada aplikasi.

3.4.1 Pengujian Model

Pengujian yang dilakukan menggunakan *accuracy* dan *loss*. *Train accuracy* 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. 1 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*.

$$Accuracy = \frac{(TP+TN)}{(TP+FP+FN+TN)} \quad (3\text{-}1)$$

$$Loss = \frac{(FP+FN)}{(TP+FP+FN+TN)} \quad (3\text{-}2)$$

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 ML terutama model klasifikasi. F1-score terdiri dari *accuracy*, *precision*, *recall*, dan *F1-score. Accuracy* merukapan presentase prediksi benar dari total prediksi.

$$Accuracy = \frac{correct\ predictions}{total\ predictions} \tag{3-3}$$

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

$$Precision = \frac{\#\ of\ True\ Positives}{\#\ of\ True\ Positives + \#\ of\ False\ Positives} \tag{3-4}$$

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

$$Recall = \frac{\#\ of\ True\ Positives}{\#\ of\ True\ Positives + \#\ of\ False\ Negatives} \tag{3-5}$$

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

$$F1\ score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{3-6}$$

3.4.2 Pengujian Aplikasi

Dikarenakan aplikasi masih berupa MVP (*Most Viable Product)* dengan fitur yang terbatas (fitur prediksi), pengujian yang akan dilakukan adalah *usability testing, interface testing, performance testing.*

1. *Usability testing* yang akan dilakukan pada penelitian ini adalah, menguji bahwa fitur prediksi dapat berfungsi dengan baik
2. *Interface testing* yang akan dilakukan adalah menguji bahwa aplikasi mudah digunakan, bahwa UI dari aplikasi mudah untuk dinavigasi, dan tombol sudah sesuai.
3. *Performance testing* yang dilakukan adalah dengan menguji performa aplikasi pada gawai yang berbeda dan kecepatan muat aplikasi

Ketiga jenis pengujian diatas akan dilakukan dengan melakukan sampel prediksi dan survei kepada sepuluh pengguna smarphone. Sampel prediksi dilakukan untuk menguji fitur yang terdapat pada aplikasi, fitur unggah foto dan fitur prediksi. Hal ini dilakukan untuk memastikan bahwa aplikasi dapat bekerja dengan semestinya, foto dapat diunggah tanpa ada error dan fitur prediksi dapat berfungsi dengan baik serta menampilkan hasil prediksi yang diharapkan

Survei dilakukan dengan memberi pertanyaan kepada pengguna smarphone setelah mencoba aplikasi web. Hal ini diperlukan untuk mengetahui bahwa aplikasi web dapat dimuat pada smarphone pengguna dan pengalaman pengguna dalam menggunakan aplikasi web.

Bab IV
Hasil dan Pembahasan

Pada bab ini akan dibagi dalam lima bagian yaitu hasil dari percobaan yang telah dilakukan, perbandingan hasil percobaan, aplikasi web, hasil pengujian aplikasi , dan hasil percobaan aplikasi web pada sepuluh *user* yang berbeda. Terdapat 16 percobaan yang dilakukan, percobaan ini dilakukan untuk mengetahui model ML mana yang memiliki performa terbaik. Pada setiap percobaan menggunakan arsitektur yang sama untuk *Transfer Learning,* yaitu EfficientNetB0. Hasil percobaan dapat dilihat pada table 4.1.

Tabel 4.1 Percobaan yang dilakukan

No.	*Optimizer*	*Learning Rate*	*Epoch*	*Image Size*
1-a	SGD	1e-3	20	128x128
1-b		1e-5 (fine tuned)	10	128x128
2-a		1e-4	20	128x128
2-b		1e-5 (fine tuned)	10	128x128
3-a		1e-4	30	128x128
3-b		1e-5(fine tuned)	15	128x128
1-a	ADAM	1e-3	20	128x128
1-b		1e-5 (fine tuned)	10	128x128
2-a		1e-4	20	128x128
2-b		1e-5 (fine tuned)	10	128x128
3-a		1e-4	30	128x128
3-b		1e-5(fine tuned)	15	128x128
1-a	RMSprop	1e-3	20	128x128
1-b		1e-5 (fine tuned)	10	128x128
2-a		1e-4	20	128x128
2-b		1e-5 (fine tuned)	10	128x128
3-a		1e-4	30	128x128
3-b		1e-5(fine tuned)	15	128x128

Pada setiap percobaan dilakukan augmentasi data. Augmentasi data dilakukan untuk mengurangi *overfitting*. Augmentasi data yang dilakukan adalah *horizontal flip* juga *random height*, *random width*, *random zoom*, dan *random rotation* yang masing masing bernilai 0,2. *Overfitting* terjadi ketika performa model pada data *training* memiliki akurasi yang tinggi, namun model memiliki

kemampuan generalisasi yang buruk data yang lain (akurasi rendah pada data validasi dan data tes). Selain itu pada setiap percobaan menggunakan *dropout* default dari EfficientNetB0 sebesar 0,2 dan batch_size dengan nilai 32.

Pada bagian hasil percobaan, hasil dari enam belas percobaan yang telah dilakukan akan dibandingkan dan dipilih satu model terbaik yang akan ditamankan pada aplikasi web. Pada bagian aplikasi web akan dijabarkan cara kerja aplikasi dan penjelasan dari *source code* aplikasi web. Pada bagian pengujian aplikasi akan dijabarkan hasil pengujian berupa hasil prediksi yang dilakukan pada aplikasi dengan sepuluh sampel gambar yang diambil secara acak dari data tes.

Pada bagian hasil percobaan aplikasi akan dijabarkan hasil survei dari sepuluh pengguna dengan *smarphone* yang berbeda. Terdapat empat pertanyaan yang diajukan pada responden. Berikut adalah pertanyaan yang diajukan

1. Apakah aplikasi dapat berjalan?
2. Seberapa cepat waktu *smarphone* dalam memuat aplikasi?
3. Apakah fitur aplikasi dapat berfungsi dengan baik?
4. Bagaimana akurasi hasil prediksi aplikasi?

4.1 Hasil Percobaan

Pada bagian ini akan dijabarkan mengenai grafik akurasi dan *loss* model selama *training*, hasil akhir *training*, *confusion matrix*, dan F1-score. *Confusion matrix* dan F1-score didapat dari hasil prediksi pada model menggunakan tes dataset. Tes dataset berisi 4134 gambar yang terbagi menjadi dua kelas (*parasitized* dan *uninfected*) dengan rasio 50:50. Penggunaan arsitektur efficientnetb0 sebagai base_model untuk *Transfer Learning* dapat dilihat pada gambar 4.1

```
1  base_model=tf.keras.applications.EfficientNetB0(include_top=F
   alse)
2  base_model.trainable = False
```

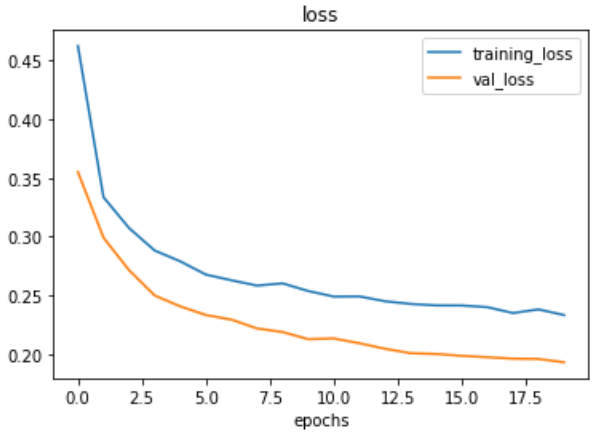
Gambar 4. 1 Cara melakukan *Transfer Learning*

Parameter `include_top=False` pada gambar 4.1 diperlukan supaya *layer* paling atas (*output layer)* dari model efficientnetb0 tidak ikut terpakai. Hal ini dikarenakan model efficientnetb0 menggunakan *benchmark* dataset imagenet yang memiliki 1000 kelas (*ouput layer = 1000* unit neuron). Pada penelitian ini model

yang dibutuhkan adalah sebagai *feature extractor* (semua *layer* pada model sebelum output layer). Percobaan dengan label "a" merupakan *training* model tanpa *fine tune*, sedangkan label "b" *training* dengan *fine tune.*

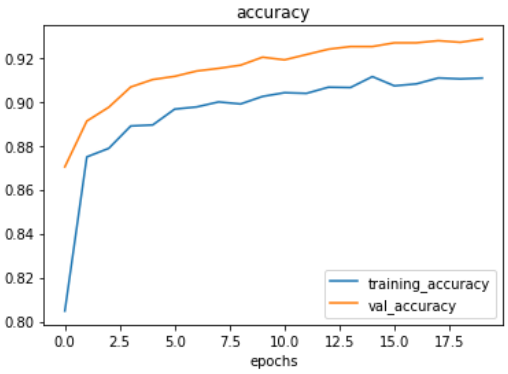
4.1.1 Percobaan SGD 1-a

Parameter yang digunakan pada percobaan ini antara lain *learning rate* 1e-3 (0,001), ukuran gambar 128x128, *epoch* berjumlah 20 dan *optimizer* SGD. Berikut adalah hasilnya.



Gambar 4. 2 Grafik *loss* percobaan SGD 1-a

Pada gambar 4.2 dapat dilihat bahwa nilai *loss* turun seiring bertambahnya *epoch* dan mulai konvergen pada *epoch* ke-15. Konvergen merupakan kondisi di mana nilai *loss* pada saat *training* telah mendekati rentang tertentu dan hanya mengalami penurunan yang kecil pada epoch berikutnya.

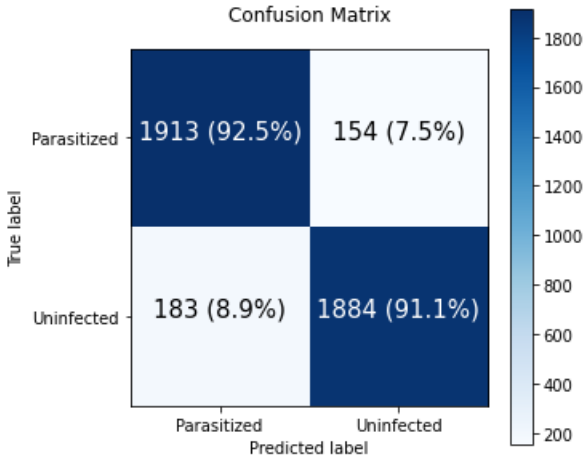


Gambar 4. 3 Grafik akurasi percobaan SGD 1-a

Pada gambar 4.3 dapat dilihat bahwa nilai akurasi terus bertambah seiring bertambahnya epoch. Untuk akurasi *training* mulai dari nilai 0,80 dan pada *epoch* ke-20 mecapai nilai 0,90, sedangkan *val_accuracy* mulai pada nilai 0,87 dan pada epoch ke-20 mencapai nilai 0,93. Selisih antara *training_accuracy* dan *val-_accuracy* kurang lebih 0,018. Hasil akhir *training* dapat dilihat pada tabel 4.2 dan gambar 4.4.

Tabel 4. 2 Hasil *training* percobaan SGD 1-a

Train loss	Training accuracy	Validation loss	Validation accuracy
0,2333	0,9107	0,1932	0,9284



Gambar 4. 4 *Confusion matrix* percobaan SGD 1-a

Pada gambar 4.4 dapat dilihat hasil prediksi model. Untuk kelas *parasitized* terprediksi benar sejumlah 1913 citra (92,5%) dari 2067 citra, sedangkan untuk kelas *uninfected* terprediksi benar sejumlah 1884 citra (91,1%) dari 2067 citra. Dari hasil *confusion matrix* model cukup baik dalam mengeneralisasi . Hal ini dapat dilihat selisih hasil prediksi yang kecil, *Uninfected* 91,1% dan *Parasitized* 92,5%. F1-score dapat dilihat pada tabel 4.3

Tabel 4. 3 F1-score percobaan SGD 1-a

Accuracy	*Precision*	*Recall*	*F1-score*
0,9184	0,918	0,9184	0,9184

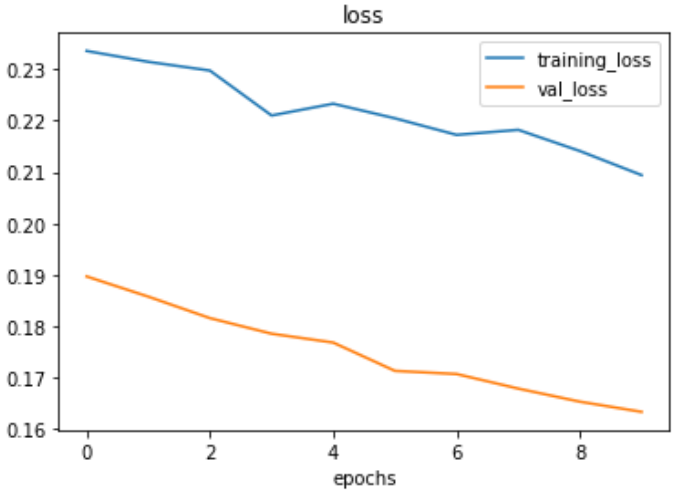
4.1.2 Percobaan SGD 1-b

Pada percobaan ini dilakukan *fine tune* pada model hasil percobaan SGD 1-a. *Fine tuning* dilakukan dengan cara merubah parameter `base_model.trainable=False` ke `base_model.trainable=True.` Kemudian model di *compile* ulang sehingga perubahan yang dibuat dapat berlaku. Fine tuning merupakan proses *un-freeze* layer dari base model sehingga weight dari layer tersebut dapat diupdate. Kode untuk fine tuning dapat dilihat pada gambar 4.5

```
1  base_model.trainable=True
2  model.compile(loss="categorical_crossentropy",
          optimizer=tf.keras.optimizers.SGD(learning_rate=1e-5),
          metrics=["accuracy"])
```

Gambar 4. 5 *Fine Tuning*

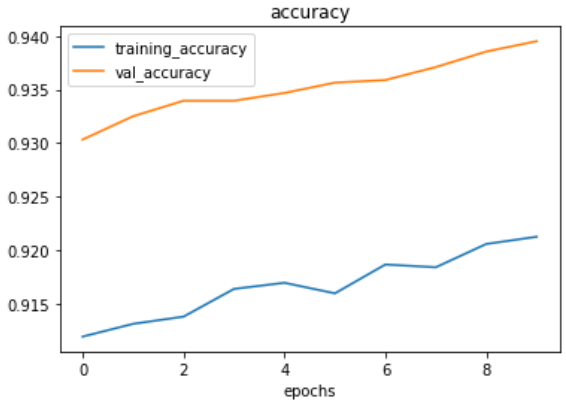
Parameter yang digunakan pada percobaan ini antara lain learning rate 1e-5 (0,00001), *epoch* berjumlah 10 ukuran gambar 128x128 dan *optimizer* SGD. *Learning rate* yang kecil pada saat *fine tuning* diperlukan untuk menghindari *overfitting*. Berikut adalah hasilnya.



Gambar 4. 6 Grafik *loss* percobaan SGD 1-b

Pada gambar 4.6 dapat dilihat bahwa seiring bertambahnya epoch nilai *loss* turun hingga epoch terakhir (ke-10), namun titik konvergen belum dapat ditentukan karena tren turunya nilai loss masih terus berlanjut hingga *epoch* ke-10 dengan nilai

akhir *training loss* 0,21 dan *validation loss* 0,16. Untuk grafik akurasi dapat dilihat pada gambar 4.7

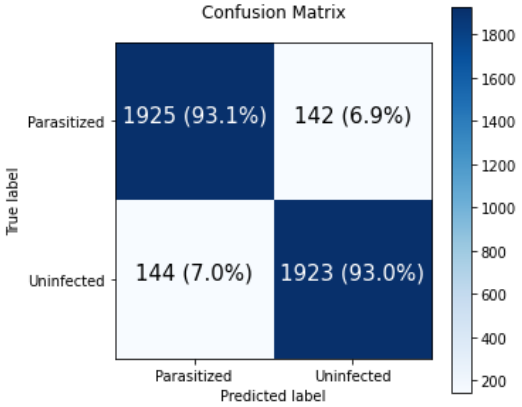


Gambar 4. 7 Grafik akurasi percobaan SGD 1-b

Pada gambar 4.7 dapat dilihat bahwa nilai akurasi terus bertambah seiring bertambahnya *epoch*. Untuk akurasi *training* mulai dari nilai 0,91 dan pada *epoch* ke-10 mecapai nilai 0,92, sedangkan pada *val_accuracy* mulai pada nilai 0,93 dan pada epoch ke-10 mencapai nilai 0,94. Selisih antara *training_accuracy* dan *val-_accuracy* kurang lebih 0,0185. Hasil akhir *training* dapat dilihat pada tabel 4.4

Tabel 4. 4 Hasil *training* percobaan SGD 1-b

Train loss	*Training accuracy*	*Validation loss*	*Validation accuracy*
0,2094	0,9213	0,1633	0,9395



Gambar 4. 8 *Confusion Matrix* percobaan SGD 1-b

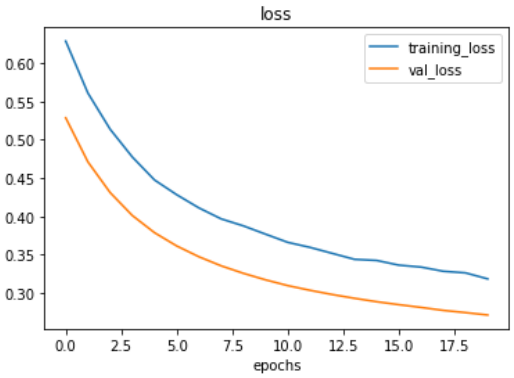
Pada gambar 4.8 dapat dilihat hasil prediksi model. Untuk kelas *parasitized* terprediksi benar sejumlah 1925 citra (93,1%) dari 2067 citra, sedangkan untuk kelas *uninfected* terprediksi benar sejumlah 1923 citra (93,0%) dari 2067 citra. . Dari hasil *confusion matrix*, model cukup baik dalam mengeneralisasi . Hal ini dapat dilihat selisih hasil prediksi yang kecil, *Uninfected* 93,0% dan *Parasitized* 93,1%. Untuk F1-score dapat dilihat pada tabel 4.5

Tabel 4. 5 F1-score percobaan SGD 1-b

Accuracy	*Precision*	*Recall*	*F1-score*
0,9308	0,9308	0,9308	0,9308

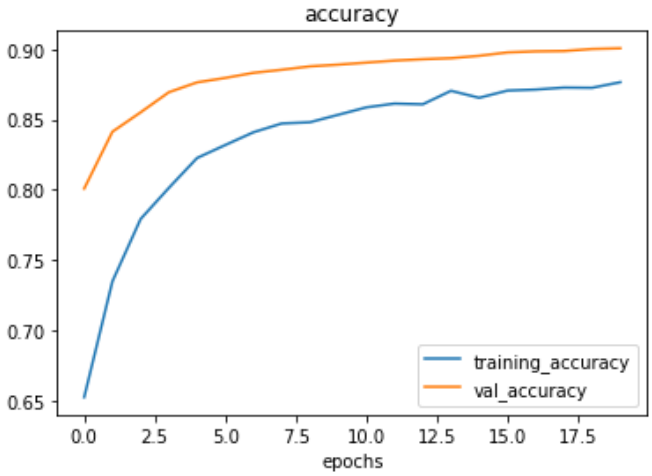
4.1.3 Percobaan SGD 2-a

Parameter yang digunakan pada percobaan ini antara lain learning rate 1e-4 (0,0001), *epoch* berjumlah 20, ukuran gambar 128x128 dan optimizer SGD.



Gambar 4. 9 Grafik *loss* percobaan SGD 2-a

Pada gambar 4.9 dapat dilihat bahwa nilai *loss* turun seiring bertambahnya *epoch* dan mulai konvergen pada *epoch* ke-17. Ketika model telah mengalami konvergen, pada *epoch* berikutnya penurunan nilai *loss* dari *epoch* sebelumnya akan semakin kecil hingga model mengalami *bottleneck* (nilai *loss* sudah tidak mengalami penurunan lagi)

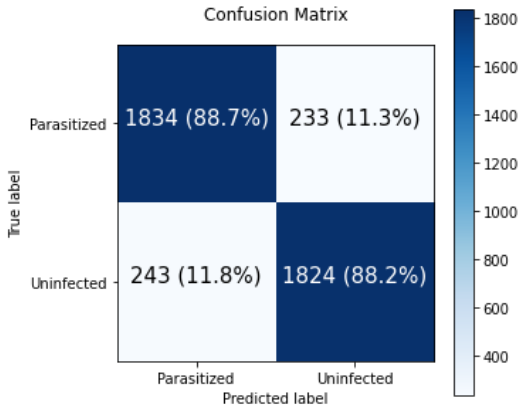


Gambar 4. 10 Grafik akurasi SGD 2-a

Pada gambar 4.10 dapat dilihat bahwa nilai akurasi terus bertambah seiring bertambahnya epoch. Untuk akurasi *training* mulai dari nilai 0,65 dan pada *epoch* ke-20 mecapai nilai 0,876, sedangkan pada *val_accuracy* mulai dari nilai 0,80 dan pada epoch ke-20 mencapai nilai 0,90. Selisih antara *training_accuracy* dan *val-_accuracy* kurang lebih 0,024. Hasil akhir *training* dapat dilihat pada tabel 4.6

Tabel 4. 6 Hasil *training* percobaan SGD 2-a

Train loss	*Training accuracy*	*Validation loss*	*Validation accuracy*
0,3185	0,8765	0,2714	0,9006



Gambar 4. 11 *Confusion Matrix* percobaan SGD 2-a

Pada gambar 4.11 dapat dilihat hasil prediksi model. Untuk kelas *parasitized* terprediksi benar sejumlah 1834 citra (88,7%) dari 2067 citra,

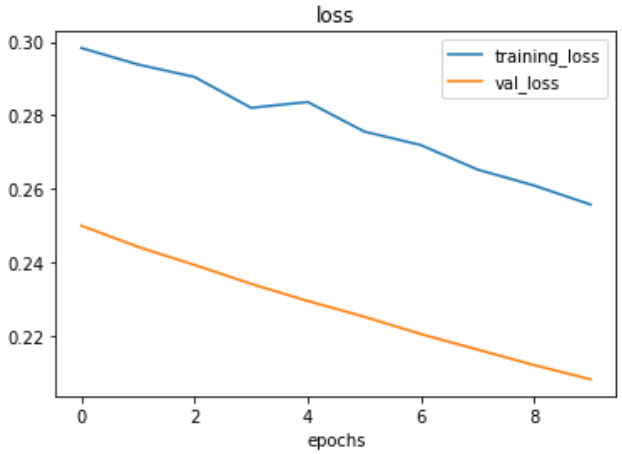
sedangkan untuk kelas *uninfected* terprediksi benar sejumlah 1824 citra (88,2%) dari 2067 citra. . Dari hasil *confusion matrix* model cukup baik dalam mengeneralisasi . Hal ini dapat dilihat selisih hasil prediksi yang kecil, *Uninfected* 88,2% dan *Parasitized* 88,7%. Untuk F1-score dapat dilihat pada tabel 4.7

Tabel 4. 7 F1-score percobaan SGD 2-a

Accuracy	*Precision*	*Recall*	*F1-score*
0,8848	0,8848	0,8848	0,8848

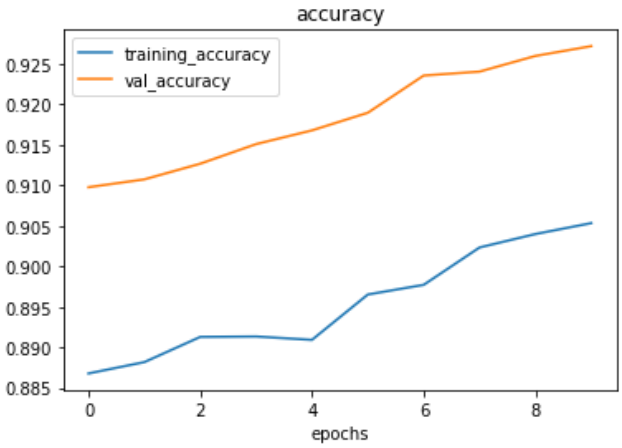
4.1.4 Percobaan SGD 2-b

Pada percobaan ini dilakukan *fine tune* pada model hasil percobaan SGD 2-a. learning rate diubah menjadi 1e-5 (0,00001). Berikut adalah hasilnya.



Gambar 4. 12 Grafik *loss* percobaan SGD 2-b

Pada gambar 4.12 dapat dilihat bahwa seiring bertambahnya *epoch* nilai *loss* turun hingga *epoch* terakhir (ke-10), namun titik konvergen belum dapat ditentukan karena tren turunya nilai loss masih terus berlanjut hingga *epoch* ke-10 dengan nilai akhir *training loss* 0,25 dan *validation loss* 0,2.

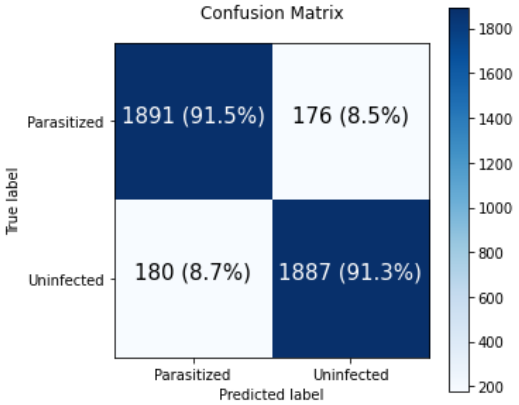


Gambar 4. 13 Grafik akurasi percobaan SGD 2-b

Pada gambar 4.13 dapat dilihat bahwa nilai akurasi terus bertambah seiring bertambahnya *epoch*. Untuk akurasi *training* mulai dari nilai 0,88 dan pada *epoch* ke-10 mecapai nilai 0,90, sedangkan pada *val_accuracy* mulai dari nilai 0,91 dan pada epoch ke-10 mencapai nilai 0,92. Selisih antara *training_accuracy* dan *val-_accuracy* kurang lebih 0,02. Hasil akhir *training* dapat dilihat pada tabel 4.8

Tabel 4. 8 Hasil *training* percobaan SGD 2-b

Train loss	*Training accuracy*	*Validation loss*	*Validation accuracy*
0,2557	0,9053	0,2082	0,9272



Gambar 4. 14 *Confusion Matrix* percobaan SGD 2-b

Pada gambar 4.14 dapat dilihat hasil prediksi model. Untuk kelas *parasitized* terprediksi benar sejumlah 1891 citra (91,5%) dari 2067 citra,

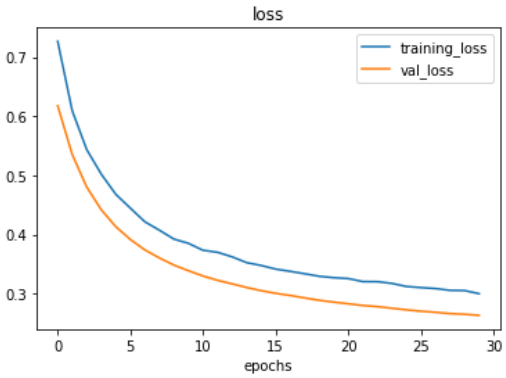
sedangkan untuk kelas *uninfected* terprediksi benar sejumlah 1887 citra (91,3%) dari 2067 citra. Dari hasil *confusion matrix* model cukup baik dalam mengeneralisasi . Hal ini dapat dilihat selisih hasil prediksi yang kecil, *Uninfected* 91,3% dan *Parasitized* 91,5%. Untuk F1-score dapat dilihat pada tabel 4.9

Tabel 4. 9 F1-score percobaan SGD 2-b

Accuracy	*Precision*	*Recall*	*F1-score*
0,9138	0,9138	0,9138	0,9138

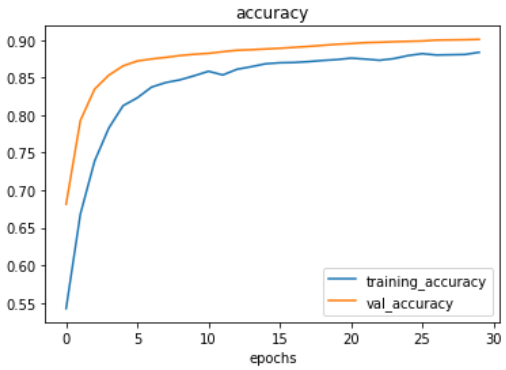
4.1.5 Percobaan SGD 3-a

Parameter yang digunakan pada percobaan ini antara lain learning rate 1e-4 (0,0001), ukuran gambar 128x128, *epoch* berjumlah 30 dan optimizer SGD.



Gambar 4. 15 Grafik *loss* percobaan SGD 3-a

Pada gambar 4.15 dapat dilihat bahwa nilai *loss* turun seiring bertambahnya *epoch* dan mulai konvergen pada *epoch* ke-25.

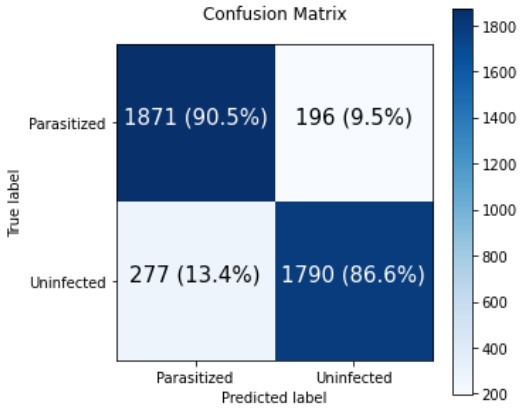


Gambar 4. 16 Grafik akurasi percobaan SGD 3-a

Pada gambar 4.16 dapat dilihat bahwa nilai akurasi terus bertambah seiring bertambahnya epoch. Untuk akurasi *training* mulai dari nilai 0,55 dan pada *epoch* ke-30 mecapai nilai 0,886, sedangkan pada *val_accuracy* mulai dari nilai 0,70 dan pada epoch ke-30 mencapai nilai 0,90. Selisih antara *training_accuracy* dan *val-_accuracy* kurang lebih 0,02. Hasil akhir *training* dapat dilihat pada tabel 4.10

Tabel 4. 10 Hasil *training* percobaan 3-a

Train loss	*Training accuracy*	*Validation loss*	*Validation accuracy*
0,3057	0,8804	0,2664	0,9001



Gambar 4. 17 *Confusion Matrix* percobaan SGD 3-a

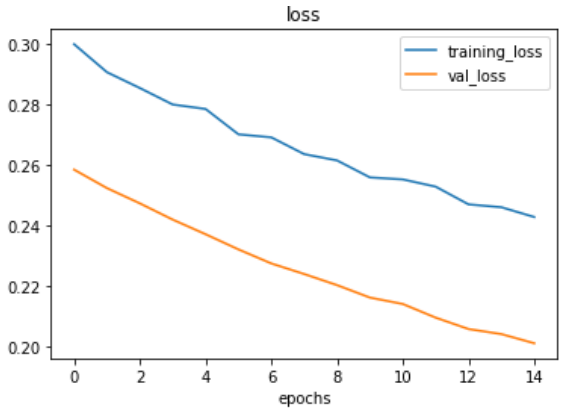
Pada gambar 4.17 dapat dilihat hasil prediksi model. Untuk kelas *parasitized* terprediksi benar sejumlah 1871 citra (90,5%) dari 2067 citra, sedangkan untuk kelas *uninfected* terprediksi benar sejumlah 1790 citra (88,6%) dari 2067 citra. . Dari hasil *confusion matrix* model cukup baik dalam mengeneralisasi . Hal ini dapat dilihat selisih hasil prediksi yang kecil, *Uninfected* 86,6% dan *Parasitized* 86,6%. Untuk F1-score dapat dilihat pada tabel 4.11

Tabel 4. 11 F1-score percobaan SGD 3-a

Accuracy	*Precision*	*Recall*	*F1-score*
0,8848	0,8848	0,8848	0,8848

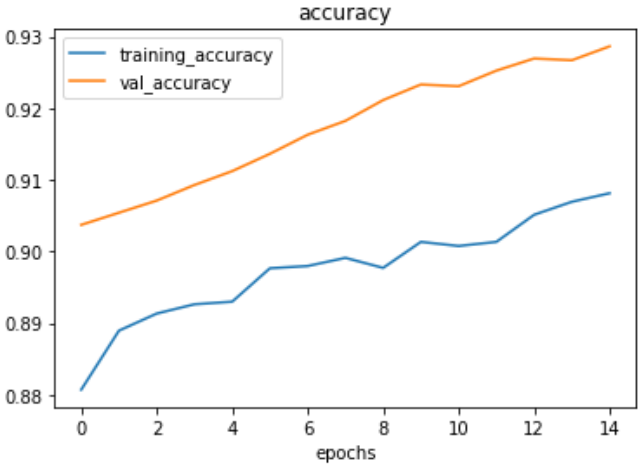
4.1.6 Percobaan SGD 3-b

Pada percobaan ini dilakukan *fine tune* pada model hasil percobaan SGD 3-a. *Parameter* yang digunakan pada percobaan ini antara lain learning rate 1e-5 (0,00001), *epoch* bernilai 15, ukuran gambar 128x128 dan *optimizer* SGD. Berikut adalah hasilnya.



Gambar 4. 18 Gambar *loss* percobaan SGD 3-b

Pada gambar 4.18 dapat dilihat bahwa seiring bertambahnya *epoch* nilai *loss* turun hingga *epoch* terakhir (ke-15), namun titik konvergen belum dapat ditentukan karena tren turunya nilai loss masih terus berlanjut hingga *epoch* ke-15 dengan nilai akhir *training loss* 0,242 dan *validation loss* 0,2.

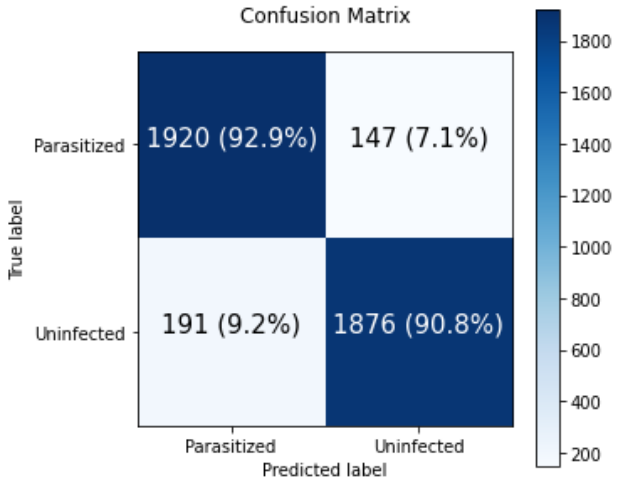


Gambar 4. 19 Grafik akurasi percobaan SGD 3-b

Pada gambar 4.19 dapat dilihat bahwa nilai akurasi terus bertambah seiring bertambahnya *epoch*. Untuk akurasi *training* mulai dari nilai 0,88 dan pada *epoch* ke-15 mecapai nilai 0,90, sedangkan pada *val_accuracy* mulai dari nilai 0,90 dan pada epoch ke-15 mencapai nilai 0,93. Selisih antara *training_accuracy* dan *val-_accuracy* kurang lebih 0,02. Hasil akhir *training* dapat dilihat pada tabel 4.12

Tabel 4. 12 Hasil *training* SGD 3-b

Train loss	*Training accuracy*	*Validation loss*	*Validation accuracy*
0,2427	0,9081	0,2009	0,9286



Gambar 4. 20 *Confusion Matrix* percobaan SGD 3-b

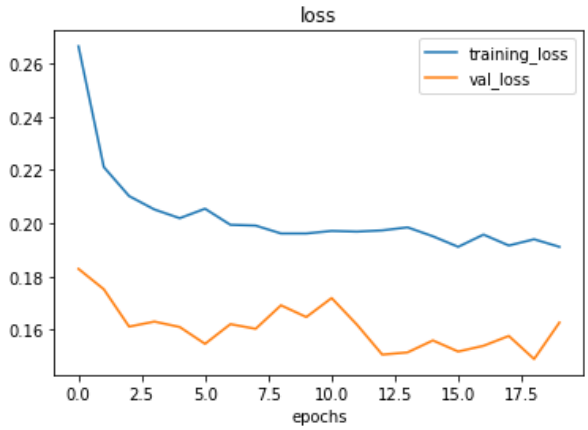
Pada gambar 4.20 dapat dilihat hasil prediksi model. Untuk kelas *parasitized* terprediksi benar sejumlah 1920 citra (92,9%) dari 2067 citra, sedangkan untuk kelas *uninfected* terprediksi benar sejumlah 1876 citra (90,8%) dari 2067 citra. . Dari hasil *confusion matrix* model cukup baik dalam mengeneralisasi . Hal ini dapat dilihat selisih hasil prediksi yang kecil, *Uninfected* 90,8% dan *Parasitized* 92,9%. Untuk F1-score dapat dilihat pada tabel 4.13

Tabel 4. 13 F1-score percobaan SGD 3-b

Accuracy	*Precision*	*Recall*	*F1-score*
0,9182	0,9184	0,9182	0,9182

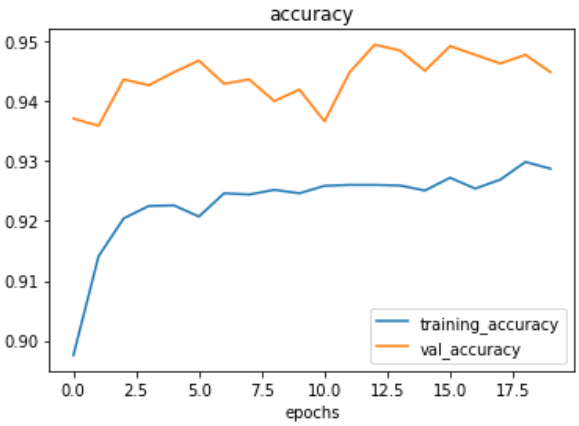
4.1.7 Percobaan Adam 1-a

Parameter yang digunakan pada percobaan ini antara lain learning rate 1e-3 (0,001), *epoch* berjumlah 20, ukuran gambar 128x128 dan optimizer Adam. Pada percobaan ini menggunakan optimizer Adam untuk mengetahui bahwa performa model akan lebih baik jika dibandingkan dengan model yang pada saat proses *training* menggunakan SGD sebagai *optimizer-nya*. Berikut adalah hasilnya.



Gambar 4. 21 Grafik *loss* percobaan Adam 1-a

Pada gambar 4.21 dapat dilihat bahwa nilai *loss* turun seiring bertambahnya *epoch* dan mulai konvergen pada *epoch* ke-5, untuk *epoch* berikutnya hanya mengalami sedikit penurunan hingga *epoch* ke-20. Berbeda dengan SGD jika menggunakan Adam, pada *epoch* pertama model telah mencapai *loss* yang relatif rendah, namun untuk *epoch* selanjutnya hanya mengalami sedikit peningkatan.

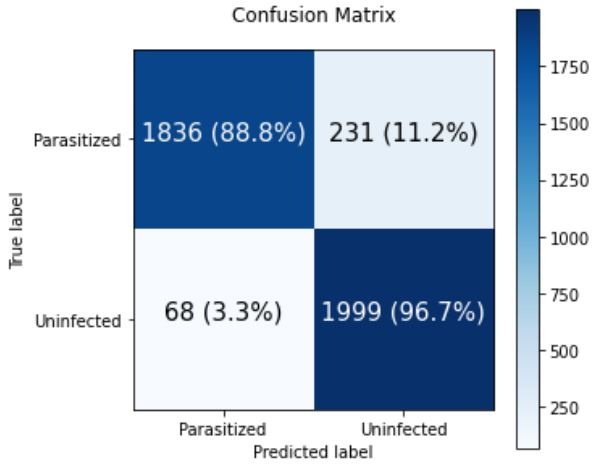


Gambar 4. 22 Grafik akurasi Adam 1-a

Pada gambar 4.22 dapat dilihat bahwa nilai akurasi terus bertambah seiring bertambahnya epoch. Untuk akurasi *training* mulai dari nilai 0,90 dan pada *epoch* ke-20 mecapai nilai 0,93, sedangkan pada *val_accuracy* mulai dari nilai 0,937 dan pada epoch ke-20 mencapai nilai 0,944. Selisih antara *training_accuracy* dan *val-_accuracy* kurang lebih 0,016. Berbeda dengan SGD jika menggunakan Adam, pada *epoch* pertama model telah mencapai *training* akurasi yang relatif tinggi (0,90), namun untuk *epoch* selanjutnya hanya mengalami sedikit penurunan. Hasil akhir *training* dapat dilihat pada tabel 4.14

Tabel 4. 14 Hasil *training* percobaan Adam 1-a

Train loss	*Training accuracy*	*Validation loss*	*Validation accuracy*
0,1912	0,9287	0,1627	0,9448



Gambar 4. 23 *Confusion Matrix* percobaan Adam 1-a

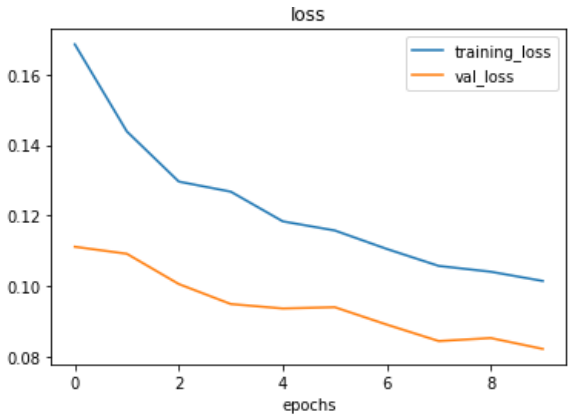
Pada gambar 4.23 dapat dilihat hasil prediksi model. Untuk kelas *parasitized* terprediksi benar sejumlah 1836 citra (88,8%) dari 2067 citra, sedangkan untuk kelas *uninfected* terprediksi benar sejumlah 1999 citra (96,7%) dari 2067 citra. . Dari hasil *confusion matrix* model relatif kurang baik dalam menggeneralisasi pada klas *parasitized* . Hal ini dapat dilihat dari selisih hasil prediksi yang relatif besar (selisih > 4%), *Uninfected* 96,7% dan *Parasitized* 88,7%. Untuk F1-score dapat dilihat pada tabel 4.15

Tabel 4. 15 F1-score percobaan Adam 1-a

Accuracy	*Precision*	*Recall*	*F1-score*
0,9276	0,93	0,927	0,9275

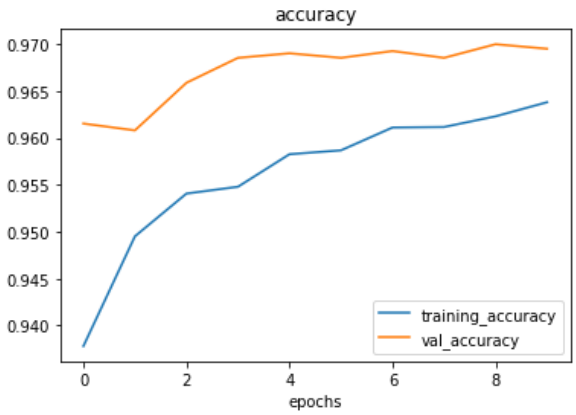
4.1.8 Percobaan Adam 1-b

Pada percobaan ini dilakukan *fine tune* pada model hasil percobaan Adam 1-a. *Parameter* yang digunakan pada percobaan ini antara lain learning rate 1e-5 (0,00001), ukuran gambar 128x128 dan *optimizer* Adam. Berikut adalah hasilnya.



Gambar 4. 24 Grafik *loss* percobaan Adam 1-b

Pada gambar 4.24 dapat dilihat bahwa seiring bertambahnya *epoch* nilai *loss* turun hingga *epoch* terakhir (ke-10), namun titik konvergen belum dapat ditentukan karena tren turunya nilai loss masih terus berlanjut hingga *epoch* ke-10 dengan nilai akhir *training loss* 0,10 dan *validation loss* 0,086.

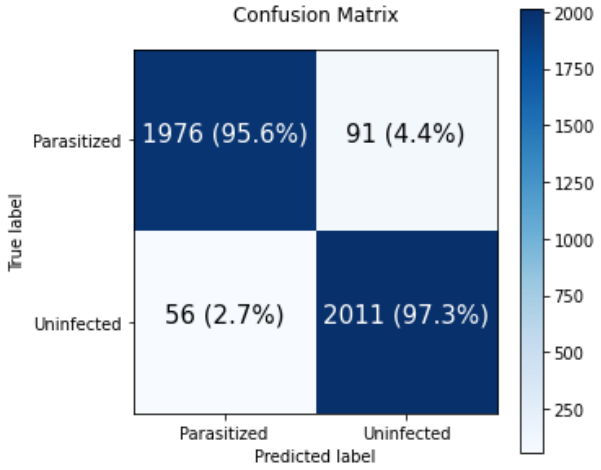


Gambar 4. 25 Grafik akurasi percobaan Adam 1-b

Pada gambar 4.25 dapat dilihat bahwa nilai akurasi terus bertambah seiring bertambahnya *epoch*. Untuk akurasi *training* mulai dari nilai 0,94 dan pada *epoch* ke-10 mecapai nilai 0,963, sedangkan pada *val_accuracy* mulai dari nilai 0,962 dan pada epoch ke-10 mencapai nilai 0,97. Selisih antara *training_accuracy* dan *val-_accuracy* kurang lebih 0,008. Hasil akhir *training* dapat dilihat pada tabel 4.16

Tabel 4. 16 Hasil *training* Percobaan Adam 1-b

Train loss	*Training accuracy*	*Validation loss*	*Validation accuracy*
0,1015	0,9638	0,0823	0,9695



Gambar 4. 26 *Confusion Matrix* percobaan Adam 1-b

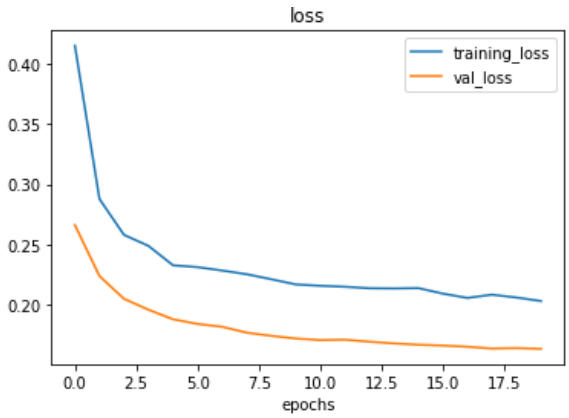
Pada gambar 4.26 dapat dilihat hasil prediksi model. Untuk kelas *parasitized* terprediksi benar sejumlah 1976 citra (95,6%) dari 2067 citra, sedangkan untuk kelas *uninfected* terprediksi benar sejumlah 2011 citra (97,3%) dari 2067 citra. Dari hasil *confusion matrix* model cukup baik dalam menggeneralisasi . Hal ini dapat dilihat dari selisih hasil prediksi yang kecil (selisih < 4%), *Uninfected* 97,3% dan *Parasitized* 95,5%. Untuk F1-score dapat dilihat pada tabel 4.17

Tabel 4. 17 F1-score percobaan Adam 1-b

Accuracy	*Precision*	*Recall*	*F1-score*
0,9644	0,9645	0,9644	0,9644

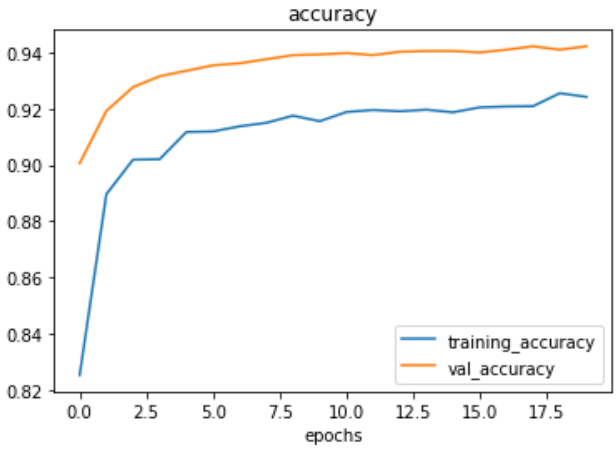
4.1.9 Percobaan Adam 2-a

Parameter yang digunakan pada percobaan ini antara lain learning rate 1e-4 (0,0001), *epoch* berjumlah 20, ukuran gambar 128x128 dan optimizer Adam. Pada percobaan ini menggunakan optimizer Adam untuk mengetahui bahwa performa model akan lebih baik jika dibandingkan dengan model yang pada saat proses *training* menggunakan SGD sebagai *optimizer-nya*. Berikut adalah hasilnya.



Gambar 4. 27 Grafik *loss* percobaan Adam 2-a

Pada gambar 4.27 dapat dilihat bahwa nilai *loss* turun seiring bertambahnya *epoch* dan mulai konvergen pada *epoch* ke-10, untuk *epoch* berikutnya hanya mengalami sedikit penurunan hingga *epoch* ke-20. Berbeda dengan SGD jika menggunakan Adam, pada *epoch* pertama model telah mencapai *loss* yang relatif rendah, namun untuk *epoch* selanjutnya hanya mengalami sedikit penurunan.

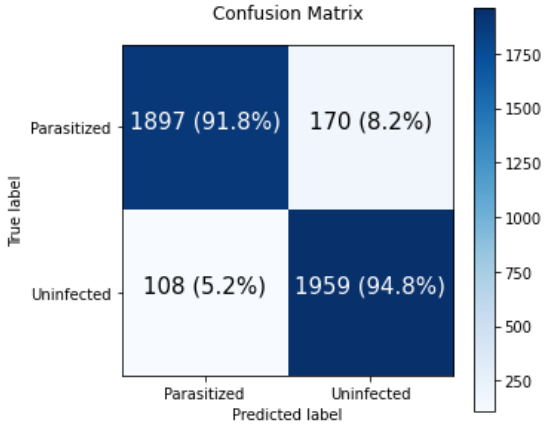


Gambar 4. 28 Grafik akurasi Adam 2-a

Pada gambar 4.28 dapat dilihat bahwa nilai akurasi terus bertambah seiring bertambahnya *epoch*. Pada akurasi *training* mulai dari nilai 0,82 dan saat *epoch* ke-20 mecapai nilai 0,924, sedangkan *val_accuracy* mulai dari nilai 0,907 dan pada epoch ke-20 mencapai nilai 0,942. Selisih antara *training_accuracy* dan *val-_accuracy* kurang lebih 0,02. Pada percobaan ini, saat *epoch* pertama model telah mencapai akurasi yang relatif tinggi (0,82), namun untuk *epoch* selanjutnya hanya mengalami sedikit peningkatan. Hasil akhir *training* dapat dilihat pada tabel 4.18

Tabel 4. 18 Hasil *training* percobaan Adam 2-a

Train loss	*Training accuracy*	*Validation loss*	*Validation accuracy*
0,1912	0,9287	0,1627	0,9448



Gambar 4. 29 *Confusion Matrix* percobaan Adam 2-a

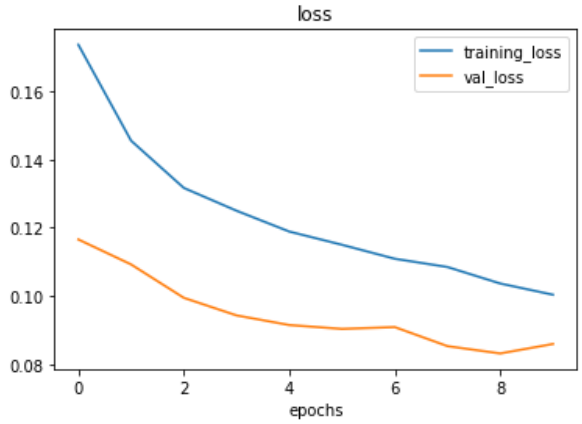
Pada gambar 4.29 dapat dilihat hasil prediksi model. Untuk kelas *parasitized* terprediksi benar sejumlah 1897 citra (91,8%) dari 2067 citra, sedangkan untuk kelas *uninfected* terprediksi benar sejumlah 1959 citra (94,8%) dari 2067 citra. . Dari hasil *confusion matrix* model relatif cukup baik dalam menggeneralisasi pada kelas *parasitized* . Hal ini dapat dilihat dari selisih hasil prediksi yang relatif kecil (selisih < 4%), *Uninfected* 94,8% dan *Parasitized* 94,8%. Untuk F1-score dapat dilihat pada tabel 4.19

Tabel 4. 19 F1-score percobaan Adam 2-a

Accuracy	*Precision*	*Recall*	*F1-score*
0,9327	0,9331	0,9327	0,9327

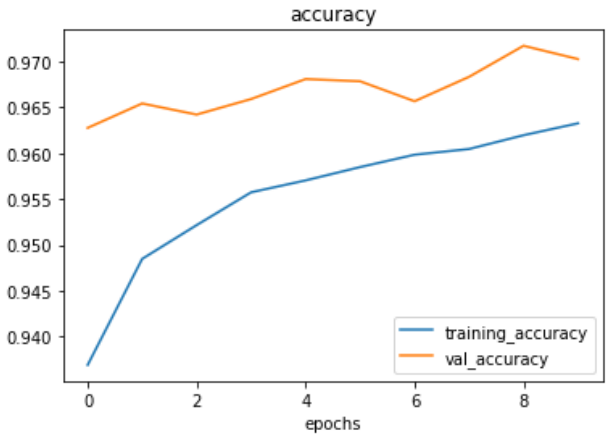
4.1.10 Percobaan Adam 2-b

Pada percobaan ini dilakukan *fine tune* pada model hasil percobaan Adam 2-a. *Parameter* yang digunakan pada percobaan ini antara lain learning rate 1e-5 (0,00001), ukuran gambar 128x128 dan *optimizer* Adam. Berikut adalah hasilnya.



Gambar 4. 30 Grafik *loss* percobaan Adam 2-b

Pada gambar 4.30 dapat dilihat bahwa seiring bertambahnya *epoch* nilai *loss* turun hingga *epoch* terakhir (ke-10) dan mulai konvergen pada *epoch* ke-8. dengan nilai akhir *training loss* 0,10 dan *validation loss* 0,086.



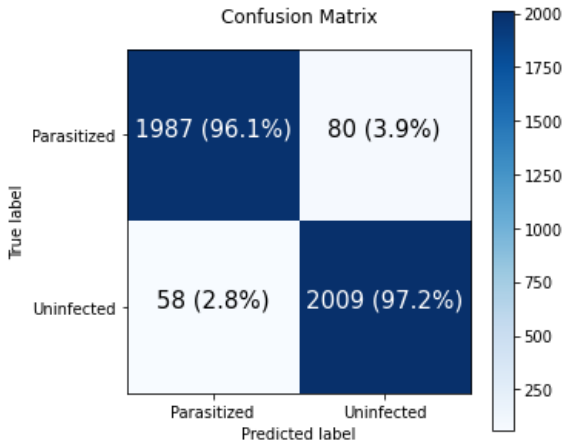
Gambar 4. 31 Grafik akurasi percobaan Adam 2-b

Pada gambar 4.31 dapat dilihat bahwa nilai akurasi terus bertambah seiring bertambahnya *epoch*. Untuk akurasi *training* mulai dari nilai 0,936 dan pada *epoch* ke-10 mecapai nilai 0,963, sedangkan pada *val_accuracy* mulai dari nilai 0,962 dan

pada epoch ke-10 mencapai nilai 0,97. Selisih antara *training_accuracy* dan *val-_accuracy* kurang lebih 0,007. Hasil akhir *training* dapat dilihat pada tabel 4.20

Tabel 4. 20 Hasil *training* Percobaan Adam 2-b

Train loss	*Training accuracy*	*Validation loss*	*Validation accuracy*
0,1004	0,9632	0,086	0,9702



Gambar 4. 32 *Confusion Matrix* percobaan Adam 2-b

Pada gambar 4.32 dapat dilihat hasil prediksi model. Untuk kelas *parasitized* terprediksi benar sejumlah 1987 citra (96,1%) dari 2067 citra, sedangkan untuk kelas *uninfected* terprediksi benar sejumlah 2011 citra (97,2%) dari 2067 citra. Dari hasil *confusion matrix* model cukup baik dalam menggeneralisasi . Hal ini dapat dilihat dari selisih hasil prediksi yang kecil (selisih < 4%), *Uninfected* 97,2% dan *Parasitized* 96,1%. Untuk F1-score dapat dilihat pada tabel 4.21

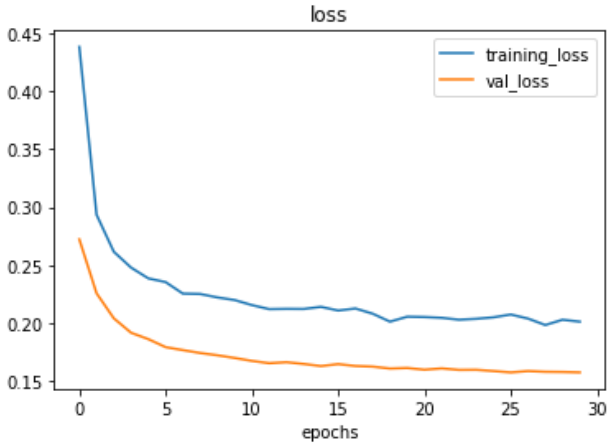
Tabel 4. 21 F1-score percobaan Adam 2-b

Accuracy	*Precision*	*Recall*	*F1-score*
0,9666	0,9666	0,9666	0,9666

4.1.11 Percobaan Adam 3-a

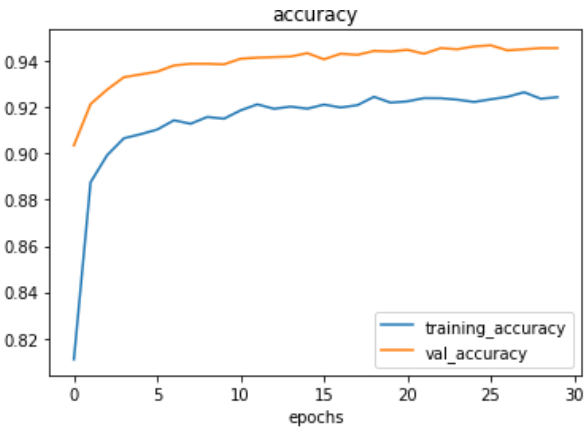
Parameter yang digunakan pada percobaan ini antara lain learning rate 1e-4 (0,0001), *epoch* berjumlah 30, ukuran gambar 128x128 dan optimizer Adam. Pada percobaan ini menggunakan optimizer Adam untuk mengetahui bahwa performa

model akan lebih baik jika dibandingkan dengan model yang pada saat proses *training* menggunakan SGD sebagai *optimizer-nya*. Berikut adalah hasilnya.



Gambar 4. 33 Grafik *loss* percobaan Adam 3-a

Pada gambar 4.33 dapat dilihat bahwa nilai *loss* turun seiring bertambahnya *epoch* dan mulai konvergen pada *epoch* ke-15, untuk *epoch* berikutnya hanya mengalami sedikit penurunan hingga *epoch* ke-30. Berbeda dengan SGD jika menggunakan Adam, pada *epoch* pertama model telah mencapai *loss* yang relatif rendah, namun untuk *epoch* selanjutnya hanya mengalami sedikit penurunan.



Gambar 4. 34 Grafik akurasi Adam 3-a

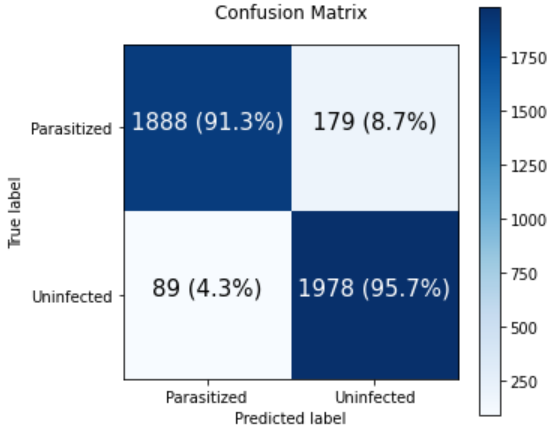
Pada gambar 4.34 dapat dilihat bahwa nilai akurasi terus bertambah seiring bertambahnya *epoch*. Pada akurasi *training* mulai dari nilai 0,82 dan saat *epoch* ke-30 mecapai nilai 0,924, sedangkan *val_accuracy* mulai dari nilai 0,907 dan pada

epoch ke-30 mencapai nilai 0,945. Selisih antara *training_accuracy* dan *val-_accuracy* kurang lebih 0,02. Pada percobaan ini, saat *epoch* pertama model telah mencapai akurasi yang relatif tinggi (0,82), namun untuk *epoch* selanjutnya hanya mengalami sedikit peningkatan. Hasil akhir *training* dapat dilihat pada tabel 4.22

Tabel 4. 22 Hasil *training* percobaan Adam 3-a

Train loss	*Training accuracy*	*Validation loss*	*Validation accuracy*
0,201	0,9244	0,1571	0,9456

Untuk *confusion matrix* dapat dilihat pada gambar berikut.



Gambar 4. 35 *Confusion Matrix* percobaan Adam 3-a

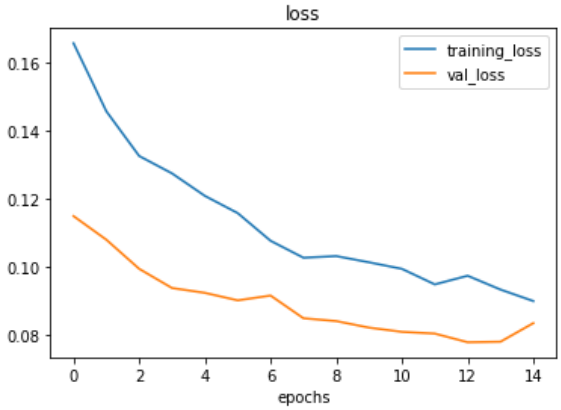
Pada gambar 4.35 dapat dilihat hasil prediksi model. Untuk kelas *parasitized* terprediksi benar sejumlah 1888 citra (91,3%) dari 2067 citra, sedangkan untuk kelas *uninfected* terprediksi benar sejumlah 1978 citra (95,7%) dari 2067 citra. Dari hasil *confusion matrix* model relatif kurang baik dalam melakukan generalisasi pada kelas *parasitized*. Hal ini dapat dilihat dari selisih hasil prediksi yang relatif besar (selisih > 4%), *Uninfected* 95,7% dan *Parasitized* 91,3%. Untuk F1-score dapat dilihat pada tabel 4.23

Tabel 4. 23 F1-score percobaan Adam 3-a

Accuracy	*Precision*	*Recall*	*F1-score*
0,9351	0,9359	0,9351	0,9351

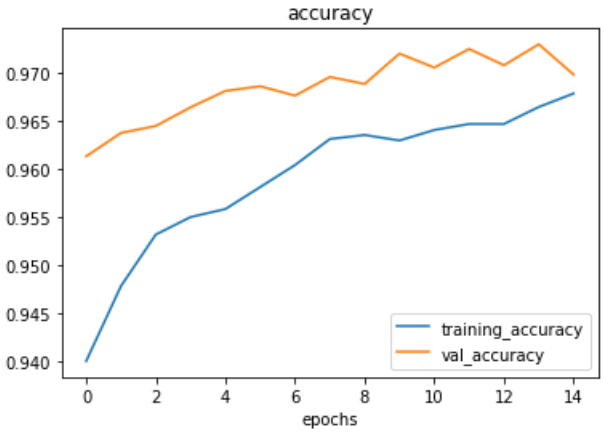
4.1.12 Percobaan Adam 3-b

Pada percobaan ini dilakukan *fine tune* pada model hasil percobaan Adam 3-a. *Parameter* yang digunakan pada percobaan ini antara lain learning rate 1e-5 (0,00001), ukuran gambar 128x128 dan *optimizer* Adam. Berikut adalah hasilnya.



Gambar 4. 36 Grafik *loss* percobaan Adam 3-b

Pada gambar 4.36 dapat dilihat bahwa seiring bertambahnya *epoch* nilai *loss* turun hingga *epoch* terakhir (ke-15) dan mulai konvergen pada *epoch* ke-11. dengan nilai akhir *training loss* 0,089 dan *validation loss* 0,083.



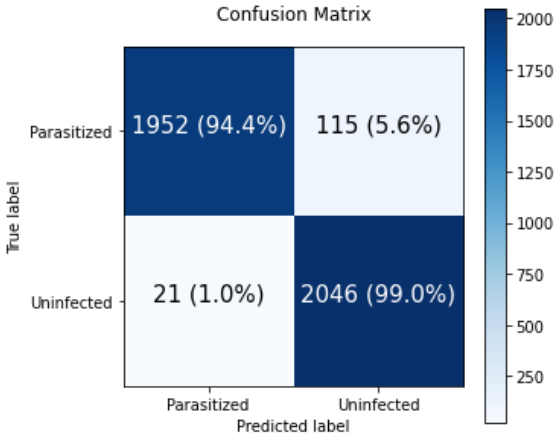
Gambar 4. 37 Grafik akurasi percobaan Adam 3-b

Pada gambar 4.37 dapat dilihat bahwa nilai akurasi terus bertambah seiring bertambahnya *epoch*. Untuk akurasi *training* mulai dari nilai 0,94 dan pada *epoch* ke-15 mecapai nilai 0,961, sedangkan pada *val_accuracy* mulai dari nilai 0,962 dan

pada epoch ke-10 mencapai nilai 0,969. Selisih antara *training_accuracy* dan *val-_accuracy* kurang lebih 0,002. Hasil akhir *training* dapat dilihat pada tabel 4.24

Tabel 4. 24 Hasil *training* Percobaan Adam 3-b

Train loss	*Training accuracy*	*Validation loss*	*Validation accuracy*
0,0899	0,9678	0,0834	0,9698



Gambar 4. 38 *Confusion Matrix* percobaan Adam 3-b

Pada gambar 4.38 dapat dilihat hasil prediksi model. Untuk kelas *parasitized* terprediksi benar sejumlah 1952 citra (94,4%) dari 2067 citra, sedangkan untuk kelas *uninfected* terprediksi benar sejumlah 2046 citra (99,0%) dari 2067 citra. Dari hasil *confusion matrix* model relatif kurang baik dalam mengeneralisasi kelas *parasitized*. Hal ini dapat dilihat dari selisih hasil prediksi yang relatif besar (selisih > 4%), *Uninfected* 99,% dan *Parasitized* 94,4%. Untuk F1-score dapat dilihat pada tabel 4.25

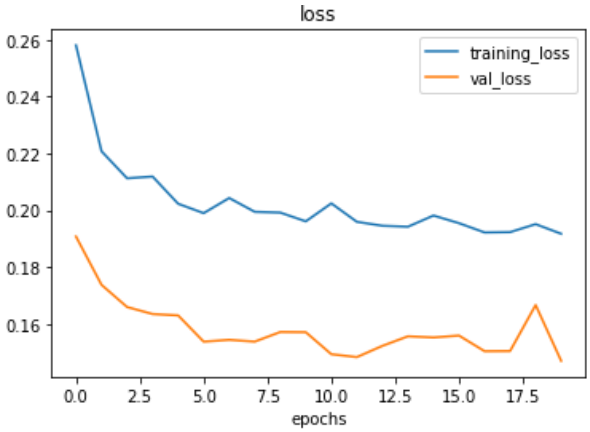
Tabel 4. 25 F1-score percobaan Adam 3-b

Accuracy	*Precision*	*Recall*	*F1-score*
0,9671	0,968	0,9671	0,967

4.1.13 Percobaan RMSprop 1-a

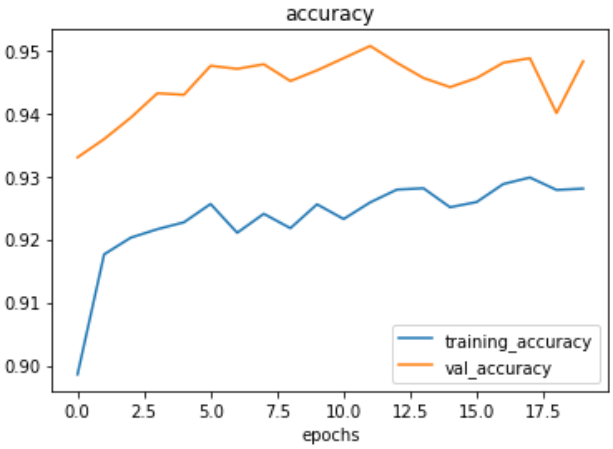
Parameter yang digunakan pada percobaan ini antara lain learning rate 1e-3 (0,001), *epoch* berjumlah 20, ukuran gambar 128x128 dan optimizer RMSprop. Pada percobaan ini menggunakan optimizer RMSprop untuk mengetahui bahwa

performa model akan lebih baik jika dibandingkan dengan model yang pada saat proses *training* menggunakan SGD atau Adam sebagai *optimizer-nya*. Berikut adalah hasilnya.



Gambar 4. 39 Grafik *loss* percobaan RMSprop 1-a

Pada gambar 4.39 dapat dilihat bahwa nilai *loss* turun seiring bertambahnya *epoch* dan mulai konvergen pada *epoch* ke-5, untuk *epoch* berikutnya hanya mengalami sedikit penurunan hingga *epoch* ke-20. Pada percobaan ini, ketika *epoch* pertama model telah mencapai *loss* yang relatif rendah bahkan lebih rendah dari SGD 1-a dan Adam 1-a, namun untuk *epoch* selanjutnya hanya mengalami sedikit penurunan.

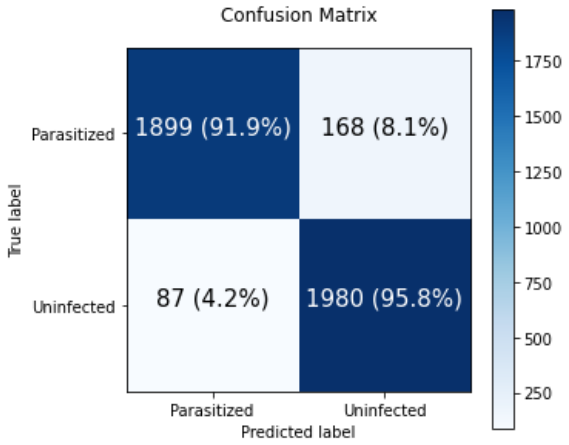


Gambar 4. 40 Grafik akurasi RMSprop 1-a

Pada gambar 4.40 dapat dilihat bahwa nilai akurasi terus bertambah seiring bertambahnya *epoch*. Pada akurasi *training* mulai dari nilai 0,90 dan saat *epoch* ke-20 mecapai nilai 0,928, sedangkan *val_accuracy* mulai dari nilai 0,933 dan pada epoch ke-30 mencapai nilai 0,948. Selisih antara *training_accuracy* dan *val-_accuracy* kurang lebih 0,02. Pada percobaan ini, saat *epoch* pertama model telah mencapai *training* akurasi yang relatif tinggi (0,90), namun untuk *epoch* selanjutnya hanya mengalami sedikit peningkatan. Hasil akhir *training* dapat dilihat pada tabel 4.26

Tabel 4. 26 Hasil *training* percobaan RMSprop 1-a

Train loss	*Training accuracy*	*Validation loss*	*Validation accuracy*
0,1917	0,928	0,1471	0,9482



Gambar 4. 41 *Confusion Matrix* percobaan RMSprop 1-a

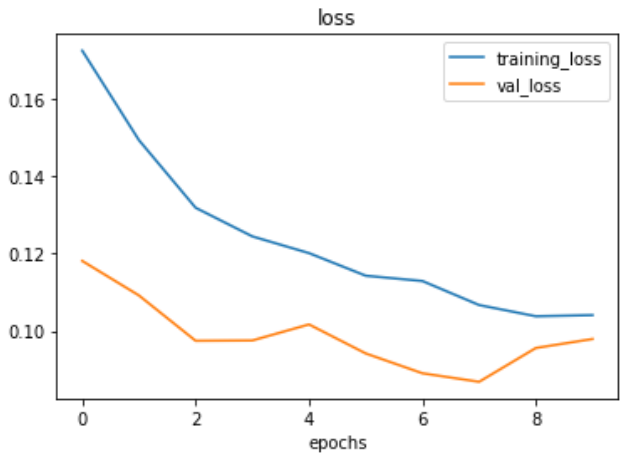
Pada gambar 4.41 dapat dilihat hasil prediksi model. Untuk kelas *parasitized* terprediksi benar sejumlah 1899 citra (91,9%) dari 2067 citra, sedangkan untuk kelas *uninfected* terprediksi benar sejumlah 1980 citra (95,8%) dari 2067 citra. Dari hasil *confusion matrix* model relatif cukup baik dalam melakukan. Hal ini dapat dilihat dari selisih hasil prediksi yang relatif kecil (selisih < 4%), *Uninfected* 95,8% dan *Parasitized* 91,9%. Untuk F1-score dapat dilihat pada tabel 4.27

Tabel 4. 27 F1-score percobaan RMSprop 1-a

Accuracy	*Precision*	*Recall*	*F1-score*
0,9383	0,9389	0,9383	0,9382

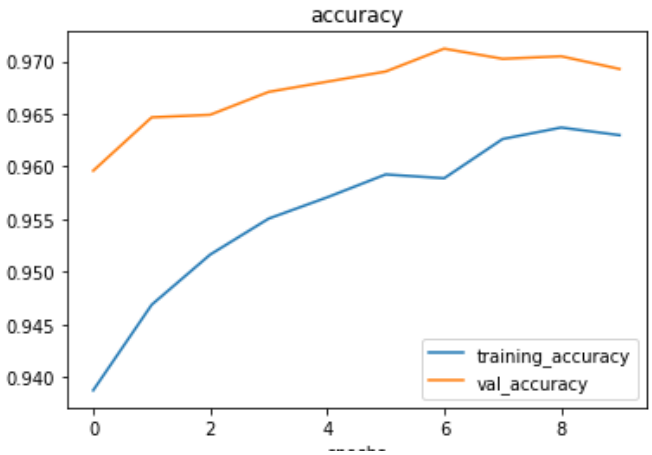
4.1.14 Percobaan RMSprop 1-b

Pada percobaan ini dilakukan *fine tune* pada model hasil percobaan RMSprop 1-a. *Parameter* yang digunakan pada percobaan ini antara lain learning rate 1e-5 (0,00001), ukuran gambar 128x128 dan *optimizer* RMSprop. Berikut adalah hasilnya.



Gambar 4. 42 Grafik *loss* percobaan RMSprop 1-b

Pada gambar 4.42 dapat dilihat bahwa seiring bertambahnya *epoch* nilai *loss* turun hingga *epoch* terakhir (ke-10) dan mulai konvergen pada *epoch* ke-8. dengan nilai akhir *training loss* 0,1 dan *validation loss* 0,097.

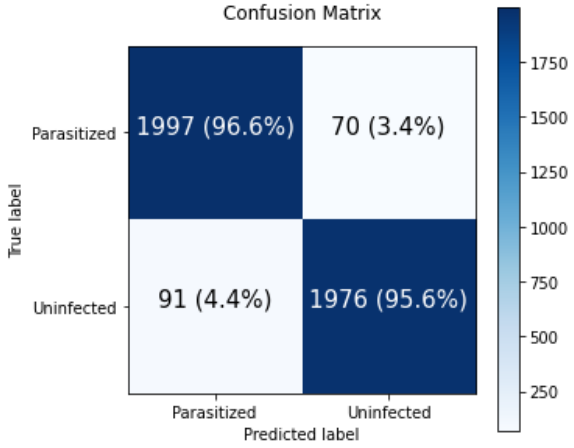


Gambar 4. 43 Grafik akurasi percobaan RMSprop 1-b

Pada gambar 4.43 dapat dilihat bahwa nilai akurasi terus bertambah seiring bertambahnya *epoch*. Untuk akurasi *training* mulai dari nilai 0,938 dan pada *epoch* ke-10 mecapai nilai 0,963, sedangkan pada *val_accuracy* mulai dari nilai 0,959 dan pada epoch ke-10 mencapai nilai 0,969. Selisih antara *training_accuracy* dan *val-_accuracy* kurang lebih 0,006. Hasil akhir *training* dapat dilihat pada tabel 4.28

Tabel 4. 28 Hasil *training* Percobaan RMSprop 1-b

Train loss	*Training accuracy*	*Validation loss*	*Validation accuracy*
0,1041	0,963	0,0979	0,9693



Gambar 4. 44 *Confusion Matrix* percobaan RMSprop 1-b

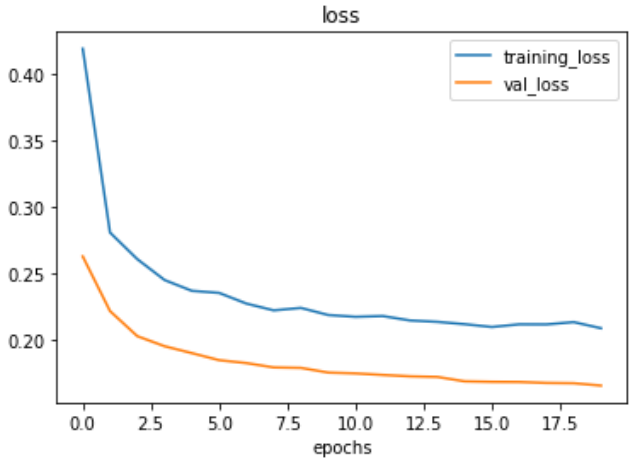
Pada gambar 4.44 dapat dilihat hasil prediksi model. Untuk kelas *parasitized* terprediksi benar sejumlah 1997 citra (96,6%) dari 2067 citra, sedangkan untuk kelas *uninfected* terprediksi benar sejumlah 1976 citra (95,6%) dari 2067 citra. Dari hasil *confusion matrix* model relatif cukup baik dalam melakukan generalisasi. Hal ini dapat dilihat dari selisih hasil prediksi yang relatif kecil (selisih < 4%), *Uninfected* 95,6% dan *Parasitized* 96,6%. Untuk F1-score dapat dilihat pada tabel 4.29

Tabel 4. 29 F1-score percobaan RMSprop 1-b

Accuracy	*Precision*	*Recall*	*F1-score*
0,961	0,9611	0,961	0,961

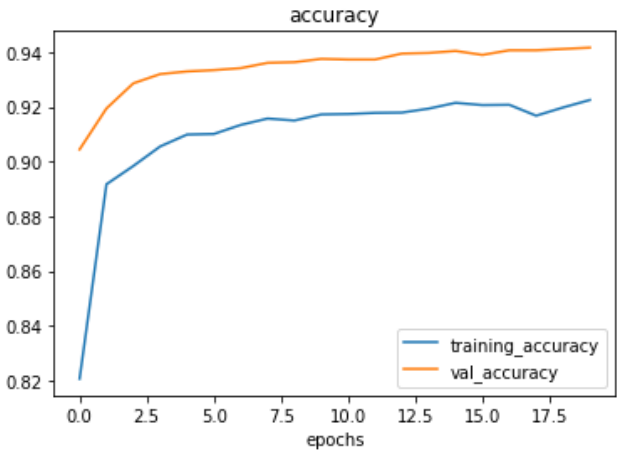
4.1.15 Percobaan RMSprop 2-a

Parameter yang digunakan pada percobaan ini antara lain learning rate 1e-4 (0,0001), *epoch* berjumlah 20, ukuran gambar 128x128 dan optimizer RMSprop. Berikut adalah hasilnya.



Gambar 4. 45 Grafik *loss* percobaan RMSprop 2-a

Pada gambar 4.45 dapat dilihat bahwa nilai *loss* turun seiring bertambahnya *epoch* dan mulai konvergen pada *epoch* ke-10, untuk *epoch* berikutnya hanya mengalami sedikit penurunan hingga *epoch* ke-20.



Gambar 4. 46 Grafik akurasi RMSprop 2-a

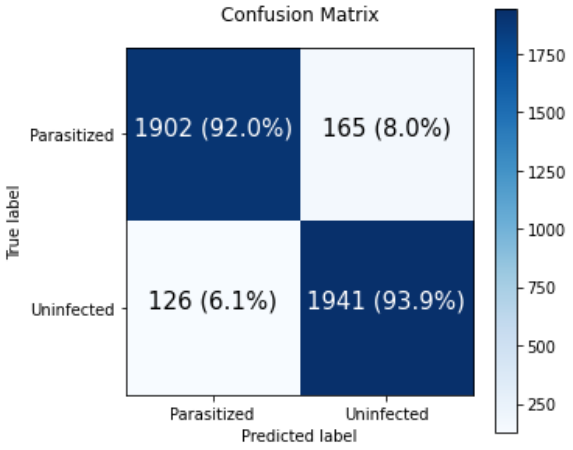
Pada gambar 4.46 dapat dilihat bahwa nilai akurasi terus bertambah seiring bertambahnya *epoch*. Pada akurasi *training* mulai dari nilai 0,82 dan saat *epoch* ke-

20 mecapai nilai 0,926, sedangkan *val_accuracy* mulai dari nilai 0,90 dan pada epoch ke-20 mencapai nilai 0,941. Selisih antara *training_accuracy* dan *val-_accuracy* kurang lebih 0,019. Hasil akhir *training* dapat dilihat pada tabel 4.26

Tabel 4. 30 Hasil *training* percobaan RMSprop 2-a

Train loss	*Training accuracy*	*Validation loss*	*Validation accuracy*
0,2083	0,9226	0,1652	0,9417

Untuk *confusion matrix* dapat dilihat pada gambar berikut.



Gambar 4. 47 *Confusion Matrix* percobaan RMSprop 2-a

Pada gambar 4.47 dapat dilihat hasil prediksi model. Untuk kelas *parasitized* terprediksi benar sejumlah 1902 citra (92,0%) dari 2067 citra, sedangkan untuk kelas *uninfected* terprediksi benar sejumlah 1941 citra (93,9%) dari 2067 citra. Dari hasil *confusion matrix* model relatif cukup baik dalam melakukan generalisasi. Hal ini dapat dilihat dari selisih hasil prediksi yang relatif kecil (selisih < 4%), *Uninfected* 93,9% dan *Parasitized* 92%. Untuk F1-score dapat dilihat pada tabel 4.31

Tabel 4. 31 F1-score percobaan RMSprop 2-a

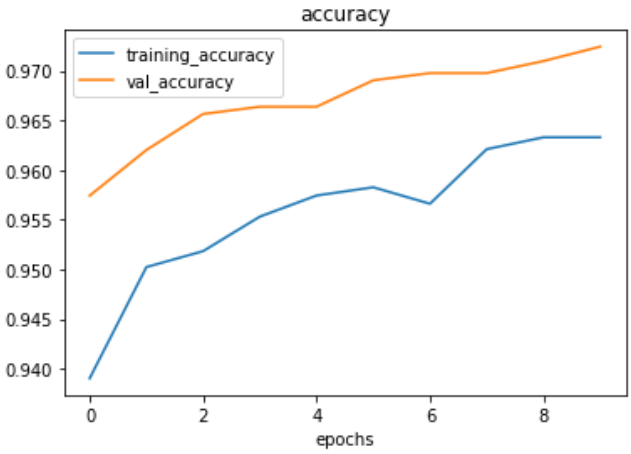
Accuracy	*Precision*	*Recall*	*F1-score*
0,9296	0,9297	0,9296	0,9296

4.1.16 Percobaan RMSprop 2-b

Pada percobaan ini dilakukan *fine tune* pada model hasil percobaan RMSprop 2-a. *Parameter* yang digunakan pada percobaan ini antara lain learning rate 1e-5 (0,00001), ukuran gambar 128x128 dan *optimizer* RMSprop. Berikut adalah hasilnya.

Gambar 4. 48 Grafik *loss* percobaan RMSprop 2-b

Pada gambar 4.48 dapat dilihat bahwa seiring bertambahnya *epoch* nilai *loss* turun hingga *epoch* terakhir (ke-10), namun titik konvergen belum dapat ditentukan karena tren turunya nilai loss masih terus berlanjut hingga *epoch* ke-10 dengan nilai akhir *training loss* 0,10 dan *validation loss* 0,086.

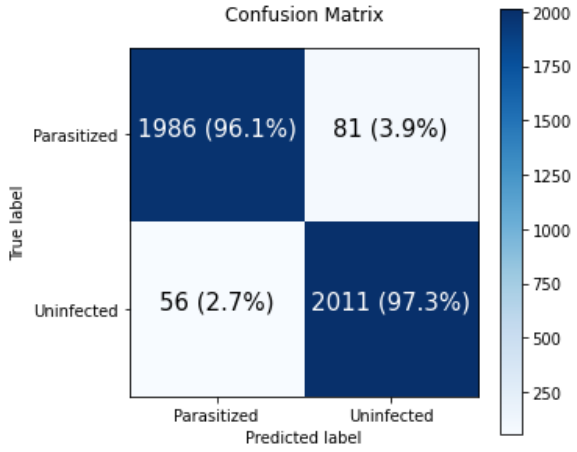


Gambar 4. 49 Grafik akurasi percobaan RMSprop 2-b

Pada gambar 4.49 dapat dilihat bahwa nilai akurasi terus bertambah seiring bertambahnya *epoch*. Untuk akurasi *training* mulai dari nilai 0,939 dan pada *epoch* ke-10 mecapai nilai 0,963, sedangkan pada *val_accuracy* mulai dari nilai 0,957 dan pada epoch ke-10 mencapai nilai 0,972. Selisih antara *training_accuracy* dan *val-_accuracy* kurang lebih 0,009. Hasil akhir *training* dapat dilihat pada tabel 4.32

Tabel 4. 32 Hasil *training* Percobaan RMSprop 2-b

Train loss	*Training accuracy*	*Validation loss*	*Validation accuracy*
0,1026	0,9633	0,0861	0,9724



Gambar 4. 50 *Confusion Matrix* percobaan RMSprop 2-b

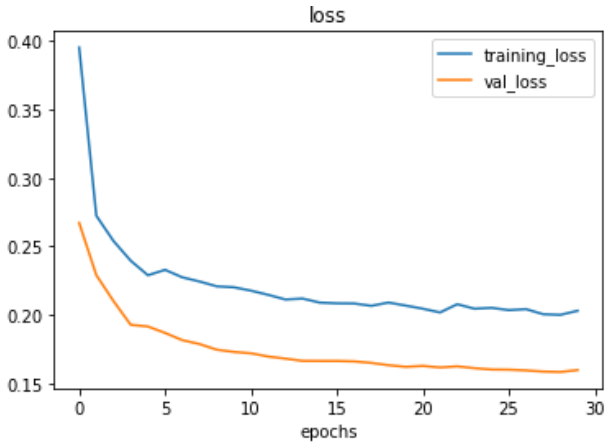
Pada gambar 4.50 dapat dilihat hasil prediksi model. Untuk kelas *parasitized* terprediksi benar sejumlah 1986 citra (96,1%) dari 2067 citra, sedangkan untuk kelas *uninfected* terprediksi benar sejumlah 2046 citra (97,3%) dari 2067 citra. Dari hasil *confusion matrix* model relatif cukup baik dalam melakukan generalisasi. Hal ini dapat dilihat selisih hasil prediksi yang relatif kecil (selisih < 4%), *Uninfected* 97,3% dan *Parasitized* 96,1%. Untuk F1-score dapat dilihat pada tabel 4.33

Tabel 4. 33 F1-score percobaan RMSprop 2-b

Accuracy	*Precision*	*Recall*	*F1-score*
0,9668	0,9669	0,9668	0,9668

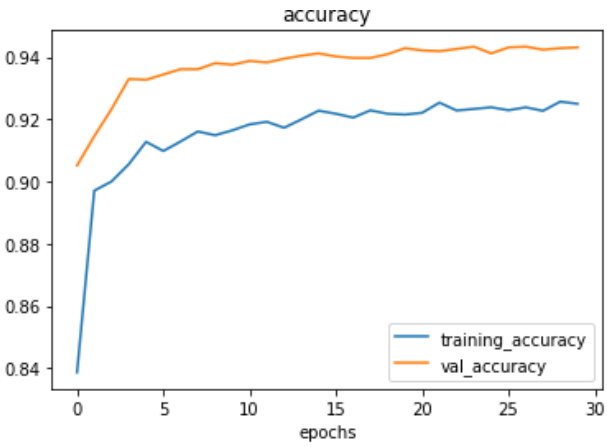
4.1.17 Percobaan RMSprop 3-a

Parameter yang digunakan pada percobaan ini antara lain learning rate 1e-4 (0,0001), *epoch* berjumlah 30, ukuran gambar 128x128 dan optimizer RMSprop. Berikut adalah hasilnya.



Gambar 4. 51 Grafik *loss* percobaan RMSprop 3-a

Pada gambar 4.51 dapat dilihat bahwa nilai *loss* turun seiring bertambahnya *epoch* dan mulai konvergen pada *epoch* ke-20, untuk *epoch* berikutnya hanya mengalami sedikit penurunan hingga *epoch* ke-30.



Gambar 4. 52 Grafik akurasi RMSprop 3-a

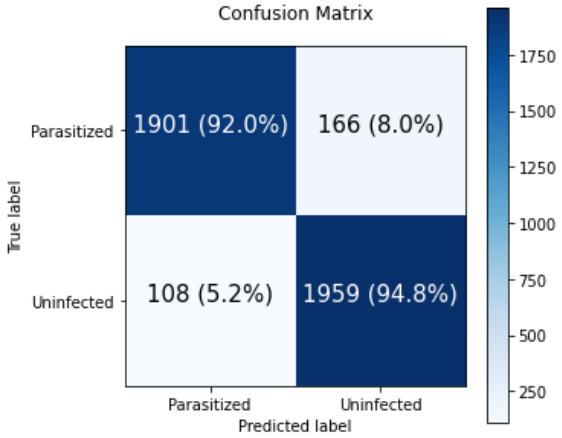
Pada gambar 4.52 dapat dilihat bahwa nilai akurasi terus bertambah seiring bertambahnya *epoch*. Pada akurasi *training* mulai dari nilai 0,83 dan saat *epoch* ke-30 mecapai nilai 0,925, sedangkan *val_accuracy* mulai dari nilai 0,90 dan pada

epoch ke-30 mencapai nilai 0,943. Selisih antara *training_accuracy* dan *val-_accuracy* kurang lebih 0,018. Hasil akhir *training* dapat dilihat pada tabel 4.34

Tabel 4. 34 Hasil *training* percobaan RMSprop 3-a

Train loss	*Training accuracy*	*Validation loss*	*Validation accuracy*
0,2029	0,925	0,1596	0,9432

Untuk *confusion matrix* dapat dilihat pada gambar berikut.



Gambar 4. 53 *Confusion Matrix* percobaan RMSprop 3-a

Pada gambar 4.53 dapat dilihat hasil prediksi model. Untuk kelas *parasitized* terprediksi benar sejumlah 1901 citra (92,0%) dari 2067 citra, sedangkan untuk kelas *uninfected* terprediksi benar sejumlah 1959 citra (94,8%) dari 2067 citra. Dari hasil *confusion matrix* model relatif cukup baik dalam melakukan generalisasi. Hal ini dapat dilihat selisih hasil prediksi yang relatif kecil (selisih < 4%), *Uninfected* 92,% dan *Parasitized* 94,8%. Untuk F1-score dapat dilihat pada tabel 4.35

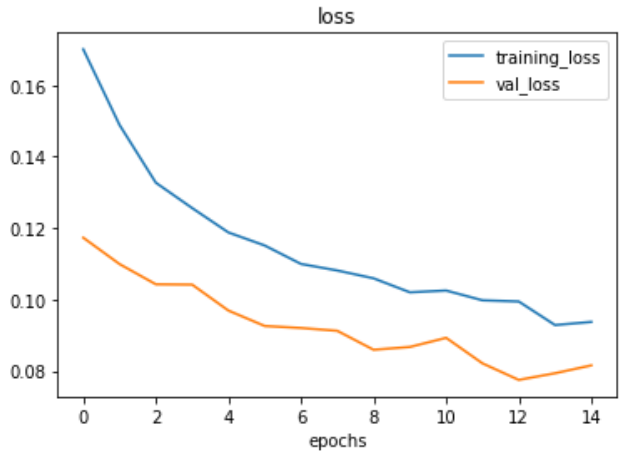
Tabel 4. 35 F1-score percobaan RMSprop 3-a

Accuracy	*Precision*	*Recall*	*F1-score*
0,9337	0,934	0,9337	0,9337

4.1.18 Percobaan RMSprop 3-b

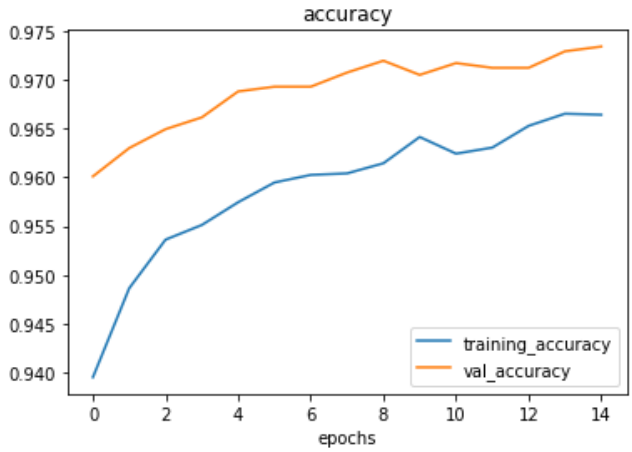
Pada percobaan ini dilakukan *fine tune* pada model hasil percobaan RMSprop 3-a. *Parameter* yang digunakan pada percobaan ini antara lain learning

rate 1e-5 (0,00001), ukuran gambar 128x128 dan *optimizer* RMSprop. Berikut adalah hasilnya.



Gambar 4. 54 Grafik *loss* percobaan RMSprop 3-b

Pada gambar 4.54 dapat dilihat bahwa seiring bertambahnya *epoch* nilai *loss* turun hingga *epoch* terakhir (ke-15) dan mulai konvergen pada *epoch* ke-9. dengan nilai akhir *training loss* 0,093 dan *validation loss* 0,0816.



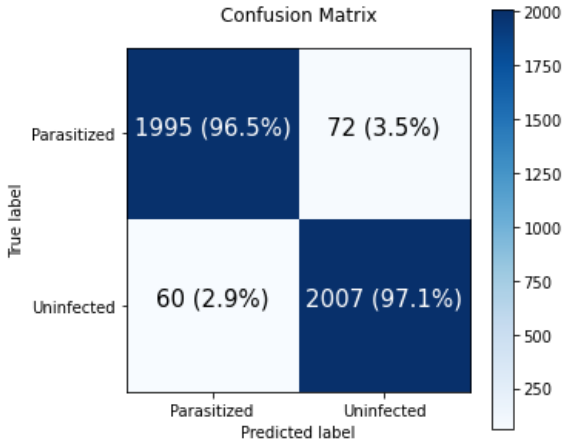
Gambar 4. 55 Grafik akurasi percobaan RMSprop 3-b

Pada gambar 4.55 dapat dilihat bahwa nilai akurasi terus bertambah seiring bertambahnya *epoch*. Untuk akurasi *training* mulai dari nilai 0,939 dan pada *epoch* ke-15 mecapai nilai 0,966, sedangkan pada *val_accuracy* mulai dari nilai 0,96 dan

pada epoch ke-10 mencapai nilai 0,973. Selisih antara *training_accuracy* dan *val-_accuracy* kurang lebih 0,007. Hasil akhir *training* dapat dilihat pada tabel 4.36

Tabel 4. 36 Hasil *training* Percobaan RMSprop 3-b

Train loss	*Training accuracy*	*Validation loss*	*Validation accuracy*
0,0937	0,9664	0,0816	0,9734



Gambar 4. 56 *Confusion Matrix* percobaan RMSprop 3-b

Pada gambar 4.56 dapat dilihat hasil prediksi model. Untuk kelas *parasitized* terprediksi benar sejumlah 1995 citra (96,5%) dari 2067 citra, sedangkan untuk kelas *uninfected* terprediksi benar sejumlah 2046 citra (97,1%) dari 2067 citra. Dari hasil *confusion matrix* model relatif cukup baik dalam melakukan generalisasi. Hal ini dapat dilihat selisih hasil prediksi yang relatif besar (selisih < 4%), *Uninfected* 96,5% dan *Parasitized* 97,1%. Untuk F1-score dapat dilihat pada tabel 4.37

Tabel 4. 37 F1-score percobaan RMSprop 3-b

Accuracy	*Precision*	*Recall*	*F1-score*
0,968	0,968	0,968	0,968

4.2 Perbandingan Hasil Percobaan

Pada bagian ini akan dibandingkan hasil dari percobaan (model). Setiap percobaan menghasilkan satu model dengan total delapan belas model.

Parameter yang digunakan untuk membandingkan model adalah hasil *training* dan perbandingan nilai F1-score.

4.2.1 Perbandingan Hasil *Training*

Parameter yang digunakan pada perbandingan hasil *training* antara lain *training loss, training accuracy, validation loss, dan validation accuracy.* Hasil *training* dapat dilihat pada tabel 4.38

Tabel 4. 38 Hasil *training*

No	Percobaan	Train loss	Train acc	Val loss	Val acc
1	SGD 1a	0,2333	0,9107	0,1932	0,9284
2	SGD 1b	0,2094	0,9213	0,1633	0,9395
3	SGD 2a	0,3185	0,8765	0,2714	0,9006
4	SGD 2b	0,2557	0,9053	0,2082	0,9272
5	SGD 3a	0,3057	0,8804	0,2664	0,9001
6	SGD 3b	0,2427	0,9081	0,2009	0,9286
7	ADAM 1a	0,1912	0,9287	0,1627	0,9448
8	ADAM 1b	0,1015	0,9638	0,0823	0,9695
9	ADAM 2a	0,2027	0,9242	0,1629	0,9422
10	ADAM 2b	0,1004	0,9632	0,086	0,9702
11	ADAM 3a	0,201	0,9244	0,1571	0,9456
12	ADAM 3b	0,0899	0,9678	0,0834	0,9698
13	RMSProp 1a	0,1917	0,928	0,1471	0,9482
14	RMSProp 1b	0,1041	0,963	0,0979	0,9693
15	RMSProp 2a	0,2083	0,9226	0,1652	0,9417
16	RMSProp 2b	0,1026	0,9633	0,0861	0,9724
17	RMSProp 3a	0,2029	0,925	0,1596	0,9432
18	RMSProp 3b	0,0937	0,9664	0,0816	0,9734

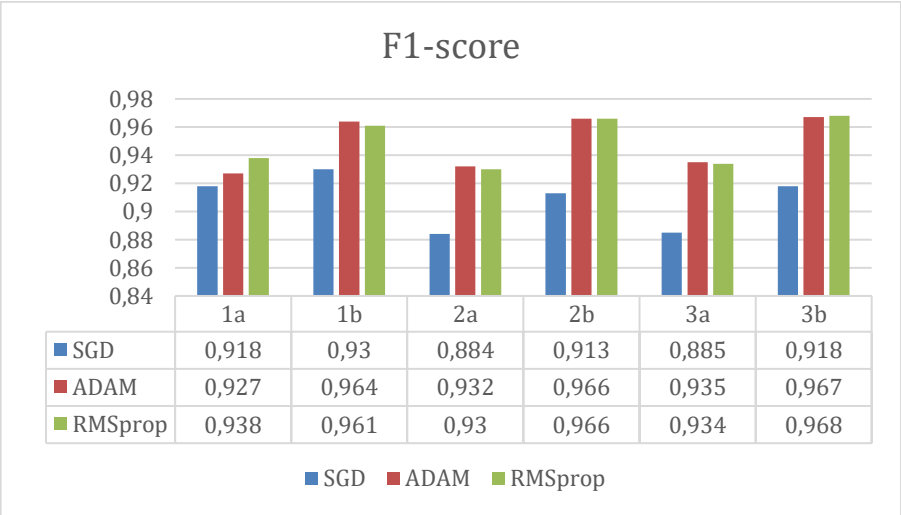
Pada tabel 4.38 dapat dilihat bahwa model dengan *val_loss* dan *val-_accuracy* tertinggi terdapat pada percobaan RMSprop 3-b dengan nilai 0,0816 dan 0,9734.

4.2.2 Perbandingan Hasil Prediksi

Parameter yang digunakan untuk membandingkan hasil prediksi adalah *F1-score* dan akurasi. Akurasi didapat dari merata-rata hasil prediksi model untuk dua kelas (*uninfected* dan *parasitized*) pada tes data. Berikut adalah hasilnya.

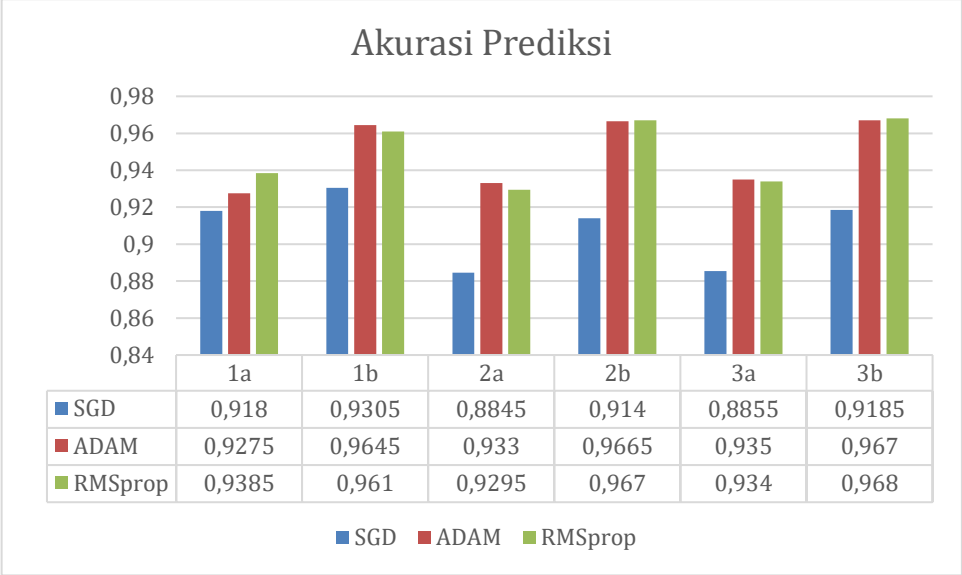
Tabel 4. 39 Hasil prediksi

Percobaan	Parasitized	Uninfected	Selisih	Accuracy	F1-score
SGD 1a	0,925	0,911	0,014	0,918	0,918
SGD 1b	0,931	0,93	0,001	0,9305	0,93
SGD 2a	0,887	0,882	0,005	0,8845	0,884
SGD 2b	0,915	0,913	0,002	0,914	0,913
SGD 3a	0,905	0,866	0,039	0,8855	0,885
SGD 3b	0,929	0,908	0,021	0,9185	0,918
ADAM 1a	0,888	0,967	0,079	0,9275	0,927
ADAM 1b	0,956	0,973	0,017	0,9645	0,964
ADAM 2a	0,918	0,948	0,03	0,933	0,932
ADAM 2b	0,961	0,972	0,011	0,9665	0,966
ADAM 3a	0,913	0,957	0,044	0,935	0,935
ADAM 3b	0,944	0,99	0,046	0,967	0,967
RMSProp 1a	0,919	0,958	0,039	0,9385	0,938
RMSProp 1b	0,966	0,956	0,01	0,961	0,961
RMSProp 2a	0,92	0,939	0,019	0,9295	0,93
RMSProp 2b	0,961	0,973	0,012	0,967	0,966
RMSProp 3a	0,92	0,948	0,028	0,934	0,934
RMSProp 3b	0,965	0,971	0,006	0,968	0,968



Gambar 4. 57 Grafik perbandingan F1-score

Pada gambar 4.57 dapat dilihat bahwa terdapat enam model yang memiliki F1-score tertinggi yaitu ADAM 1-b, RMSprop 1-b, ADAM 2-b, RMSprop 2-b, Adam 3-b, dan RMSprop 3-b. Masing-masing bernilai 0,964, 0,961, 0,966, 0,966, 0,967, dan 0,968. Untuk grafik perbandingan akurasi dapat dilihat pada gambar



Gambar 4. 58 Grafik perbandingan hasil prediksi

Pada gambar 4.58 dapat dilihat bahwa terdapat enam model yang memiliki akurasi tertinggi yaitu ADAM 1-b, RMSprop 1-b, ADAM 2-b, RMSprop 2-b, Adam 3-b, dan RMSprop 3-b. Masing-masing bernilai 0,964, 0,961, 0,966, 0,967, 0,967, dan 0,968. Dari enam model yang memiliki performa terbaik, baik dari nilai F1-score dan akurasi, akan dipilih satu model yang akan digunakan pada aplikasi web.

Karena nilai akurasi dan F1-score dari enam model kurang lebih sama, maka akan digunakan nilai selisih dari hasil prediksi dari dua kelas *uninfected* dan *parasitized* untuk menentukan model yang akan dipilih. Model yang dipilih adalah model RMSprop 3-b, karena selain memiliki akurasi yang tinggi, model RMSprop 3-b juga model yang paling baik dalam melakukan generalisasi. Nilai selisih hasil prediksi dari dua kelas *uninfected* dan *parasitized* adalah 0,006, paling kecil dari lima model yang lain. ADAM 1-b = 0,017, RMSprop 1-b = 0,01, ADAM 2-b = 0,011, RMSprop 2-b = 0,012, dan ADAM 3-b = 0,046.

4.2 Aplikasi Web

Aplikasi web dibuat menggunakan *package* Streamlit dan dengan bahasa pemrograman python. Selain streamlit, *library* lain yang diperlukan untuk membuat aplikasi web adalah Tensorflow, Numpy, Math, dan Pillow. Streamlit memungkinkan pemuatan aplikasi web berbasis python dengan cepat. Proses prediksi dan *image processing* dapat dilakukan secara langsung pada aplikasi web sehingga tidak diperlukan server tambahan untuk *deploy* model secara terpisah. Streamlit berjalan sama seperti python (berjalan secara procedural) yang berarti aplikasi web akan menjalankan kode dari awal hinga akhir. Aplikasi bekerja dengan cara mengambil *file* gambar dari penyimpanan *smartphone*, lalu *user* menekan tombol *predict* untuk memprediksi gambar lalu menampilkan hasil prediksi berupa presentase atau *confidence level* bahwa gambar yang telah diunggah merupakan sel yang *parasitized* atau *uninfected.* Kode *snippet* untuk aplikasi web dapat dilihat pada gambar berikut.

```
1   upload_file=st.file_uploader("Upload Cell Images",type=
    ['png,'jpg,'jpeg])
2   Generate_pred=st.button("Predict")
```

Gambar 4. 59 Kode untuk membuat tombol *upload file* dan *predict*

`st.file_uploader` berfungsi untuk menambahkan *input widget* untuk menunggah gambar pada aplikasi web, sedangkan `st.button` berfungsi untuk menambahkan *input widget* berupa tombol.

```
1   if upload_file:
2       Image=Image.open(upload_file)
3       With st.expander('Cell Image', expanded = True):
4           st.image(image, use_column_width = True)
```

Gambar 4. 60 Kode yang dijalankan ketika tombol upload ditekan

Ketika tombol *upload* ditekan oleh *user*, kode pada gambar 4.60 akan dijalankan untuk menampilkan gambar yang telah dipilih *user*. `st.expander` diperlukan agar tampilan gambar disesuaikan oleh ukuran layar dari *device* yang digunakan oleh *user*.

```
@st.cache(allow_output_mutation=True)
def load_models(model_name):
    model = tf.keras.models.load_models(model_name)
    return model

model = load_models('efficientnet_model_RMSProp3_ft.h5')
```

Gambar 4. 61 Fungsi untuk memuat model

Pada gambar 4.60 terdapat kode `@st.cache` yang memungkinkan mekanisme *cache* pada fungsi `load_models`. Hal ini berfungsi ketika prediksi selesai dan *user* ingin melakukan prediksi baru, aplikasi web tidak perlu memuat ulang model.

```
def import_n_pred(image_data, model):
    size=(128,128)
    image=ImageOps.fit(image_data,size,Image.ANTIALIAS)
    img=np.asarray(image)
    reshape=img[np.newaxis,…]
    pred=model.predict(image)
    return pred
```

Gambar 4. 62 Fungsi untuk *processing* gambar dan prediksi

Fungsi `import_n_pred` pada gambar 4.62 berfungsi untuk merubah input gambar menjadi 128x128, lalu merubah array dari input gambar menjadi tiga dimensi pada kode `reshape = img[np.newaxis,...]` kemudian dilakukan prediksi dengan kode `pred = model.predict(reshape)`.

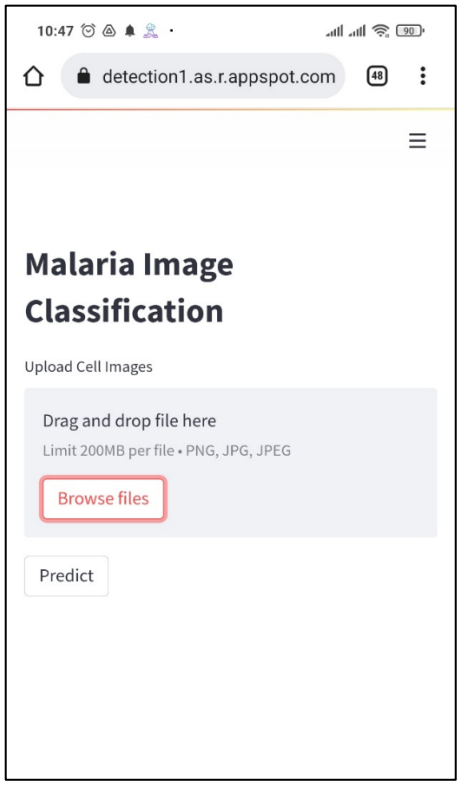
```
if Generate_pred:
    pred=import_n_pred(image,model)
    labels=['Parasitized', 'Uninfected']
    prediction_result=math.floor(pred[0,np.argmax(pred)] * 10000)/10000
    confidence_level="{0:.4f}".format(prediction_result)
```

Gambar 4. 63 Kode yang dijalankan ketika tombol *predict* ditekan

Kode pada gambar 4.63 berfungsi untuk memanggil fungsi `import_n_pred` kemudian mengubah nilai output dari prediksi menjadi empat digit dibelakang koma sebagai *confidence level*. Kode untuk menampilkan *confidence level* hasil prediksi dapat dilihat pada gambar 4.64

```
1  St.title('Prediction of image is {} with {} confidence".format
   (labels[np.argmax(pred)], confidence_level))
```

Gambar 4. 64 Kode untuk menampilkan *confidence level*



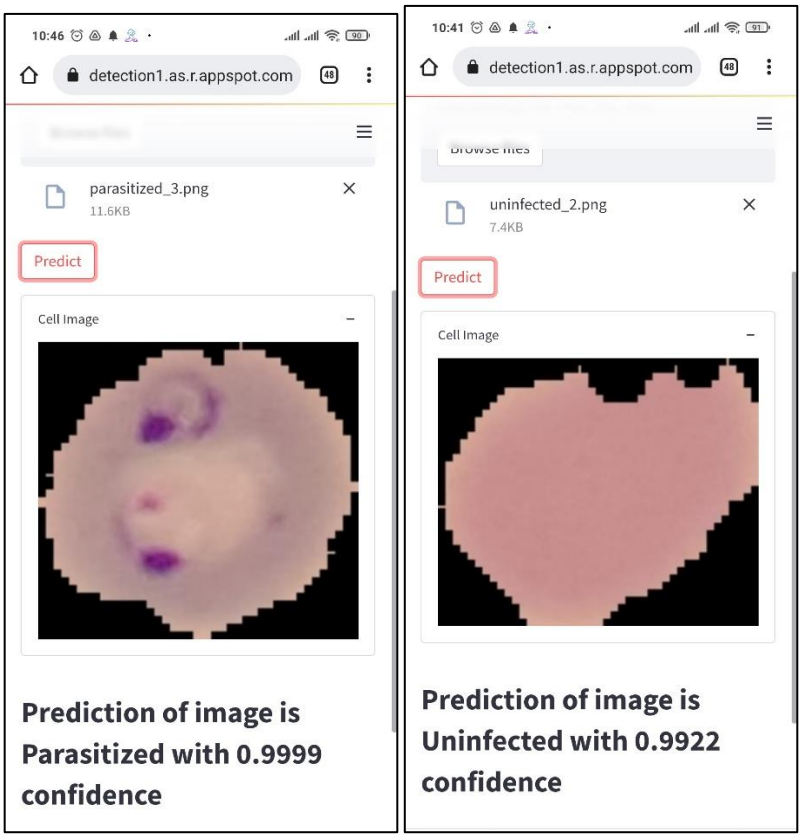
Gambar 4. 65 Tampilan antarmuka aplikasi web

4.3 Pengujian Aplikasi

Pengujian aplikasi dilakukan dengan dua cara, cara pertama adalah dengan menguji fitur prediksi dan yang melakukan survei pada sepuluh pengguna *smartphone*.

4.3.1 Pengujian Fitur Prediksi

Pengujian dilakukan dengan mengambil sampel gambar secara acak untuk masing masing kelas *parastized* dan *uninfected* dari data tes. Kemudian dilakukan prediksi pada aplikasi untuk mengetahui bahwa aplikasi dapat berfungsi dengan baik atau tidak.



Gambar 4. 66 Sampel prediksi

Hasil dari sampel prediksi menunjukan fitur prediksi dari aplikasi web dapat berfungsi dengan baik dan hasil dari prediksi memiliki akurasi yang tinggi tidak jauh berbeda dengan hasil prediksi model dengan tes data.

4.3.2 Survei *User Smarphone*

Survei dilakukan dengan cara setiap responden diberikan tautan folder yang berisi gambar sel *parasitized* dan *uninfected. User* diminta untuk mengunduh satu gambar sel *parasitized* dan *uninfected* dari tautan yang telah diberikan. Setiap *user* membuka aplikasi web pada *smartphone* masing-masing. Setiap *user* diberi arahan bagimana cara menggunakan aplikasi. Setelah diberi arahan, setiap *user* melakukan prediksi dengan gambar yang telah diunduh. Terakhir setiap *user* diminta untuk menjawab pertanyaan yang diberikan.

Terdapat empat pertanyaan yang diberikan, berikut adalah pertanyaanya.

1. Bagaimana kecepatan *loading* aplikasi?
2. Apakah fitur prediksi dapat berfungsi dengan baik?

3. Bagimana akurasi hasil prediksi?
4. Apakah aplikasi mudah digunakan?

Hasil survei dapat dilihat pada tabel 4.40 . Kolom baris pertama dengan nomor 1 mewakili pertanyaan pertama, kolom dengan nomor 2 mewakili pertanyaan kedua, kolom dengan nomor 3 mewakili pertanyaan ketiga, kolom dengan nomor 4 mewakili pertanyaan keempat.

Tabel 4. 40 Hasil Survei

Responden	1	2	3	4
User1 (IphoneXr)	cepat	Ya	Akurat	Mudah
User2 (Realme5Pro)	cepat	Ya	Akurat	Mudah
User3 (Samsung A7)	cepat	Ya	Sangat Akurat	Mudah
User4 (AsusMaxProM1)	cepat	Ya	Sangat Akurat	Mudah
User5 (Infinix)	cepat	Ya	Sangat Akurat	Mudah
User6 (Iphone7)	aplikasi tidak dapat berjalan	-	-	-
User7 (RedmiNote4)	cepat	Ya	Sangat Akurat	Mudah
User8 (Mi 10T Pro)	cepat	Ya	Sangat Akurat	Mudah
User9 (Mi 8 Light)	cepat	Ya	Sangat Akurat	Mudah
User10 (Iphone7Plus)	cepat	Ya	Sangat Akurat	Mudah

Setiap *user* memiliki perangkat yang berbeda-beda. Dari sepuluh perangkat yang digunakan, aplikasi web dapat berjalan pada perangkat tersebut kecuali pada perangkat Iphone7. Alasan kenapa aplikasi web tidak berjalan tidak diketahui.

Bab V
Kesimpulan dan Saran

5.1 Simpulan

Model yang memiliki performa terbaik adalah model pada eksperimen RMSprop 3-b dengan performa model saat *training* adalah *training loss* = 0,0937, *validation loss* = 0,0816, *training accuracy* = 0,9664, *validation accuracy* = 0,9734 sedangkan hasil prediksi pada data tes adalah akurasi = 96,8% dan F1-score = 0,968. Waktu yang dibutuhkan untuk menjalankan *training* sebelum *fine tuning* (percobaan dengan label "a") adalah 15 menit dan pada saat *fine tuning* (percobaan dengan label "b") adalah 22,25 menit.

Aplikasi yang telah diuji memiliki kecepatan muat yang relatif cepat berdasarkan pengalaman responden. Aplikasi dapat digunakan dan semua fitur dapat berfungsi dengan baik. Terdapat *smartphone* yang tidak dapat memuat aplikasi web (Iphone7), untuk saat ini alasan mengapa aplikasi tidak dapat dimuat tidak diketahui.

5.2 Saran

Untuk penelitian selanjutnya, jika peneliti ingin meningkatkan performa model, peneliti dapat mengubah ukuran citra menjadi lebih besar misal 224x224. Peneliti dapat membagi dataset menjadi dua, dataset dengan ukuran kecil untuk melatih *base model* dan dataset dengan ukuran besar untuk melakukan *fine tuning*. Peneliti juga dapat mengubah *batch size* dan menerapkan konsep *cyclical learning rate.*

Daftar Pustaka

Alodokter, 2020. *Malaria - Gejala, penyebab, dan mengobati - Alodokter*. [Online] Available at: https://www.alodokter.com/malaria [Accessed 10 Maret 2022].

Bonaccorso, G., 2018. *Machine Learning ALgorithm.* 2nd ed. Birmingham: Packt Publishing.

Brownlee, J., 2017. *A Gentle Introduction to Transfer Learning for Deep Learning.* [Online] Available at: https://machinelearningmastery.com/transfer-learning-for-deep-learning/ [Accessed 12 Maret 2022].

Calderaro, A. et al., 2021. Malaria Diagnosis in Non-Endemic Settings: The European. *Microoorganism*, 9(11), p. 2265.

Christopher, A., 2021. *K-Nearest Neighbor. A comlete explaination of K-NN*. [Online] Available at: https://medium.com/swlh/k-nearest-neighbor-ca2593d7a3c4 [Accessed 11 Maret 2022].

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

Delgado-Ortet, M. et al., 2020. A Deep Learning Approach for Segmentation of Red. *Entropy*, 22(6), p. 657.

Dhuha, N. S., 2018. *Apa yang dimaksud dengan Pengujian perangkat lunak atau SoftwareTesting.*[Online] Available at: https://www.dictio.id/t/ apa-yang-dimaksud-dengan-pengujian-perangkat-lunak-atau- oftware-testing/ 15136 [Accessed 13 Maret 2022].

Docker Inc, 2022. *Docker Overview | Docker Documentation.* [Online] Available at: https://docs.docker.com/get-started/overview/ [Accessed 10 Juni 2022].

Evani,S.,2020.*Diagnosis Malaria - Alomedika.* [Online] Available at:https://www.alomedika.com/ penyakit/penyakitinfeksi/malaria diagnosis [Accessed 10 Maret 2022].

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

Fumo, D., 2017. *Types of Machine Learning Algorithms You Should Know*. [Online] Available at: https://towardsdatascience.com/types-of-machine-

learning-algorithms-you-should-know-953a08248861 [Accessed 11 Maret 2022].

Green, J., 2020. *What is Google Cloud AI Platform.* [Online] Available at: https://towardsdatascience.com/google-cloud-ai-platform-hyper-accessible-ai-machine-learning-cddd8c3348b3 [Accessed 13 Maret 2022].

Gudimella, A. et al., 2017. Deep Reinforcement Learning for Dexterous Manipulation with Concept Network. *arXiv:1709.06977[cs.AI]*.

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

Hamilton, T., 2022. *Application testing tutorials: What it is, How to.* [Online] Available at: https://www.guru99.com /application-testing.html [Accessed 13 Maret 2022].

Han, X. et al., 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,* pp. 770-778.

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

IBM, 2020. *What is Machine Learning?.* [Online] Available at: https://www.ibm.com/cloud/learn/machine-learning [Accessed 10 Maret 2022].

J.Bigelow, S., 2020. *What is Google Cloud Platform (GCP).* [Online] Available at: https://www.techtarget.com/ searchcloudcomputing/ definition/Google-Cloud-Platform [Accessed 12 Maret 2022].

KemenkesRI, 2019. *Profil Kesehatan Indonesia,* s.l.: Kementrian Kesehatan Republik Indonesia.

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

LeCun, Y., Bengio, Y. & Hinton, G., 2015. Deep Learing. *Nature,* 521(7553), pp. 436-444.

LeCun, Y. et al., 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), pp. 2278-2324.

Li, S., 2021. *Streamlit Hands-On: From ero to Your First Awesome Web App.* [Online] Available at: https://towardsdatascience.com/streamlit-hands-on-from-zero-to-your-first-awesome-web-app-2c28f9f4e214 [Accessed 12 Maret 2022].

Lutkevich, B., 2022. *What is Google App Engine?*. [Online] Available at: https://www.techtarget.com/searchaws/definition/Google-App-Engine [Accessed 10 Juni 2022].

Masud, M. et al., 2020. Leveraging Deep Learning Techniques for Malaria Parasite. *Wireless Communications and Mobile Computing,* Volume 2020.

Montalbo, F. J. P. & Alon, A. S., 2021. Empirical Analysis of a Fine-Tuned Deep Convolutional Model in Classifying and Detecting Malaria Parasites from Blood Smears. *KSII Transactions on Internet and Information Systems,* 15(1), pp. 147-165.

Na'imah, S., 2021. *Penyakit Malaria: Gejala,Penyebab, Hingga Pengobatan.* [Online] Available at: https://hellosehat.com/infeksi/infeksi-serangga/ penyakit-malaria/ [Accessed 10 Maret 2022].

paperswithcode, 2019. *CIFAR-100 Benchmark (Image Classification).* [Online] Available at: https://paperswithcode.com/sota/image-classification-on-cifar-100 [Accessed 12 Maret 2022].

Pradana, B. J., 2021. Pemanfaatan Transfer Learning Pada CNN Untuk Pendeteksian Malaria.

PythonInstitute, 2022. *About Python.* [Online] Available at: https://pythoninstitute.org/about- python [Accessed 12 Maret 2022].

Rajan, S., 2021. *Streamlit Web App Build Web Applicaions using Streamlit.* [Online]Available at:https://www.analyticsvidhya.com/blog/2021/06/ build -web-app-instantly-for-machine-learning-usingstreamlit/#:~: text=What %20is%20Streamlit%3F,you%20write%20a%20python%20code. [Accessed 12 Maret 2022].

Reddy, A. S. B. & Juliet, D. S., 2019. *Transfer learning with ResNet-50 for malaria cell-image classification.* Chennai, India, IEEE.

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

softwaretestinghelp, 2022. *Mobile App Testing Tutorials.* [Online] Available at: https://www.softwaretestinghelp.com/beginners-guide-to-mobile-application-testing/[Accessed 13 Maret 2022].

Stack Overflow, 2021. *Stack Overflow Developer Survey 2021*. [Online] Available at: https://insights.stackoverflow.com/survey/2021#most-popular-technologies-language [Accessed 12 Maret 2022].

Statistic Solution, 2022. *What is Linear Regression?*. [Online] Available at: https://www.statisticssolutions.com/free-resources/directory-of-statistical-analyses/what-is-linear-regression/ [Accessed 12 Maret 2022].

Takdirillah, R., 2020. *Apa itu Machine Learning? Beserta Pengertian dan Cara Kerjanya.*[Online] Available at: https://www.dicoding.com/blog/machine-learning-adalah/ [Accessed 11 Maret 2022].

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

Wang, Z. J. et al., 2021. CNN Explainer: Learning Convolutional Neural Networks with Interactive Visualization. *IEEE Transactions on Visualization and Computer Graphics,* 27(2), pp. 1396- 406.

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

WHO, 2020. *World Malaria Report,* s.l.: World Health Organization.

WHO, 2022. *Malaria.* [Online] Available at: https://www.who.int/news-room/fact-sheets/detail/malaria [Accessed 10 Maret 2022].

Zhuang, F. et al., 2020. A comprehensive survey on transfer learning. *Proceedings of IEEE,* 109(1), pp. 43-76.

Lampiran

A. Tutorial *Deployment* Aplikasi

Deployment aplikasi dibagi menjadi dua tahap, tahap pertama adalah menyiapkan file yang diperlukan untuk *deployment.* Tahap kedua adalah melakukan *deployment* pada Google App Engine.

File yang diperlukan untuk melakukan *deployment* adalah Dockerfile, app.yaml, dan requirements.txt. *File* harus diletakan dalam satu folder dengan *source code* dari aplikasi web. Berikut adalah penjelasan dari masing-masing *file*.

Dockerfile

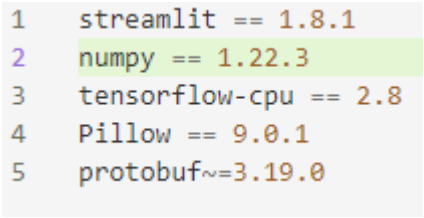
```
1   FROM python:3.8-slim
2
3   ## App engine stuff
4   # Expose port you want your app on
5   EXPOSE 8080
6
7   #Copy Requirements.txt file into app directory
8   COPY requirements.txt app/requirements.txt
9
10  RUN pip install -r app/requirements.txt
11
12  # Create a new directory for app (keep it in its own directory)
13  COPY . /app
14  WORKDIR /app
15
16  # Run
17  ENTRYPOINT ["streamlit", "run","main.py","--server.port=8080", "--server.address=0.0.0.0"]
```

Gambar A. 1 Dockerfile

Dockerfile diperlukan untuk membuat docker *container*, sehingga aplikasi dapat diunggah pada Google App Engine. Baris pertama pada gambar A.1 menunjukan bahwa aplikasi web dibuat dengan menggunakan python. Baris 5 merupakan port yang digunakan untuk server aplikasi web. Baris 8 membuat *directory* baru pada *container* dengan nama app/ lalu melakukan operasi *copy file* requirements.txt. lalu *paste* pada *directory* yang telah dibuat. Baris 10 adalah perintah *install* semua *packages* yang terdapat pada requirements.txt. Baris 13 dan 14, adalah *copy* semua file ke *directory* /app dan membuat /app menjadi *working directory*. Pada baris 17 merupakan *entrypoint* atau perintah yang akan dijalankan ketika *container* pertama kali dijalankan.

Gambar A. 2 *file* app.yaml

File app.yaml diperlukan untuk *deployment* dan *configurasi* VM yang akan digunakan oleh aplikasi web. Pada gambar A.2 baris 7 – 9, berisi *core cpu* yang digunakan, RAM yang digunakan dan ukuran *hardisk* dari VM *instance*.



Gambar A. 3 *File* requirements.txt

Pada gambar A.3, requirements.txt berisi lima *package/library* yang diperlukan untuk membuat aplikasi web. Berikutnya adalah tahapan yang diperlukan untuk melakukan *deployment* aplikasi web pada Google App Engine.

1. *Install* Google Cloud SDK terlebih dahulu
2. Pada konsol Google Cloud Platform, buat projek baru
3. Pada menu navigasi, pergi ke bagian App Engine
4. Pada *dashboard* App Engine, tekan *Create Application*, lalu pilih *Region* untuk *server*
5. Pergi ke Command Line, buka *directory* aplikasi web
6. Jalankan perintah "`gcloud init`" pada Command Line
7. Setelah selesai, jalankan perintah "`gcloud app deploy app.yaml`"
8. Aplikasi web dapat dibuka pada target url yang disediakan.

   Biaya yang diperlukan untuk deploy aplikasi pada GCP bergantung pada spesifikasi VM instance yang digunakan. Pada penelitian ini VM instance yang

digunakan adalah flexible environment (4 core cpu, 4 gb ram, 15 gb cloud storage) dengan biaya 7,19 USD per hari atau setara dengan 107.500 rupiah.

B. Tutorial Docker

Berikut adalah referensi yang dapat digunakan untuk mempelajari docker dan membuat python app docker container

- docker.com “How to containerize your python application”. Link: https://www.docker.com/blog/how-to-dockerize-your-python-applications/
- medium.com “Deploying a streamlit app with docker” by Isha Terdal. Link: https://medium.com/@ishaterdal/deploying-a-streamlit-app-with-docker-db40a8dec84f
- towardsdatasciene.com “Create an awesome streamlit app & deploy it with docker” by Zoumana Keita. Link: https://towardsdatascience.com/create-an-awesome-streamlit-app-deploy-it-with-docker-a3d202a636e8
- freecodecamp.com “The docker handbook – learn docker for beginners” by Farhan Hasin Chowdhury. Link: https://www.freecodecamp.org/news/the-docker-handbook/