PENGARUH HYPERPARAMETER TUNING PADA DETEKSI MALARIA MENGGUNAKAN CNN

PRAKTIK KERJA LAPANGAN

GREGORIUS GUNTUR SUNARDI PUTRA

311810015

TEKNIK INFORMATIKA

FAKULTAS SAINS DAN TEKNOLOGI

UNIVERSITAS MA CHUNG

MALANG

2021

LEMBAR PENGESAHAN

PENGARUH HYPERPARAMMETER TUNING PADA
DETEKSI MALARIA MENGGUNAKAN CNN

Oleh:
GREGORIUS GUNTUR SUNARDI PUTRA
311810015

dari
TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS MA CHUNG

Dosen Pembimbing

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

Dekan Fakultas Sains dan Teknologi

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

KATA PENGANTAR

Puji dan syukur dipanjatkan kepada Tuhan Yang Maha Esa atas karunia-Nya, laporan Praktik Kerja Lapangan (PKL) ini, dapat dibuat dan diselesaikan dengan baik. Laporan dengan judul "Pengaruh Hyperparameter Tuning Pada Deteksi Malaria Menggunakan CNN" disusun berdasarkan apa yang telah dilakukan pada saat praktik kerja lapangan selama 4 bulan lebih 2 minggu mulai 13 September 2021 hingga 17 Desember 2021 di Pusat Studi Artificial Intellegence in Digital Images and Technoprenuership (AiDiTech).

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

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

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

Malang, 20 Desember 2021

Gregorius Guntur Sunardi Putra

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

Bab I
Pendahuluan

1.1 Latar Belakang

Malaria merupakan penyakit mematikan di berbagai negara, yang disebabkan oleh parasit darah *Plasmodium*. *Plasmodium* ada berbagai jenis seperti *Plasmodium vivax, Plasmodium ovale,* dan *Plasmodium falciparum*. *Plasmodium* disebarkan oleh gigitan nyamuk *Anopheles* betina dan dapat menular melalui transfusi darah atau penggunaan jarum suntik secara bersamaan (Poostchi, 2018). Gejala yang sering terjadi seperti demam, sakit kepala, dan pucat karena kekurangan darah. Komplikasi yang terjadi dapat mengakibatkan gangguan pernafasan sampai mengalami kematian. Di dunia terdapat lebih dari 200 juta kasus Malaria di tahun 2019 dan menjadi endemi di 89 negara (WHO, 2020). Malaria menyumbang lebih 400 ribu kematian dengan 67% di antaranya masih berusia balita. Pada tahun yang sama Indonesia memiliki lebih 250 ribu kasus Malaria dan 80% di antaranya berada di Papua. Data selama satu dekade terakhir menunjukkan penurunan penderita Malaria baik di Indonesia maupun di dunia. Akan tetapi, kewaspadaan akan Malaria harus tetap dijaga dengan melakukan deteksi dini untuk mengurangi kasus penderita Malaria dan mencegah risiko kematian.

Berdasarkan protokol WHO, dalam mendeteksi Malaria dapat dilakukan dengan menggunakan pemeriksaan *Peripheral Blood Morphology* atau *Blood Smear*. *Blood Smear* mengevaluasi sel darah merah, sel darah putih, dan trombosit (Bian, 2005). Sampel darah pasien akan diwarnai dengan *flourscense* dan kemudian dilihat menggunakan mikroskop untuk melihat adanya parasit *Plasmodium* dalam darah. Para teknisi yang melakukan pemeriksaan mikroskop dinamakan Phlebotomist. Dengan jumlah Phlebotomist yang sedikit sehingga para teknisi mengalami kelelahan fisik yang mengakibatkan kesalahan analisis dan interpretasi saat melakukan pemeriksaan *Blood Smear*. Untuk mengatasi kesalahan interpretasi, para peneliti berlomba-lomba mencari alat bantu yang memudahkan tenaga medis melakukan deteksi dini malaria menggunakan perkembangan teknologi. Salah satu

perkembangannya, dengan menggunakan Teknologi *Deep Learning* untuk mendiagnosis Malaria berbasis citra tekstur warna pada sampel darah menggunakan protokol *Blood Smear*.

Convolutional Neural Network (CNN) merupakan model *Deep Learning* yang digunakan dalam mendeteksi Malaria. Kemampuan CNN dalam mengelola data yang besar dapat mendeteksi malaria pada citra *Blood Smear* dengan baik (Rajaraman, et al., 2019). Model dari CNN akan mengklasifikasikan citra *Blood Smear* ke dalam 2 kelas, *Uninfected* dan *Paracitized*. Dalam penelitian sebelumnya, sudah ada berbagai macam arsitektur CNN yang digunakan dalam memprediksi Malaria. Seperti Arsitektur LeNet5 yang menghasilkan akurasi sebesar 95% (Harahab, et al., 2021) dan Arsitektur MM-RestNet yang memiliki akurasi 98% pada gambar bercitra rendah (Pattanaik, et al., 2020). Sedangkan Arsitektur GoogleNet memiliki performa akurasi sebesar 98,13% (Yuhang, et al., 2017) akan tetapi nilai *error rate* yang tinggi sebesar 25.86%. Sehingga dibutuhkan studi untuk meningkatkan performa model tanpa memperbesar nilai *error*.

Studi tentang peningkatan performa model CNN sudah banyak dilakukan sebelumnya. Peningkatan performa dapat dilakukan dengan melakukan tuning pada preprocessing (Swastika, et al., 2021). Dengan menggunakan preprocessing, model menjadi lebih akurat 0,5% dibandingkan tanpa preprocessing. Selain itu, penggunaan *Transfer Learning* juga dapat mempengaruhi performa model. Dengan melakukan tuning pada hyperparameter dapat meningkatkan performa dari model, cara ini dinamakan Hyperparameter Tuning. Dalam studi Jaiswal melakukan Hyperparameter Tuning pada nilai filter dengan menggunakan *Genetic Algorithm* menghasilkan akurasi terbaik sebesar 95,17% pada data testing (Jaiswal, et al., 2020). Pada studi (Suriya, et al., 2019) melakukan tuning pada hyperparameter *epochs*, dan *optimizers* pada Deep CNN menghasilkan performa akurasi sebesar 98%. Pencarian hyperparameter pada studi Zhao dengan menggunakan citra beresolusi rendah menghasilkan hyperparameter terbaik dengan *optimizer* SGD, *learning-rate* 10^{-5}, dan *batch-size 64* (Zhao, et al., 2020). Pada studi ini, mencoba melakukan pencarian hyperparameter menggunakan *Grid Search* untuk mendapatkan hyperparameter terbaik pada arsitektur costum Rajaraman, dan BaselineNet.

1.2 Identifikasi Masalah

Berdasarkan latar belakang yang sudah dijelaskan di atas, dapat di identifikasikan masalah yaitu akurasi pada deteksi Malaria kurang optimal yang disebabkan oleh hyperparameter yang digunakan bukan yang terbaik pada arsitektur CNN yang digunakan.

1.3 Batasan Masalah

Batasan masalah dari pengerjaan penelitian ini, ialah sebagai berikut.

1. Penelitian dilakukan dengan menggunakan data citra *Blood Smear* dari Lister Hill National Center for Biomedical Communications.
2. Arsitektur yang digunakan merupakan Arsitektur Rajaraman dan BaselineNet.
3. Penelitian ini menggunakan GPU- High Ram pada Google Colab Pro.
4. Parameter yang diujicobakan adalah *batch-size*, *optimizer*, *epoch*, dan *learning-rate*.
5. Pencarian hyperparameter menggunakan Grid Search tanpa Cross Validation
6. Pencarian hyperparameter tidak mempertimbangkan model mengalami *overfitting* atau undefitting di semua area pencarian yang dilakukan.

1.4 Rumusan Masalah

Sesuai dengan latar belakang yang sudah dijelaskan di atas, rumusan masalah yang digunakan pada penelitian ini, ialah apakah hyperparameter tuning mempengaruhi hasil akurasi pada deteksi Malaria dengan menggunakan CNN.

1.5 Tujuan

Tujuan dari pengerjaan penelitian ini ialah menguji pengaruh *hyperparameter tuning* pada deteksi Malaria dan menemukan *hyperparameter* terbaik pada setiap Arsitektur yang digunakan pada CNN.

1.6 Manfaat

Manfaat dari pengerjaan penelitian ini ialah sebagai berikut.

1. Bagi Universitas Ma Chung khususnya Program Studi Teknik Informatika dapat mempersiapkan lulusan yang berkompeten dan siap kerja dengan memberikan bekal kepada mahasiswa selama proses kegiatan Praktik Kerja Lapangan.
2. Bagi Mahasiswa dapat menerapkan ilmu yang telah diperoleh selama belajar di Universitas Ma Chung dan memberikan pengalaman dunia kerja.
3. Bagi Dunia Akademik dan Praktisi dapat memberikan informasi baru mengenai pengaruh Hyperparameter Tuning pada Model Deteksi CNN Citra Malaria menggunakan Grid Search.

Bab II
Gambaran Umum Perusahaan

2.1 Universitas Ma Chung

Universitas Ma Chung merupakan salah satu universitas swasta di kota Malang yang berdiri pada tanggal 7 Juli 2007. Universitas ini dinaungi oleh Yayasan Harapan Bangsa Sejahtera yang dibentuk oleh Alumni Sekolah Menengah Ma Chung pada Reuni Akbar ke 60. Minum Air Tidak Lupa Sumbernya merupakan salah satu filosofi yang membentuk Universitas Ma Chung sebagai Institusi yang menyesuaikan diri terhadap tuntutan zaman dan menjadi sumber ilmu pengetahuan berbasis sumber daya yang unggul. Universitas Ma Chung memiliki 10 prodi, yaitu Sastra Inggris, PPBM, Farmasi, Manajemen, Akuntansi, Desain Komunikasi Visual, Teknik Industri, Teknik Informatika, Kimia, dan Sistem Informasi yang terbagi kedalam 3 Fakultas, Fakultas Sains dan Teknologi, Fakultas Bahasa dan Seni, dan Fakultas Ekonomi dan Bisnis. Universitas Ma Chung beralamat di Villa Puncak Tidar N-01, Kota Malang, Jawa Timur.

2.2 Teknik Informatika

Program Studi Teknik Informatika (TIF) merupakan program studi Universitas Ma Chung yang mempelajari berbagai prinsip yang berkaitan dengan ilmu komputer mulai dari desain sampai evaluasi sistem perangkat lunak. Program Studi Teknik Informatika Universitas Ma Chung telah mendapatkan nilai akreditasi B oleh Badan Akreditasi Nasional Perguruan Tinggi (BANPT). TIF mempunyai 2 bidang konsentrasi keilmuan atau penjurusan, yaitu Sistem Cerdas dan Sistem Komputer. Sistem Cerdas mempelajari tentang simulasi kecerdasan manusia dalam sebuah mesin, sedangkan Sistem Komputer mempelajari tentang Networking dan implementasi Internet of Things (IoT). Selain penjurusan, TIF mempunyai Pusat Studi Human Machine Interaction (HMI), Pusat Studi Artificial Intellegence in Digital Images and Technoprenuership (AiDiTec), Kelompok Studi Pression Agriculture, dan Google Developer Student Club Universitas Ma Chung (GDSC Ma Chung)

.

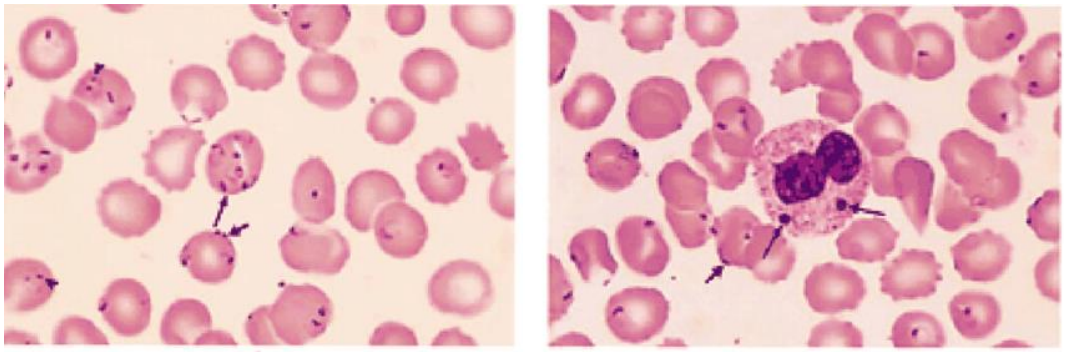
2.3 Pusat Studi AiDiTech

Pusat Studi AiDiTech merupakan kepanjangan dari Artificial Intellegence in Digital Images and Technoprenuership. AiDiTech berada di bawah naungan Teknik Informatika Universitas Ma Chung yang dibentuk pada bulan September 2021. AiDiTech memiliki ruangan operasi di Lantai 6 Gedung RND Universitas Ma Chung. Pusat Studi ini dibentuk dengan tujuan mendukung studi keilmuan di bidang kecerdasan buatan pada citra digital dan technoprenuership. AiDiTech berfokus pada bidang *Deep Learning, Adversarial Attack,* dan pengoptimalan akurasi dengan metode optimasi, regularisasi, dan hyperparameter tuning pada *Deep Learning.*

Bab III
Tinjauan Pustaka

3.1 Malaria

Malaria merupakan penyakit mematikan di berbagai negara, yang disebabkan oleh parasit darah *Plasmodium*. *Plasmodium* disebarkan oleh gigitan nyamuk *Anopheles* betina dan dapat menular melalui transfusi darah atau penggunaan jarum suntik secara bersamaan (Poostchi, 2018). Gejala yang terjadi seperti deman, sakit kepala, dan pucat karena kekurangan darah. Komplikasi yang terjadi dapat mengakibatkan gangguan pernafasan sampai mengalami kematian. Di dunia terdapat lebih dari 200 juta kasus Malaria di tahun 2019 dan menjadi endemi di 89 negara (WHO, 2020). Malaria menyumbang lebih 400 ribu kematian dengan 67% di antaranya masih berusia balita. Pada tahun yang sama Indonesia memiliki lebih 250 ribu kasus Malaria dan 80% di antaranya berada di Papua. Data selama satu dekade terakhir menunjukkan penurunan penderita Malaria baik di Indonesia maupun di dunia.

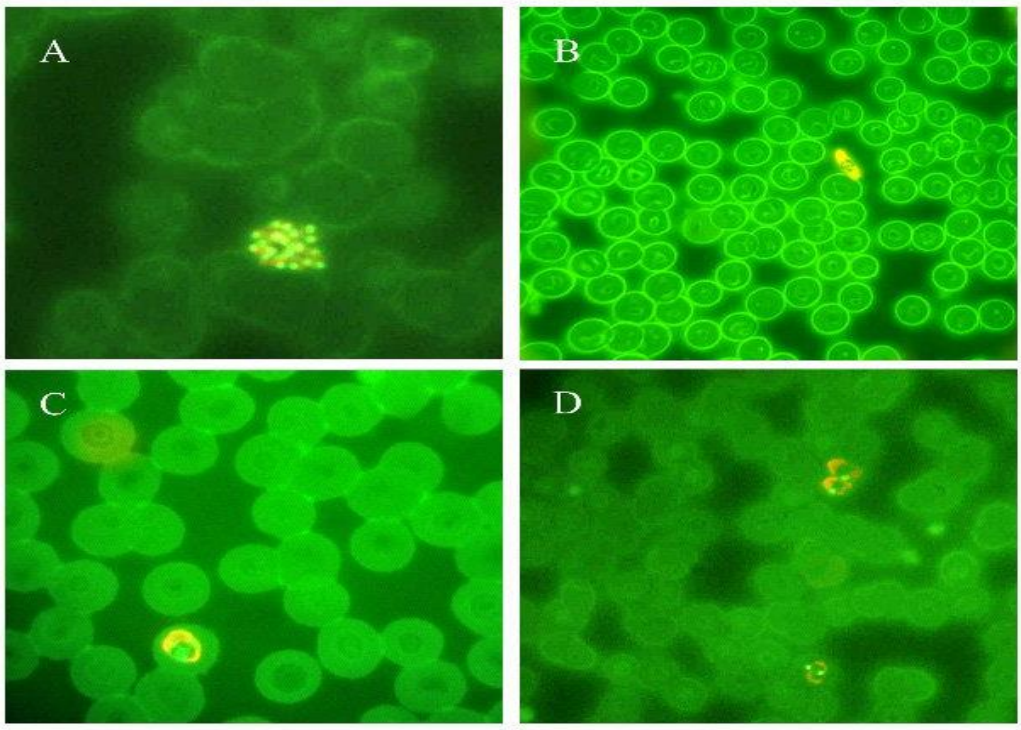


Gambar 3.1 *Peripheral Blood Smear* pewarnaan *Giemsa*

Gambar dari (Ziaee & Abedi, 2014)

Berdasarkan protokol WHO, dalam mendeteksi Malaria dapat dilakukan dengan pemeriksaan *Peripheral Blood Morphology* atau *Blood Smear*, pemeriksaan ini mengevaluasi sel darah merah, sel darah putih, dan trombosit (Bian, 2005). Darah dari pasien akan diwarnai dengan *giemsa (blue purple)* atau *fluorescence (ardiane orange)* ke dalam preparat. Selanjutnya, preparat akan dilihat di bawah mikroskop dengan lensa objektif 10x fokus dan pembesaran 100x. Pemeriksaan dilakukan

dengan metode zig-zag dan melihat adanya bintik *Schuffner* dan bintik *Mauer* pada sel darah merah. Bintik *Schuffner* hanya terlihat pada sel darah merah yang mengandung *plasmodium vivax* atau *plasmodium ovale*, Sedangkan bintik *Mauer* muncul pada sel darah merah yang mengandung *plasmodium falciparum*. Pemeriksaan *Blood Smear* dilakukan sebanyak 3 kali dengan rentang waktu minimal 6 jam (ALOMEDIKA, 2020). Pemeriksaan *Blood Smear* dapat digantikan oleh komputer dengan mengambil citra sampel darah pada preparat dalam bentuk kanal *Red-Green-Blue* (RGB).



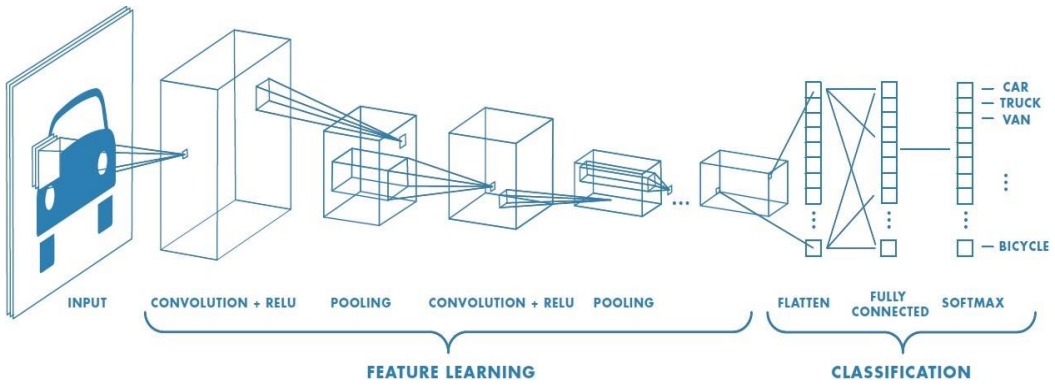
Gambar 3.2 *Peripheral Blood Smear* pewarnaan *fluorescence*

Gambar dari (Calderaro, et al., 2021)

3.2 Convolutional Neural Network

Perkembangan Teknologi memunculkan kecerdasan buatan atau *Artificial Intelligence* (AI). AI merupakan simulasi atau tiruan kecerdasan manusia dalam berpikir dan belajar yang diterapkan ke dalam mesin. AI belajar sendiri dengan menggunakan *Machine Learning* (ML) dan *Deep Learning* (DL) yang menggunakan data untuk memperkaya pengetahuannya. *Machine Learning* merupakan pembelajaran yang berfokus untuk mengembangkan algoritma dalam

melakukan pengambilan keputusan. Secara umum *Machine Learning* belajar dengan metode *Supervised Learning*, *Unsupervised Learning* dan *Reinforcement Learning* (Dahwan, 2021). *Supervised Learning* merupakan cara belajar mesin dengan memberikan label solusi pada proses pembelajarannya, *Unsupervised Learning* merupakan cara belajar mesin dengan melakukan pengelompokan data secara mandiri, *Reinforcement Learning* merupakan proses belajar dengan sistem *reward* dan penalti. Sedangkan *Deep Learning* merupakan cara belajar mesin yang meniru konsep otak manusia dalam mengambil keputusan melalui jaringan neuron atau *Neural Network*. *Deep Learning* menyelesaikan masalah kompleks seperti *Computer Vision* menggunakan *Convolutional Neural Network* dan *Natural Language Processing* menggunakan *Artificial Neural Network*. *Convolutional Neural Network* yang biasa disingkat CNN atau ConvNet merupakan bagian dari *Deep Neural Network* yang terinspirasi dari *Visual Cortex*, yaitu bagian otak yang bertugas untuk memproses informasi dalam bentuk visual (Algorit.ma, 2019).



Gambar 3.3 *Convolutional Neural Network* by MatLab

CNN didesain untuk menangani data dua dimensi dan diaplikasikan ke dalam data citra. Data citra dibuat dari array yang berisi nilai dari pixel dalam resolusi tinggi*panjang*dimensi yang disebut dengan lapisan *channel*. *Channel* mempresentasikan *Red-Green-Blue* yang terbagi ke dalam 3 lapisan *channel* atau *grayscale* yang memiliki 1 lapisan *channel*. Dalam arsitektur CNN terdiri dari 2 bagian besar, yaitu Feature Extraction Layer dan Fully-Connecter Layer.

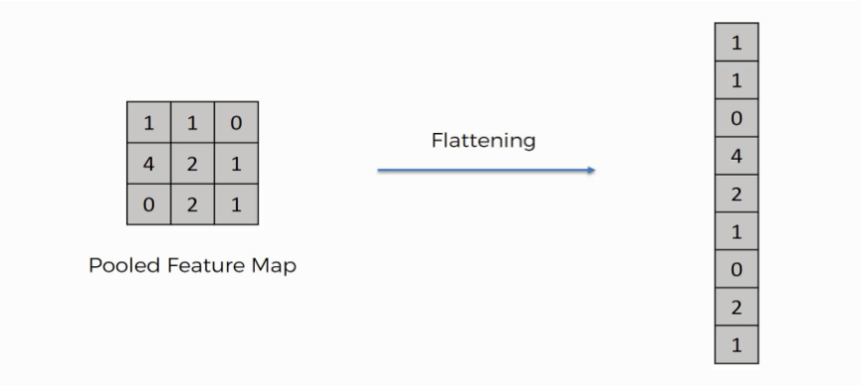
3.2.1 Feature Extraction Layer dan Fully-Connecter Layer.

Feature Extraction Layer terdiri dari dua bagian yaitu Convolutional Layer dan Pooling Layer. Dalam beberapa paper dan riset yang dilakukan

terkadang tidak menggunakan pooling layer. Convolutional layer terdiri dari neuron yang tersusun sedemikian rupa sehingga membentuk sebuah filter dengan panjang dan tinggi bersatuan pixels. Hasil konvolusi dinamakan dengan feature map, dengan fungsi aktivasi ReLu atau *Retified Linear Unit* menggubah nilai negatif pada feature map menjadi positif. Sedangkan Pooling layer merupakan tahapan pengurangan dimensi dari feature map ketika ukuran citra terlalu besar.

Gambar 3.4 Ilustrasi Max Pooling pada Pooling Layer

Gambar dari (https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/)



Gambar 3.5 Ilustrasi Max Flatten pada Fully-Connected Layer

Gambar dari (https://www.superdatascience.com/blogs/convolutional-neural-networks-cnn-step-3-flattening)

Setelah dilakukan pooling, model dalam menghasilkan array multidimensi yang diubah ke dalam vektor pada *fully-connected layer* yang berbentuk seperti *Neural Network*, proses ini dinamakan *Flatten*. Kemudian dengan fungsi aktivasi softmax atau sigmoid digunakan untuk mengklasifikasikan output ke dalam multiclass atau biner (Sena, 2017). Model dari Arsitektur CNN akan di *compile* dengan hyperparameter untuk

mengoptimalkan nilai akurasi dan meminimalkan nilai *loss*. Hyperparameter yang digunakan optimasi seperti *Batch-Size*, *Epochs*, *Learning-Rate*, dan *Optimizers*.

3.2.2 Batch dan Epoch

Nilai dari *batch* merupakan jumlah sampel yang akan digunakan sebelum memperbaharui nilai parameter internal dalam satu *epoch*. *Epoch* merupakan jumlah putaran yang harus dilalui oleh dataset ketika melakukan pelatihan. Nilai *batch* dibatasi oleh jumlah data yang digunakan sedangkan nilai epoch tidak terbatas, akan tetapi semakin besar nilai *epoch* semakin lama model melakukan pelatihan.

3.2.3 Learning-rate dan Optimizer

Learning-rate merupakan parameter yang menghitung koreksi dari bobot saat melakukan proses pelatihan. Nilai *learning-rate* berkisar pada angka nol sampai satu. Semakin besar nilai *learning-rate* maka proses pelatihan akan semakin cepat akan tetapi bisa mengalami kegagalan dalam memperoleh nilai *global optimum*. Semakin kecil nilai *learning-rate* model akan mendapatkan *loss* yang kecil akan tetapi waktu dalam proses pelatihan akan semakin lambat. Nilai *learning-rate* akan mempengaruhi *optimizer* dalam mencari model yang optimal. *Optimizer* merupakan algoritma yang memilih parameter terbaik berdasarkan kategori dalam hal ini adalah *accuracy*. *Optimizer*s miliki banyak algoritma yang bisa digunakan seperti SGD, ADAM, dan ADAMAX.

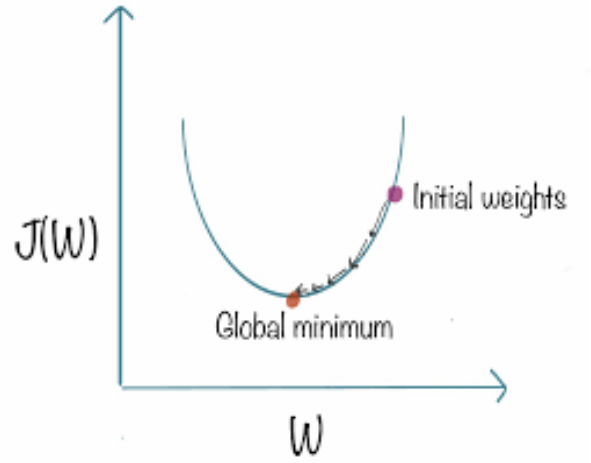
3.2.4 Optimizer SGD

$$\theta_j = \theta_j + \propto \sum_{i=1}^{m} \left(y^{(i)} - h_\theta\left(x^{(i)}\right) \right) x_j^{(i)} \qquad (3.\ 1)$$

Dimana :

- θ_j = bobot
- $\propto$ = learning-rate

Fundamental dari SGD dan ADAM yaitu GD kependekan dari *Gradient Descent* atau penurunan gradient. Persamaan GD secara lengkap dapat dilihat pada rumus 3.1. Dalam data *training* berisi pasangan (x dan y), kemudian diberikan fungsi h untuk prediksi x. Dimana θ merupakan parameter bobot dari fungsi yang akan dicari dan di update. Langkah diulangi menggunakan cost funcition yang menghitung perbedaan h dengan y yang hasilnya dikalikan dengan $\propto$ sebagai *learning-rate*. Pada saat menghitung *cost funcition* perlu melakukan penjumlahan semua perbedaan h dengan y yang dimana jika memiliki data yang besar maka proses ini memakan waktu yang lama (Kurniawan, 2018). Sehingga masalah ini diatasi oleh SGD.



Gambar 3.6 Gradient Descent

Gambar dari (https://www.quora.com/Is-gradient-descent-always-3-dimensional)

SGD merupakan kependekan dari *Stochastic Gradient Descent* merupakan algoritma optimasi yang memperbaharui *weight* setiap *batch*-nya atau setiap nilai m. SGD merupakan perkembangan dari *Gradient Descent* (GD) yang melakukan *update* setiap *epoch*-nya. SGD mengorbankan nilai akurasi untuk mendapatkan kecepatan pelatihan dan penggunaan hardware yang optimal. Tujuan optimasi SGD adalah mencari global minimum dari *loss function*. SGD dipengaruhi oleh hyperparameter momentum, dan *learning-rate*. Momentum merupakan parameter yang digunakan untuk menghindari *local minimum*. Dengan menganalogikan bola, semakin besar nilai momentum semakin cepat bola tersebut berputar (SKILLPLUS, 2019).

3.2.5 Optimizer ADAM

ADAM merupakan algoritma optimasi yang digunakan untuk memperbaharui *weight* secara iteratif berdasarkan data *training*. ADAM merupakan kependekan dari "*adaptive moment estimation*" yang mengestimasi nilai gradien untuk mengadaptasi nilai *learning-rate* untuk setiap weight *Neural Network*. Dengan kemampuan ADAM dalam mendapatkan *global optimum* membuat ADAM menjadi *optimizer* default di semua arsitektur CNN (Tilawah, 2020).

Secara sederhana ADAM merupakan gabungan dari dua *Optimizer* yaitu GD+ momentum dan RMSP. Momentum menurunkan gradien menggunakan rata-rata-rata bobot. Dengan begitu algoritma dapat bekerja dengan cepat. Sehingga Adam memiliki 2 perhitungan bobot yang satu untuk bobot momentum yang satunya untuk RMSprop. Sehingga perhitungan ADAM secara sederhana dapat dijelaskan pada rumus 3.2. rumus ini digunakan untuk melakukan pembaruan nilai bobot pada setiap epochnya.

$$w_{i+1} = w_i - \widehat{m_j}\left(\frac{\alpha}{\sqrt{\widehat{v_j}+\epsilon}}\right) \qquad (3.\ 2)$$

Dimana :

- w = bobot
- $\propto$ = learning-rate
- m = momentum
- V = jumlah gradien kuadrat sebelumnya
- ϵ = kontanta positif (10^{-8})

ADAMAX merupakan salah satu variasi dari ADAM. ADAMAX didesain dengan dasar ADAM yang lebih cepat dalam menemukan *global optimum* dengan pendekatan norma tak terbatas (Machine Learning Mastery, 2021). Secara sederhana perhitungan ADAMAX dapat dilihat pada rumus 3.3

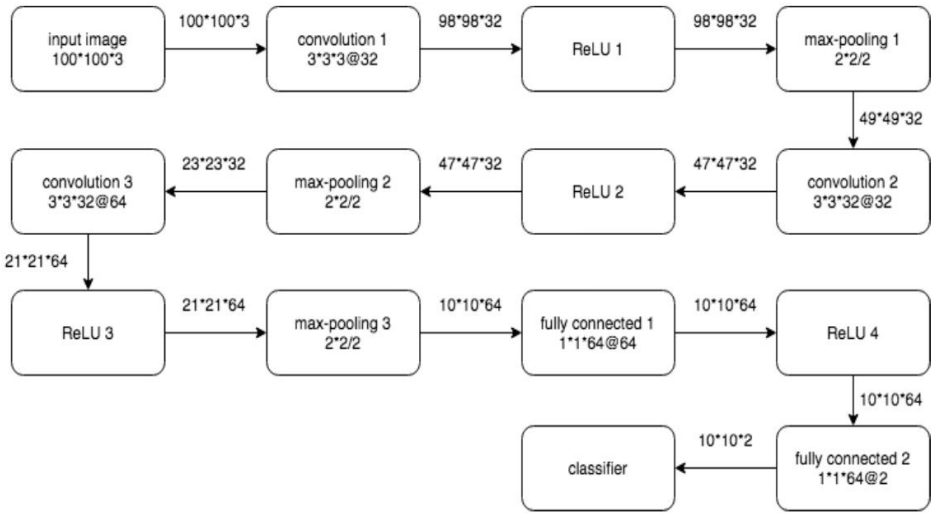
$$w_i = max(\beta_2 \cdot w_{i-1}, |V_i|) \quad (3.3)$$

Dimana :

- w = bobot
- β = eksponensial decay rate (0,1)
- m = momentum
- V = jumlah gradien kuadrat sebelumnya

3.2.6 Arsitektur Model

Model dari CNN memiliki berbagai arsitektur yang berbeda seperti VGG-16, RestNet, dan BaselineNet. Arsitektur CNN dapat dibuat secara *custom* untuk tujuan khusus seperti arsitektur Rajaraman (Rajaraman, et al., 2018) yang memiliki performa paling baik saat mendeteksi Malaria. Dalam arsitektur yang dapat dilihat di gambar 3.7, citra RGB akan dikonversi ke 100 x 100 ukuran *input.* Kemudian citra dimasukkan ke dalam *hidden layer* dengan konvolusi 3x3, fungsi aktivasi ReLU dan dilakukan max pooling. Kemudian citra dimasukkan ke dalam fully-connected layer, dengan fungsi aktivasi ReLU yang akan menghasilkan 2 kelas, *Paracitized* dan *Uninfected*.



Gambar 3.7 Arsitektur Rajaraman

Gambar 3.8 Arsitektur Baseline Network

Arsitektur BaselineNet diperkenalkan oleh (Yi, et al., 2016) yang diperlihatkan pada gambar 3.8. Citra RGB akan di ke 64x64 ukuran input. Kemudian dimasukkan ke dalam empat hidden layer dan empat max-pooling, dengan fungsi aktivasi ReLU. Setelah itu feature-map akan dibawa ke fully-connected dan menghasilkan 2 kelas, *Paracitized* dan *Uninfected*.

3.3 Hyperparameter Tuning

Hyperparameter Tuning merupakan serangkaian percobaan yang mencari hyperparameter untuk mendapatkan model yang berperforma paling baik. Hyperparameter sendiri merupakan sebuah parameter yang dapat mempengaruhi output dari sebuah arsitektur. Perlakuan percobaan dalam hyperparameter seperti *epoch*, *batch-size*, *optimizer*, dan lainya untuk mendapatkan akurasi model yang paling baik. Pencarian ini dilakukan secara trail dan error dan berulang-ulang. Tiga algoritma dasar yang sering digunakan dalam mencari nilai hyperparameter terbaik yaitu *Grid Search*, *Random Search*, dan *Bayesian Optimization* (Google Cloud AI Platform, 2021).

3.3.1 Grid Search

Grid Search merupakan pencarian hyperparameter terbaik dengan mengombinasikan semua parameter yang dibutuhkan. Metode ini berhasil

mendapatkan *accuracy* terbaik dengan mengorbankan waktu yang dibutuhkan dalam pencarian (Malik, 2020). Evaluasi dari setiap pencarian menggunakan nilai rata-rata dari *accuracy* model yang dikemudian dibandingkan dengan pencarian nilai hyperparameter lain. Nilai rata-rata *accuracy* tertinggi akan menjadi best hyperparameter. Penggunaan *Grid Search*, bisa dengan menggunakan library yang ada di keras yaitu dengan menggunakan Grid Search Cross Validation dan *Halving Grid Search Cross Validation*.

3.3.2 Random Search

Algoritma *Random Search* memiliki kemiripan dengan algoritma grid search, akan tetapi random search hanya mengambil beberapa hyperparameter secara random sebanyak yang ditentukan. Dengan pengambilan nilai hyperparameter secara acak, random search dapat mengurangi waktu pelatihan yang dibutuhkan akan tetapi random search dapat gagal dalam mencari hyperparameter yang optimal. Library yang sering digunakan saat melakukan pencarian random seperti Randomized Search Cross Validation dan Halving Randomized Cross Validation.

3.3.3 Bayesian Optimization

Algoritma *Bayesian Optimization* merupakan algoritma pencarian yang berfokus untuk memilih hyperparameter yang tampak menjanjikan. Konsep dari *bayesian optimization* adalah pencarian hyperparameter berdasarkan nilai evaluasi berdasarkan uji coba sebelumnya (Koehrsen, 2018).

3.4 Confusion Matrix

Salah satu teknik yang digunakan untuk mengevaluasi kinerja dari suatu model adalah *Confusion matrix*. *Confusion matrix* melakukan perbandingan antara hasil prediksi dengan kondisi sebenarnya. Seperti terlihat pada tabel 3.1, *Confusion Matrix* memiliki 4 metrik, *True Positive (FP)*, *False Positive (FP)*, *False Negative*, dan *True Negative(TN)*. *True Positive* merupakan metrik yang menyatakan bahwa

hasil prediksi positif sesuai dengan hasil positif. *False Negative* merupakan metrik yang menyatakan bahwa hasil prediksi negatif tetapi hasil sebenarnya positif. *True Negative* merupakan metrik yang menyatakan hasil prediksi negatif dan hasil sebenarnya juga negatif. *False Negative* merupakan metrik yang menyatakan hasil prediksi positif, akan tetapi hasil sebenarnya negatif.

Tabel 3.1. *Confusion matrix*

Kondisi Sesungguhnya	Hasil Prediksi	
	Paracitized	*Uninfected*
Paracitized	*TP*	*FN*
Uninfected	*FP*	*TN*

Confusion matrix menjelaskan kesalahan dari model dalam melakukan prediksi. Dari nilai *confusion matrix* dapat mengevaluasi model menggunakan *accuracy, f1-score, precision, recall*, dan *matthew correlation coefficient*. *Accuracy* menggambarkan seberapa akurat model dapat mengklasifikasikan dengan benar. *Precision* menggambarkan rasio prediksi benar positif dibandingkan dengan keseluruhan hasil prediksi positif. *Recall* menggambarkan rasio prediksi benar positif dengan keseluruhan hasil sebenarnya positif. *Matthew Correlation Coefficient (*MCC*)* menggambarkan kesalahan klasifikasi terutama pada sampel negatif. *F1-score* menggambarkan perbandingan rata-rata *precision* dan *recall* yang dibobotkan.

$$Accuracy = \frac{(TP+TN)}{(TP+FP+FN+TN)} \quad (3.4)$$

$$F1-score = \frac{2*TP}{2*TP+FN+FP} \quad (3.5)$$

$$Precision = \frac{(TP)}{(TP+FP)} \quad (3.6)$$

$$Recall = \frac{(TP)}{(TP+FN)} \quad (3.7)$$

$$MCC = \frac{TN*TP-FP*FN}{\sqrt{(TP+FN)(FP+TP)(TN+FP)(FNTP)}} \quad (3.8)$$

Di mana:

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

3.5 Google Colab

Colab merupakan website yang dibuat oleh Google dengan yang menjalankan code Python melalui browser dengan mudah, dapat dibagikan ke orang lain. Colab memiliki fungsi yang sama dengan Notebook, sebuah dokumen yang dapat mengeksekusi code di satu tempat yang sama tanpa perlu berpindah-pindah. Dalam penggunaan nya colab mengenalkan GPU, CPU, dan TPU sebagai eksekutor. Dalam versi berbayar memiliki runtime high-ram dari TPU dan GPU, dengan runtime yang lebih panjang.

3.6 Python

Python merupakan bahasa program tingkat tinggi yang dirilis pada 1991 oleh Guido van Rossum. Python didesain untuk kegiatan analitik dan komputasi bagi user yang tidak terlalu mendalami coding. Python memiliki struktur sederhana dengan lebih dari 140 ribu libray dan dikembangkan secara open source.

3.7 Penelitian Terkait

Penelitian ini melanjutkan beberapa penelitian sebelumnya, berikut adalah gambaran penelitian yang diambil menjadi rujukan sekaligus keterkaitan dengan penelitian yang akan dilakukan. Penelitian dan Rujukan utama sudah dirangkum pada tabel 3.2.

Tabel 3.2. Penelitian Terkait

No	Referensi	Keterkaitan dengan Penelitian yang akan dilakukan
1	Rajaraman S, at all. 2018. Understanding the learned behavior of customized convolutional neural network toward malaria parasite detection in thin blood smear images. *Journal of Medical Imaging.* Volume 5(3), pp. 034501	CNN merupakan Deep Learning yang ditujukan untuk citra. Penelitian ini melakukan evaluasi dari arsitektur Rajaraman, Vgg-16, ResNet-50, Xception, Inception-V3 dan DenseNet 121. Yang hasilnya kemudian dievaluasi

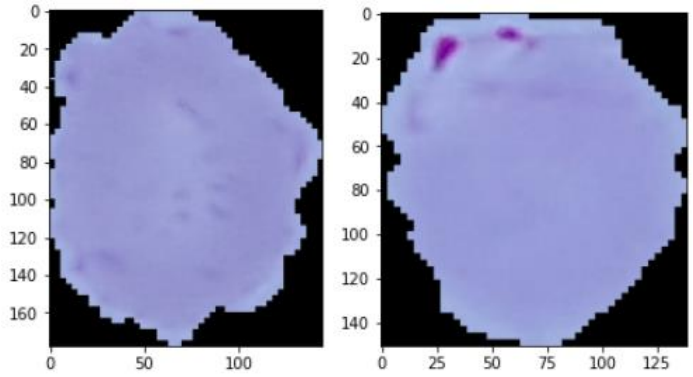
No	Referensi	Keterkaitan dengan Penelitian yang akan dilakukan
		menggunakan Uji Kruskal Walliss. Penelitian ini berhasil membuktikan adanya perbedaan signifikan antara arsitektur yang berbeda dengan menggunakan data yang besar. Keterkaitan dengan penelitian yang akan dilakukan, adalah menggunakan Arsitektur Custum simple CNN sebagai salah satu arsitektur yang akan diujicobakan. Arsitektur ini menghasilkan *accuracy* 0,94, F1-score 0,941, dan MCC 0,880 pada optimizer SGD
2	Suriya M., at all. 2019. Enhanced deep convolutional neural network for malaria parasite classification. *International Journal of Computers and Appications.* DOI : 10.1080/1206212X.2019.1672277	Studi ini membuat satu arsitektur baru yang dinamakan Deep Convolutional Neural Network (DCNN) untuk melakukan deteksi malaria. Dengan melakukan hyperparameter tuning pada optimizer ADAM dan Adagrad dengan beberapa variasi epoch. Evaluasi model menggunakan Kappa Coeficient dan Mattew's correlation coefficient. Secara keseluruhan DCNN menghasilkan akurasi sebesar 98,9%

No	Referensi	Keterkaitan dengan Penelitian yang akan dilakukan
		Pada penelitian yang akan dilakukan, pencarian hyperparameter dilakukan dengan cara yang sama pada pencarian DCNN. Pencarian ini sama dengan Grid Search.
3	Swastika W., at all. 2020. Effective preprocessed thin blood smear images to improve malaria parasite detection using deep learning. *Journal of Physics: Conference Series.* 1869 pp. 012092	Penelitian ini membahas mengenai pengaruh preprocessing pada pendeteksian malaria menggunakan deep learning. Arsitektur yang digunakan adalah VGG-16, RestNet, dan Rajaraman. Preprocessing dilakukan dengan gray-world normalization dan comprehensize normalization. Hasil yang didapatkan dapat meningkatkan akurasi sebesar 0,525% dan 0,035% pada RestNet-50 dan Rajaraman. Pada penelitian yang akan dilakukan kami mengambil konsep evaluasi yang telah dilakukan.

Bab IV
Perancangan Sistem

4.1 Dataset Malaria

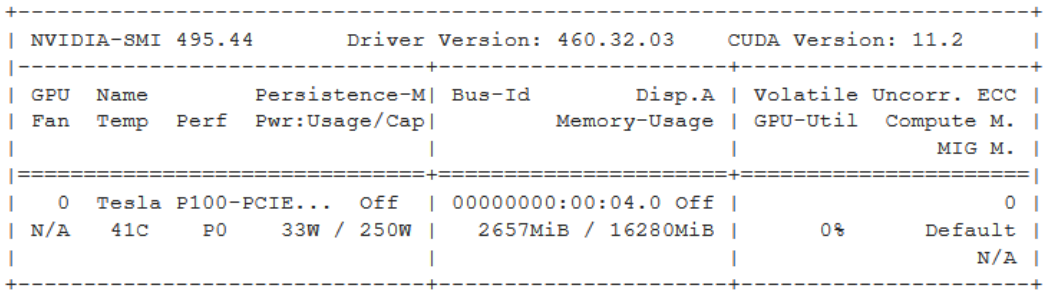
Dataset yang digunakan adalah dataset citra Malaria yang terdiri dari dua jenis gambar, yaitu normal dan yang parasit Malaria. Dataset ini disediakan oleh Lister Hill National Center for Biomedical Communications (LHNCBC) yang bergabung ke dalam United States National Library of Medicine (NLM). Dataset ini terdiri dari 27.558 citra *blood smear* yang terbagi ke dalam 2 kelas, *Paracitized* dan *Uninfected* dengan jumlah yang sama. Sampel citra yang akan digunakan dapat dilihat pada gambar 4.1. Kemudian setiap citra akan di reshape ke 1/255.0 dan diubah menyesuaikan kebutuhan input dari arsitektur yang digunakan. Kemudian, dataset akan bagi ke dalam data *training, testing,* dan *validation* dengan rasio 80:10:10. Data *training* dan *validation* akan digunakan untuk melakukan pencarian hyperparameter pada model dan data *testing* sebagai data evaluasi di luar sampel.



Gambar 4.1. Sampel Citra *Paracitized* (kanan) dan *Uninfected* (kiri) pada Dataset Malaria LHNCBC

4.2 Kebutuhan Sistem

Sistem yang digunakan untuk melakukan penelitian menggunakan Google colab pro dengan runtime GPU (High-Ram). GPU yang digunakan merupakan GPU dari NVIDIA-SMI secara detail dapat dilihat pada gambar 4.2.



Gambar 4.2. GPU NVIDIA-SMI pada Google Colab Pro

4.3 Arsitektur Model

Penelitian ini dilakukan dengan menggunakan 4 model yang terdiri dari 2 jenis arsitektur, custom CNN Rajaraman dan BaselineNet dengan 2 perlakuan, *Hyperparameter Tuning* menggunakan *Grid Search* dan tanpa *Hyperparameter Tuning*. Setiap arsitektur yang ditambahkan *Hyperparameter Tuning*, dilakukan pencarian *Grid* secara manual dengan menggunakan fungsi *looping* pada python. Model tanpa *Hyperparameter Tuning* atau disebut juga model acuan dilakukan percobaan selama 5 kali dengan hyperparameter yang digunakan pada tabel 3.1. Model tanpa Hyperparameter didapatkan dengan mengambil nilai random yang memiliki akurasi pada diatas 0,93%. Ketika model Hyperparameter Tuning ketika sudah mendapatkan Best Hyperparameter kemudian dievaluasi sebanyak 5 kali dan dibandingkan dengan model tanpa Hyperparameter Tuning. Penelitian ini dilakukan hingga mendapatkan model terbaik pada setiap arsitekturnya.

Tabel 4.1. 2 Arsitektur Model dengan Grid Search dan Tanpa Grid Search

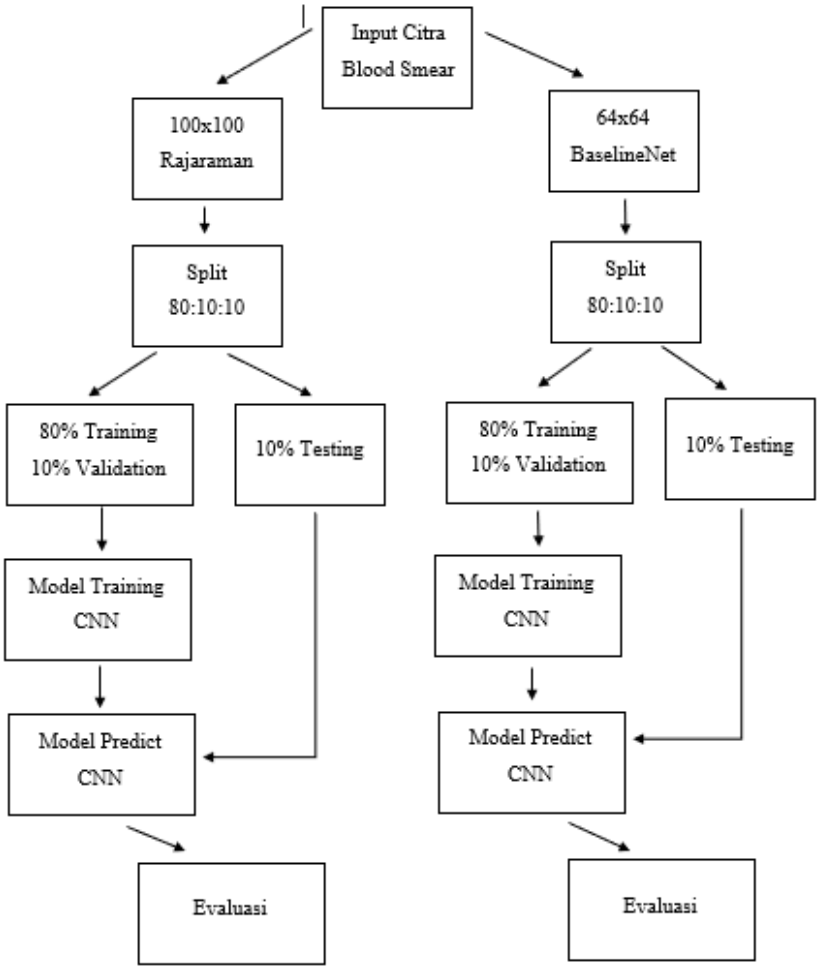
No	Arsitektur	Perlakuan	Input Resolution	Learning Rate	Optimizer	Batch-Size	Epochs
1	Rajaraman	None	100 x 100	0,01	SGD	32	60
2	Rajaraman	Grid Search	100 x 100	Best Hyperparameter dari Grid Search			
3	BaselineNet	None	64 x 64	0,001	ADAM	64	20
4	BaselineNet	Grid Search	64 x 64	Best Hyperparameter dari Grid Search			

4.4 Hyperparameter Tuning

Jenis Hyperparameter sekaligus area pencarian yang digunakan pada penelitian ini adalah sebagai berikut.

a. Jumlah Batch Size : 16, 24 dan 32
b. Jumlah Epochs: 10, 20, 30, dan 40
c. Learning Rate : 0.1, 0.01, dan 0.001
d. Optimizer : ADAM, ADAMAX, dan SGD

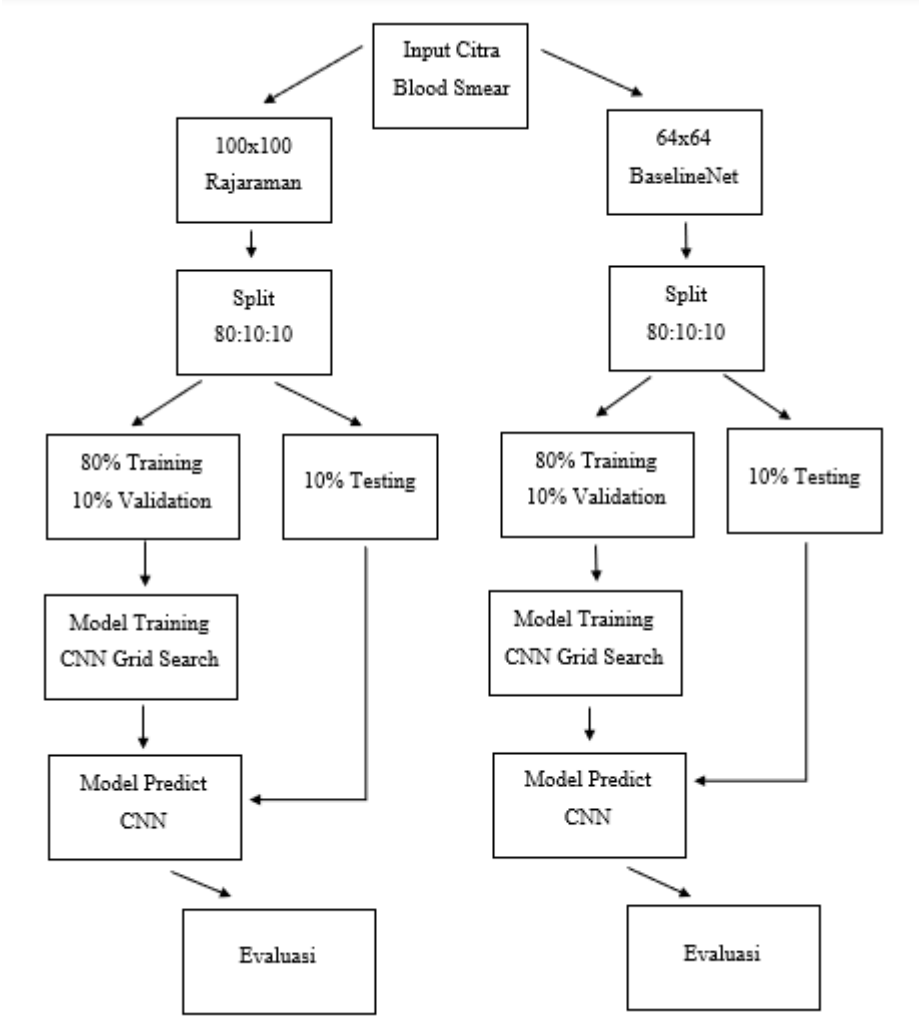
4.5 Desain Sistem



Gambar 4.3 Deteksi Malaria tanpa Hyperparameter Tuning pada model CNN

Sistem dimulai dengan input berupa citra dari dataset Malaria. Kemudian dilakukan preprocessing dengan mengubah ukuran ke 64x64x3 untuk BaselineNet dan 100x100x3 untuk Rajaraman yang kemudian disimpan dalam bentuk Array. Citra diberikan label *Paracitized* untuk yang terkena Malaria dan *Uninfected* untuk

yang sehat. Dataset dibagi ke dalam data *training*, *testing*, dan *validation* dengan rasio 80:10:10. Kemudian intensitas citra dijadikan antara 0 dan 1 dengan membagi citra 1/255.0 dan menjadikan data siap digunakan. Langkah ini dilakukan baik saat tidak melakukan hyperparameter tuning seperti pada gambar 4.3 dan melakukan hyperparameter tuning menggunakan grid search seperti pada gambar 4.4.



Gambar 4.4. Deteksi Malaria Hyperparameter Tuning menggunakan Grid Search pada model CNN

Pada gambar 4.3, sistem melakukan prediksi secara langsung kepada model CNN dan dilakukan evaluasi menggunakan data test. Sedangkan pada gambar 4.4 sistem melakukan pencarian hyperparameter menggunakan grid search untuk menemukan model akurasi tertinggi yang menjadi best model kemudian dievaluasi menggunakan data test. Pencarian grid search dapat dilihat pada potongan kode di bawah ini.

```
1
2
3
4   metriks = []
5   for valueLearningRate in learnRate:
6      for valueOptimizer in optimizers:
7          for valueBatch in batchSize:
8              for valueEpoch in epochs:
9                  metriks.append(valueLearningRate)
10                 metriks.append(valueOptimizer)
11                 metriks.append(valueBatch)
12                 metriks.append(valueEpoch)
13
14  for i in range(0,len(metriks,4):
15     compilemodel(metriks[i], metriks[i+1],
16                 metriks[i+2], metriks(i+3))
17
18
```

Kode diatas digunakan untuk melakukan penyetelan hyperparameter tuning dengan menggunakan value yang kemudian dilakukan looping untuk melakukan pergantian hyperparameter kemudian menjalankan perulangan model setiap pergantian hyperparameter dan dievaluasi menggunakan akurasi dari model. Nilai akurasi akan dibandingkan dengan nilai akurasi parameter yang lain dan diambil best hyperparameter untuk nilai akurasi tertinggi.

Pencarian hyperparameter tidak mempertimbangkan apakah model mengalami underviting, *overfitting*, ataupun tidak. Sehingga disarankan untuk melakukan uji coba terlebih dahulu sebelum mempertimbangkan penggunaan *best hyperparameter* secara masal di semua kasus deteksi Malaria menggunakan Arsitektur Rajaraman dan BaselineNet.

4.6 Evaluasi

Pada saat pencarian Grid, model akan dievaluasi berdasarkan akurasi pada data *validation*, dan diambil model akurasi tertinggi sebagai hyperparameter terbaik. Kemudian model dievaluasi pada data test menggunakan *confusion matrix*. Dari *confusion matrix* diambil nilai *f1-score, accuracy, recall, precision,* dan MCC dan dibandingkan dengan non tuning, untuk mengetahui seberapa banyak pengaruh dari hyperparameter non tuning dengan best hyperparameter.

4.7 Uji Signifikansi

Uji Signifikansi terdiri dari 2 metode yaitu *Parametrik Test* dan *Nonparametrik Test*. Syarat untuk melakukan uji *Parametrik* yaitu data yang digunakan harus berdistribusi normal sedangkan *Nonparametrik* distribusi data diluar normal. Kemudian setelah diketahui distribusi sebuah data, untuk Uji *Parametrik* dilakukan uji *homogenitas* atau uji kesamaan. Pada Uji *Parametrik* dan *Nonparametrik* dilakukan Uji *Signifikansi* atau uji asumsi pengaruh yang digunakan.

Pada penelitian ini, uji *Normalitas* menggunakan *Shapiro-Wilk*, dan uji *Homogenitas* menggunakan *Lavene Test*. Ketika data berdistribusi normal, Uji *Signifikansi Parametrik* menggunakan *Independent Samples T-test*, sedangkan ketika data tidak berdistribusi normal, *Uji Signifikansi Nonparamertik* menggunakan *Two Sample Kolmogorov Smirnov Test.* Asumsi dan hipotesis yang akan digunakan terangkum pada tabel 4.2 yang berisi uji *Normalitas, Homogenitas,* dan *Signifikansi*.

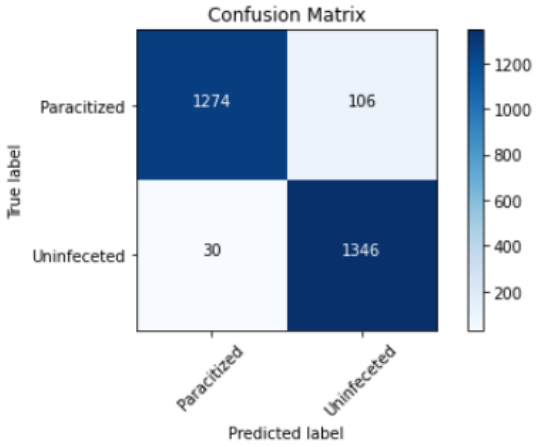
Tabel 4.2. Asumsi dan hipotesis uji normalitas, homogenitas, dan signifikansi

Pengujian	Asumsi	Hipotesis
Normalitas	p > 0,05 maka H0 gagal ditolak	H0 : Data berdistribusi normal
	p < 0,05 maka H0 ditolak	H1 : Data tidak berdistribusi normal
Homogenitas	p > 0,05 maka H0 gagal ditolak	H0 : Data homogen
	p < 0,05 maka H0 ditolak	H1 : Data tidak homogen
Signifikansi	p > 0,05 maka H0 gagal ditolak	H0 : Data tidak memiliki perbedaan signifikan
	p < 0,05 maka H0 ditolak	H1 : Data memiliki perbedaan signifikan

Bab V
Hasil dan Pembahasan

5.1 Arsitektur Rajaraman

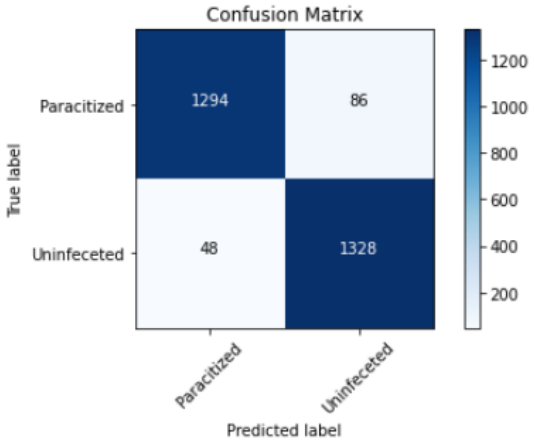
Rajaraman dengan input 100x100, di reshape 1/255.0, dengan hyperparameter *learning-rate* 0.01, *optimizer* SGD, *batch-size* 32, dan *epochs* 60. Model menghasilkan *confusion matrix* yang dapat dilihat pada gambar 5.1 dengan nilai *TP* 1274, *FP* 30, *TN* 1346, dan *FN 10*6. Dari nilai *confusion matrix*, didapatkan nilai *metrix accuracy 0,9506, precision 0,9269, recall 0,9781, f1-score 0,9519, dan* MCC *0,9026.*



Gambar 5.1 *Confusion matrix* pada Rajaraman

5.2 Arsitektur BaselineNet

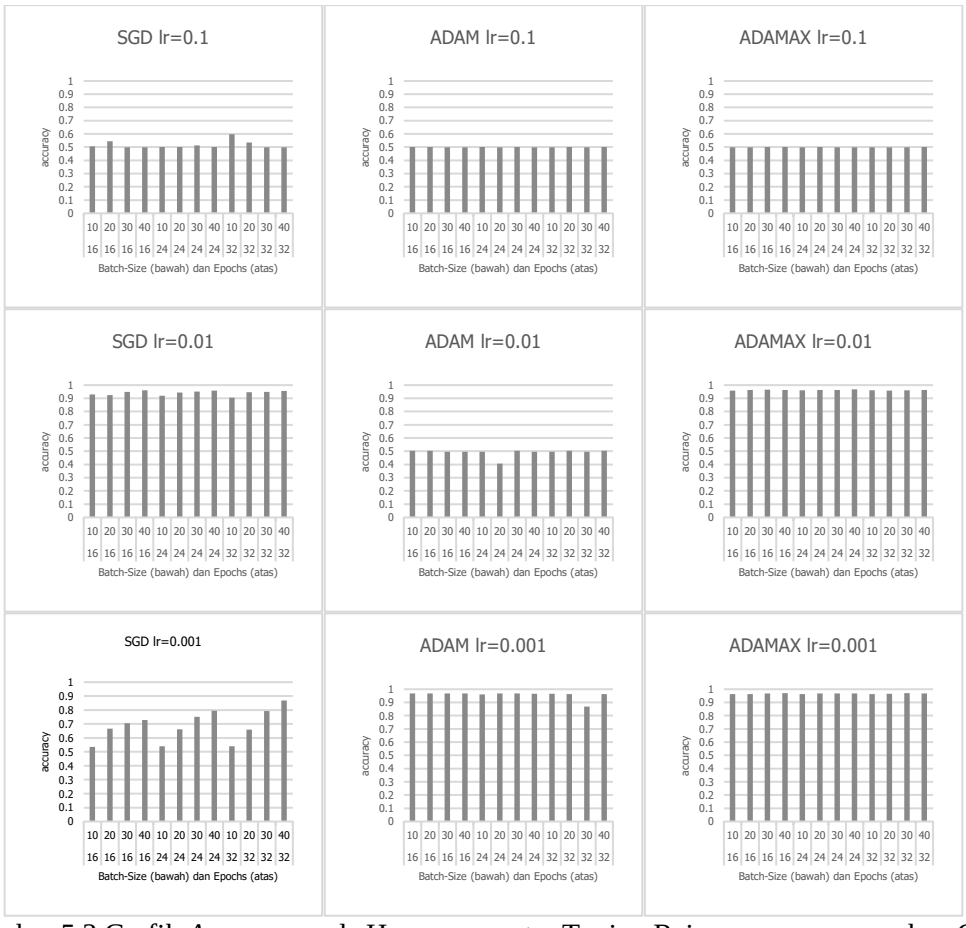
BaselineNet dengan input 64x64, di reshape 1/255.0, dengan hyperparameter *learning-rate* 0.001, *optimizer* ADAM, *batch-size* 64, dan *epochs* 20 menghasilkan *TP* 1294, *FP* 48, *TN* 1328, dan *FN* 86. Dari nilai *confusion matrix* didapatkan nilai evaluasi dengan *accuracy 0,9513, precision 0,9391, recall 0,9651, f1-score 0,9519, dan* MCC *0,9031.*



Gambar 5.2 *Confusion matrix* pada BaselineNet

5.3 Hyperparameter Tuning Rajaraman menggunakan Grid Search

Rajaraman dengan input 100x100, di reshape 1/255.0, dan dilakukan pencarian dengan menggunakan *Grid Search*. Hasil *accuracy Grid Search* dapat dilihat pada Gambar 5.3 dan hasil pencarian dapat dilihat di Lampiran A.

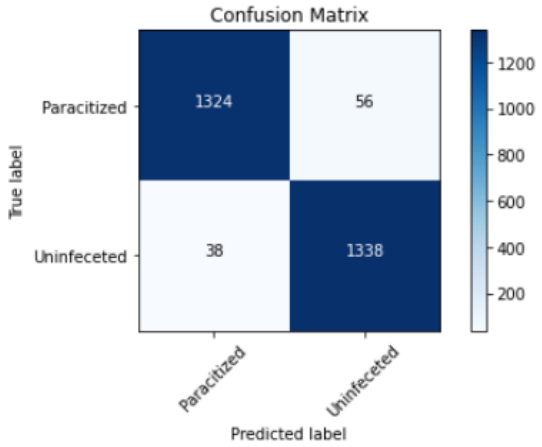


Gambar 5.3 Grafik *Accuracy* pada Hyperparameter Tuning Rajaraman menggunakan Grid Search

Dari Gambar 5.3 dapat dilihat pada saat *Learning-rate* 0,1 model gagal mencari *global optimum* dan berhenti di *local optimum* dan ketika *Learning-rate* diubah ke dalam 0,01 model mulai keluar dari *local optimum* dan bergerak ke *global optimum*. Dengan menggunakan *Optimizer* ADAM dan ADAMAX model memiliki akurasi yang stabil dan berdekatan sedangkan menggunakan *optimizer* SGD, model mengalami kenaikan dan penurunan akurasi secara tiba-tiba. Pada *optimizer* SGD mengalami anomali pada *learning-rate* 0,01 dan 0,001 yang dimana tingkat akurasi pada *learning-rate* 0,001 lebih buruk dari 0,01. Hal ini, berbeda dengan hasil akurasi pada *Optimizer* ADAM dan ADAMAX pada *learning-rate* yang sama.

Hasil Hyperparameter terbaik dari *Grid Search* pada Rajaraman adalah 0,9695 sebagai akurasi tertinggi dengan *Optimizer* ADAMAX, *learning-rate* 0,001, *batch-size* 32, dan *epoch* 30. Hyperparameter menghasilkan *confusion matrix* pada gambar 5.4 dengan nilai *TP* 1324, *FP* 38, *TN* 1338, dan *FN* 56. Dari nilai *confusion matrix* didapatkan nilai evaluasi dengan *accuracy 0,9658, precision 0,9598, recall 0,9723, f1-score 0,9660, dan* MCC *0,9318.*

,



Gambar 5.4 *Confusion matrix* pada Best Hyperparameter pada Rajaraman

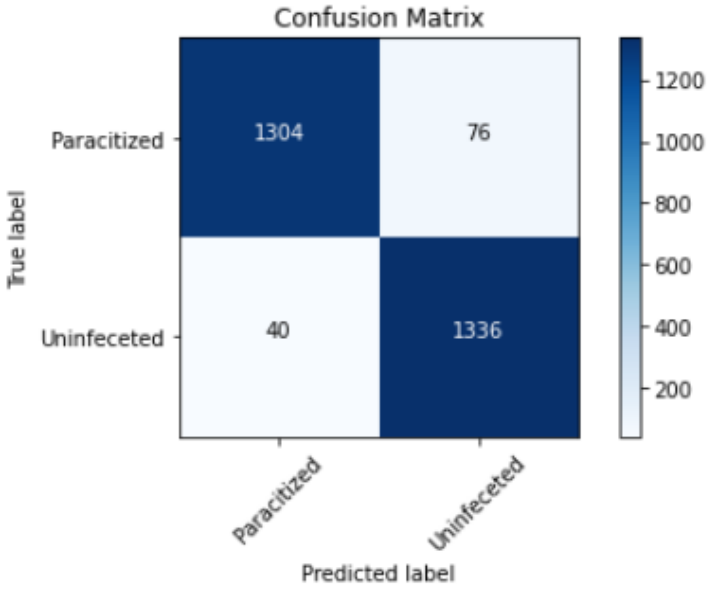
5.4 Hyperparameter Tuning BaselineNet menggunakan Grid Search

BaselineNet dengan input 64x64, di reshape 1/255.0, dan dilakukan pencarian dengan menggunakan *Grid Search*. Hasil *Grid Search* dapat dilihat pada gambar 5.5 dan hasil pencarian dapat dilihat di Lampiran B. Dari gambar 5.5 dapat dilihat saat model menggunakan *Learning-rate* 0,1 pada *optimizer* ADAM dan ADAMAX model gagal mencari *global optimum* dan berhenti di *local optimum*.

Model yang menggunakan *optimizer* ADAM dan ADAMAX memiliki tingkat akurasi yang stabil. Sedangkan model yang menggunakan *optimizer* SGD tingkat akurasi tidak stabil. Berbeda dengan *optimizer* ADAM dan ADAMAX yang nilai *learning-rate* berbanding terbalik dengan nilai akurasi, SGD memiliki nilai akurasi yang berbanding lurus dengan nilai *learning-rate*.

Gambar 5.5 Grafik *Accuracy* pada Hyperparameter Tuning BaselineNet menggunakan Grid Search

Hasil Hyperparameter terbaik dari *Grid Search* pada BaselineNet adalah 0,9612 sebagai akurasi tertinggi dengan *Optimizer* SGD, *learning-rate* 0,1, *batch-size* 32, dan *epoch* 20. Hyperparameter menghasilkan *confusion matrix* pada gambar 5.6 dengan nilai *TP* 1304, *FP* 40, *TN* 1336, dan *FN* 76. Dari nilai *confusion matrix* didapatkan nilai evaluasi dengan *accuracy 0,9578, precision 0,9461, recall 0,9709, f1-score 0,9583, dan* MCC *0,9161*.



Gambar 5.6 *Confusion matrix* pada Best Hyperparameter pada BaselineNet

5.5 Perbandingan Evaluasi Rajaraman dengan Rajaraman Hyperparameter Tuning

Rajaraman dengan input 100x100, dan di-*reshape* 1/255.0, dan hyperparameter model yang digunakan adalah 60 untuk *epoch*, *optimizer* SGD, *batch-size* 32, dan *learning-rate* 0,01. Model dievaluasi menggunakan *confusion matrix* yang menghasilkan nilai *accuracy*, *precision*, *recall*, *f1-score*, MCC dan percobaan diulangi sebanyak 5 kali. Kemudian dibandingkan dengan model dari Best Hyperparameter yang telah didapatkan melalui hyperparameter tuning pada Rajaraman menggunakan *Grid Search* yaitu *optimizer* ADAMAX, *learning-rate* 0,001, *batch-size* 32, dan *epoch* 30. Kemudian model best hyperparameter akan dievaluasi menggunakan *confusion matrix* yang menghasilkan nilai *accuracy*, *precision*, *recall*, *f1-score*, MCC dan perhitungan model diulangi sebanyak 5 kali. Hasil percobaan dapat dilihat pada tabel 5.1.

Tabel 5.1. Hasil *Accuracy, Precision , Recall, F1-score,* dan MCC pada arsitektur Rajaraman dan Best Hyperparameter Rajaraman

Percobaan		Accuracy	Precision	Recall	F1-score	MCC
Rajaraman	1	0,9597	0,9391	0,9842	0,9611	0,9204
	2	0,9600	0,9416	0,9820	0,9614	0,9209
	3	0,9560	0,9333	0,9835	0,9577	0,9134
	4	0,9535	0,9278	0,9849	0,9555	0,9087
	5	0,9589	0,9433	0,9777	0,9602	0,9185
Rata-rata		**0,9576**	**0,9370**	**0,9825**	**0,9592**	**0,9164**
Rajaraman Best Hyperparameter	1	0,9651	0,9495	0,9835	0,9662	0,9309
	2	0,9677	0,9611	0,9756	0,9683	0,9354
	3	0,9673	0,9631	0,9727	0,9679	0,9347
	4	0,9680	0,9547	0,9835	0,9689	0,9365
	5	0,9669	0,9534	0,9828	0,9679	0,9343
Rata-rata		**0,9670**	**0,9564**	**0,9796**	**0,9678**	**0,9344**

Pada arsitektur Rajaraman perlakuan tuning pada hyperparameter menggunakan Grid Search memberikan performa *accuracy* sebesar 96,70% yang meningkat 0,94% dibandingkan tanpa hyperparameter tuning. Performa *precision* pada perlakuan hyperparameter tuning menggunakan Grid Search meningkat sebesar 1,93% dibandingkan tanpa hyperparameter tuning. Performa *recall* pada perlakuan hyperparameter tuning menggunakan Grid Search menurun sebesar 0,28% dibandingkan tanpa hyperparameter tuning. Performa *f1-score* pada perlakuan hyperparameter tuning menggunakan Grid Search meningkat sebesar 0,87% dibandingkan tanpa hyperparameter tuning. Kemampuan model mengenali prediksi negatif yang terdapat pada MCC di perlakuan hyperparameter tuning menggunakan Grid Search meningkat sebesar 1,80% dibandingkan tanpa hyperparameter tuning.

5.6 Perbandingan Evaluasi BaselineNet dengan BaselineNet Hyperparameter Tuning

BaselineNet dengan input 64x64, dan di-*reshape* 1/255.0, dan hyperparameter model yang digunakan adalah 20 untuk *epoch*, *optimizer* ADAM, *batch-size* 64, dan *learning-rate* 0,001. Model dihitung menggunakan *accuracy*,

precision , recall, f1-score, MCC dan percobaan diulangi sebanyak 5 kali. Kemudian dibandingkan dengan model dari Best Hyperparameter yang telah didapatkan melalui hyperparameter tuning pada BaselineNet menggunakan *Grid Search* yaitu *optimizer* SGD, *learning-rate* 0,1, *batch-size* 32, dan *epoch* 20. Kemudian model dari best hyperparameter akan dievaluasi menggunakan *confusion matrix* yang menghasilkan nilai *accuracy, precision , recall, f1-score,* MCC dan perhitungan model diulangi sebanyak 5 kali. Hasil percobaan dapat dilihat pada tabel 5.2.

Tabel 5.2. Hasil *Accuracy, Precision , Recall, F1-score,* dan MCC pada arsitektur BaselineNet dan Best Hyperparameter BaselineNet

Percobaan		Accuracy	Precision	Recall	F1-score	MCC
BaselineNet	1	0,9495	0,958	0,9407	0,9492	0,899
	2	0,9495	0,9416	0,9598	0,9506	0,8992
	3	0,955	0,9378	0,9641	0,9559	0,9101
	4	0,9574	0,9475	0,9699	0,9585	0,9153
	5	0,9513	0,9431	0,962	0,9524	0,9029
Rata-rata		**0,9525**	**0,9456**	**0,9593**	**0,9533**	**0,9053**
BaselineNet Best Hyperparameter	1	0,9539	0,9561	0,9527	0,9544	0,9078
	2	0,9575	0,945	0,9727	0,9587	0,9154
	3	0,9524	0,962	0,9434	0,9526	0,9051
	4	0,96	0,9445	0,9785	0,9612	0,9207
	5	0,9568	0,9388	0,9785	0,9582	0,9144
Rata-rata		**0,9561**	**0,9493**	**0,9652**	**0,9570**	**0,9127**

Pada arsitektur BaselineNet perlakuan tuning pada hyperparameter menggunakan Grid Search memberikan performa *accuracy* sebesar 95,61% yang meningkat 0,36% dibandingkan tanpa hyperparameter tuning. Performa *precision* pada perlakuan hyperparameter tuning menggunakan Grid Search meningkat sebesar 0,37% dibandingkan tanpa hyperparameter tuning. Performa *recall* pada perlakuan hyperparameter tuning menggunakan Grid Search meningkat sebesar 0,59% dibandingkan tanpa hyperparameter tuning. Performa *f1-score* pada perlakuan hyperparameter tuning menggunakan Grid Search meningkat sebesar 0,37% dibandingkan tanpa hyperparameter tuning. Kemampuan model mengenali prediksi negatif yang terdapat pada MCC di perlakuan hyperparameter tuning

menggunakan Grid Search meningkat sebesar 0,74% dibandingkan tanpa hyperparameter tuning.

5.7 Perbandingan Keseluruhan Model CNN

Hasil hyperparameter terbaik dari *Grid Search* arsitektur Rajaraman dan BaselineNet kemudian dibandingkan dengan Rajaraman dan BaselineNet tanpa *hyperparameter tuning* atau tanpa *Grid Search.* Hasil perbandingan best hyperparameter dengan tanpa pencarian *Grid Search* dapat dilihat pada tabel 5.3.

Tabel 5.3. Hasil *Accuracy, Precision , Recall, F1-score,* dan MCC pada model tanpa Grid Search dan Best Hyperparameter dari Grid Search

Perlakuan	Arsitektur	Accuracy	Precision	Recall	F1-score	MCC
Tanpa Grid Search	BaselineNet	0,9513	0,9391	0,9651	0,9519	0,9031
	Rajaraman	0,9506	0,9269	0,9781	0,9519	0,9026
Rata-rata		**0,9509**	**0,9330**	**0,9716**	**0,9519**	**0,9028**
Dengan Grid Search	BaselineNet	0,9578	0,9461	0,9709	0,9538	0,9161
	Rajaraman	0,9658	0,9598	0,9723	0,9660	0,9318
Rata-rata		**0,9618**	**0,9529**	**0,9716**	**0,9599**	**0,9239**

Berdasarkan tabel 5.3 diatas, dapat diketahui bahwa hyperparameter tuning meningkatkan *accuracy* sebesar 1,08% dibandingkan tanpa menggunakan Hyperparameter Tuning. Performa *precision* pada perlakuan hyperparameter tuning menggunakan Grid Search meningkat sebesar 1,99% dibandingkan tanpa hyperparameter tuning. Performa *recall* pada perlakuan hyperparameter tuning menggunakan Grid Search tidak mengalami peningkatan akan tetapi range error lebih sedikit dibandingkan tanpa hyperparameter tuning. Performa *f1-score* pada perlakuan hyperparameter tuning menggunakan Grid Search meningkat sebesar 0,08% dibandingkan tanpa hyperparameter tuning. Kemampuan model mengenali prediksi negatif yang terdapat pada MCC di perlakuan hyperparameter tuning menggunakan Grid Search meningkat sebesar 2,11% dibandingkan tanpa hyperparameter tuning.

5.8 Uji Signifikansi

Penelitian ini menggunakan uji signifikansi untuk mengetahui pengaruh perlakuan tuning pada model CNN. Pengujian dilakukan menggunakan Uji *Normalitas Shapiro-Wilk* untuk menentukan data berdistribusi normal atau tidak. Untuk data berdistribusi normal dilakukan Uji *Homogenitas Lavene* untuk menentukan data homogen atau tidak dan Uji *Signifikansi Parametrik Independent Samples T-test.* Sedangkan untuk data berdistribusi tidak normal menggunakan Uji *Signifikansi Nonparametrik Mann-Whitney.* Uji *Signifikansi* dilakukan untuk mengetahui pengaruh *hyperparameter tuning* menggunakan *Grid Search* pada model CNN. Hasil pengujian Uji Nomalitas, Uji Parametrik, dan Uji Nonparamterik dapat dilihat pada gambar 5.7, 4.8, dan 4.9.

Tests of Normality

	Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Statistic	df	Sig.	Statistic	df	Sig.
Accuracy	.140	20	.200*	.928	20	.142
Precission	.133	20	.200*	.964	20	.620
Recall	.181	20	.085	.851	20	.005
F1score	.120	20	.200*	.944	20	.290
Mcc	.129	20	.200*	.933	20	.179

*. This is a lower bound of the true significance.

a. Lilliefors Significance Correction

Gambar 5.7 Hasil Uji Normalitas menggunakan Shapiro-Wilk (dikotak merah)

Dari hasil pengujian Uji Normalitas menggunakan Shapiro-Wilk dapat diambil kesimpulan bahwa Data *Accuracy, Precission, F1-Score,* dan MCC memiliki nilai Sig > 0.05 sehingga gagal menolak H0 yang berarti data berdistribusi normal. Sedangkan Data pada *Recall* memiliki nilai Sig < 0.05 sehingga menolak H0 yang berarti data tidak berdistribusi normal. Kemudian pada data *Accuracy, Precission, F1-Score,* dan MCC dilakukan Uji *Parametrik* yang hasilnya terdapat pada gambar 5.8 dan data *Recall* dilakukan Uji *Nonparametrik* yang hasilnya terdapat pada gambar 5.9.

Independent Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means							
						Significance		Mean	Std. Error	95% Confidence Interval of the Difference	
		F	Sig.	t	df	One-Sided p	Two-Sided p	Difference	Difference	Lower	Upper
Accuracy	Equal variances assumed	5.281	.034	-2.798	18	.006	.012	-.0064800	.0023160	-.0113458	-.0016142
	Equal variances not assumed			-2.798	15.544	.007	.013	-.0064800	.0023160	-.0114015	-.0015585
Precission	Equal variances assumed	.309	.585	-3.153	18	.003	.005	-.0115100	.0036502	-.0191789	-.0038411
	Equal variances not assumed			-3.153	17.993	.003	.006	-.0115100	.0036502	-.0191791	-.0038409
F1score	Equal variances assumed	3.264	.088	-2.584	18	.009	.019	-.0061800	.0023915	-.0112044	-.0011556
	Equal variances not assumed			-2.584	16.138	.010	.020	-.0061800	.0023915	-.0112463	-.0011137
Mcc	Equal variances assumed	4.391	.051	-2.710	18	.007	.014	-.0126800	.0046789	-.0225101	-.0028499
	Equal variances not assumed			-2.710	15.842	.008	.016	-.0126800	.0046789	-.0226069	-.0027531

Gambar 5.8 Hasil Uji Homogenitas dan Uji Signifinaksi Parametrik menggunakan Lavene dan Independent Samples T-test

Test Statistics[a]

	Recall
Mann-Whitney U	49.000
Wilcoxon W	104.000
Z	-.076
Asymp. Sig. (2-tailed)	.940
Exact Sig. [2*(1-tailed Sig.)]	.971[b]

a. Grouping Variable: Perlakuan

b. Not corrected for ties.

Gambar 5.9 Hasil Uji Signifinaksi Nonparametrik Mann-Whitney

Dari hasil pengujian Uji *Parametrik* pada gambar 5.8 dapat dilihat saat melakukan uji Homogenitas menggunakan Lavene pada data *Accuracy* memiliki nilai Sig < 0.05 sehingga menolak H0 yang berarti variansi data tidak homogen sehingga Uji *Signifikansi Independent Samples T-test* menggunakan nilai kedua yaitu 0.013 yang kurang dari 0.05 sehingga menolak H0 yang berarti *Hyperparameter Tuning* menggunakan *Grid Search* memiliki pengaruh signifikan. Pada data *Precission, F1-score,* dan MCC dilakukan uji Homogenitas menggunakan Lavene memiliki nilai sig 0.585, 0.088, dan 0.051 yang berarti nilai Sig > 0.05 sehingga menerima H0 atau variansi data homogen. Kemudian data *Precission, F1-score,* dan MCC dilakukan uji Signifikansi Independent Samples T-test menghasilkan nilai sig 0.005, 0.019, dan 0.14 atau nilai sig < 0.05 sehingga menolak H0 yang artinya Hyperparameter Tuning menggunakan Grid Search memiliki pengaruh signifikan. Dari hasil pengujian *Nonparametrik* pada gambar 5.9 dapat dilihat dari Uji *Signifikansi Mann-Whitney* menghasilkan nilai sig 0.759 > 0.05 yang

berarti Hyperparameter Tuning menggunakan Grid Search tidak berpengaruh secara signifikan pada metrik *recall.*

5.9 Penelitian Tambahan

Berdasarkan tinjauan pustaka yang telah dilakukan dan analisa hasil yang didapatkan saat melakukan pencarian hyperparameter maka peneliti melakukan penelitian tambahan untuk memvalidasi hasil yang telah dilakukan sesuai dengan teori yang ada pada tinjauan pustaka. Penelitian ini menggunakan BaselineNet tanpa hyperparameter tuning sebagai model validasi percobaan yang akan dilakukan. Model BaselineNet tanpa hyperparameter tuning menggunakan hyperparameter *epoch* 20, *optimizer* ADAM, *batch-size* 64, dan *learning-rate* 0,001.

5.9.1 Pengaruh learning-rate pada model CNN arsitektur BaselineNet

Percobaan *learning-rate* pada model BaselineNet dengan menguji pengaruh *learning-rate* menggunakan nilai dengan range (0,1 - 0,00001). Percobaan yang akan dilakukan dapat dilihat pada tabel 5.4.

Tabel 5.4. Hyperparameter percobaan *learning-rate* menggunakan BaselineNet

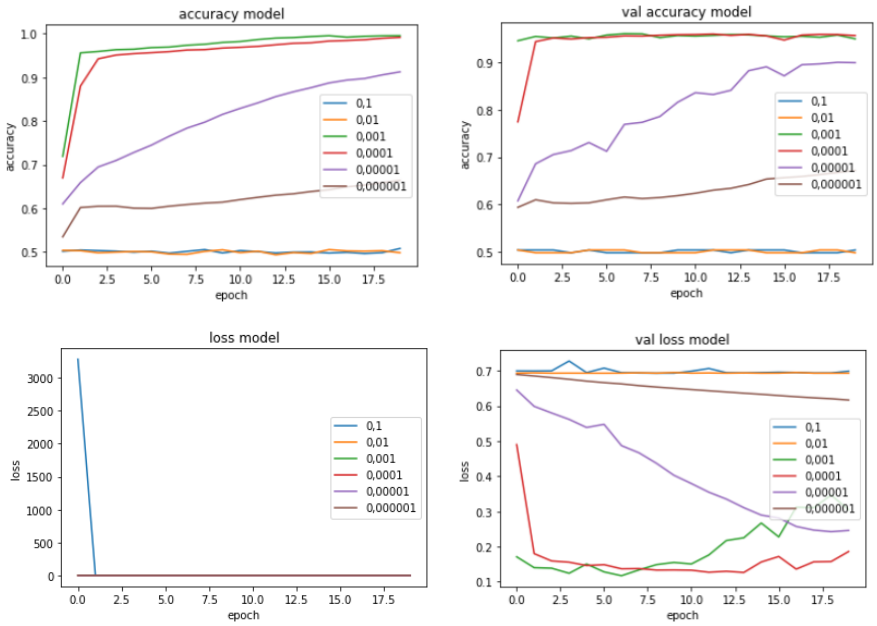
No	Arsitektur	Input Resolution	Learning Rate	Optimizer	Batch-Size	Epochs
1	BaselineNet	64 x 64	0,1	ADAM	64	20
2	BaselineNet	64 x 64	0,01	ADAM	64	20
3	BaselineNet	64 x 64	0,001	ADAM	64	20
4	BaselineNet	64 x 64	0,0001	ADAM	64	20
5	BaselineNet	64 x 64	0,00001	ADAM	64	20
6	BaselineNet	64 x 64	0,000001	ADAM	64	20

Kemudian percobaan dievaluasi dengan *confusion matrix* yang kemudian menghasilkan nilai *accuracy*, *precision* , *recall, f1-score*, dan MCC. Hasil percobaan dapat dilihat pada tabel 5.5.

Tabel 5.5. Hasil *accuracy, precision, recall, f1-score, dan* MCC dari *confusion matrix* pada percobaan *learning-rate*

Learning-Rate	Accuracy	Precision	Recall	F1-score	MCC
0,1	0,4934	0,0000	0,0000	0,0000	0,0000
0,01	0,5065	0,5065	1	0,6724	0,0000
0,001	**0,9531**	**0,9554**	0,952	0,9537	0,9063
0,0001	**0,9531**	0,9408	**0,9684**	**0,9544**	**0,9067**
0,00001	0,8998	0,8603	0,9577	0,9064	0,8046
0,000001	0,6563	0,6513	0,6919	0,671	0,3127

Semakin rendah nilai *learning-rate* tidak selalu meningkatkan akurasi, hal ini dibuktikan dengan *learning-rate* 0,00001 yang memiliki akurasi 65,63% lebih rendah dari *learning-rate* 0,001. Pada *metrix precision* memiliki nilai paling tinggi 95,54 pada *learning-rate* 0,001. *Metrix recall* memiliki nilai paling tinggi 96,84 pada *learning-rate* 0,0001. *Metrix f1-score* memiliki nilai paling tinggi 95,54 pada *learning-rate* 0,0001. *Metrix* MCC memiliki nilai paling tinggi 90,67% pada *learning-rate* 0,0001. Berdasarkan nilai tertinggi maka model BaselineNet memiliki model berperforma paling optimal pada *learning-rate* 0,0001. Grafik *history accuracy* dan *loss* dapat dilihat pada gambar 5.10.



Gambar 5.10 Grafik history *accuracy, val accuracy, loss,* dan *val loss* model pada percobaan *learning-rate*

Pada grafik *history val loss* dapat dilihat bahwa model mengalami *overfitting* pada *learning-rate* 0.1 , 0.01, 0.001, 0.00001, dan 0.000001. Sedangkan pada *learning-rate* 0,0001 model sudah optimal. Ketika model kurang dari nilai *learning-rate* yang optimal model berhenti di *local optimum* sedangkan ketika model lebih tinggi dari nilai *learning-rate* yang optimal model mengalami *overfitting*.

5.9.2 Pengaruh epoch pada model CNN arsitektur BaselineNet

Percobaan *epochs* pada model BaselineNet dengan menguji pengaruh *epochs* menggunakan nilai dengan range (10 – 320). Percobaan yang akan dilakukan dapat dilihat pada tabel 5.6.

Tabel 5.6. Hyperparameter percobaan epochs menggunakan BaselineNet

No	Arsitektur	Input Resolution	Learning Rate	Optimizer	Batch-Size	Epochs
1	BaselineNet	64 x 64	0,001	ADAM	64	10
2	BaselineNet	64 x 64	0,001	ADAM	64	20
3	BaselineNet	64 x 64	0,001	ADAM	64	40
4	BaselineNet	64 x 64	0,001	ADAM	64	80
5	BaselineNet	64 x 64	0,001	ADAM	64	160
6	BaselineNet	64 x 64	0,001	ADAM	64	320

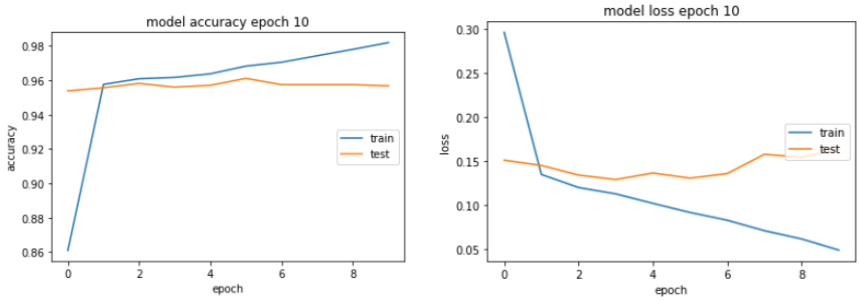
Kemudian percobaan dievaluasi dengan *confusion matrix* yang kemudian menghasilkan nilai *accuracy, precision , recall, f1-score,* dan MCC. Hasil percobaan dapat dilihat pada tabel 5.7.

Tabel 5.7. Hasil *accuracy, precision, recall, f1-score,* dan MCC dari *confusion matrix* pada percobaan *epoch*

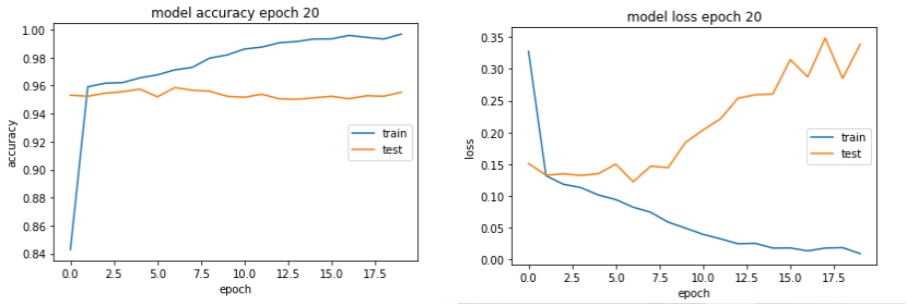
Epoch	Accuracy	Precision	Recall	F1-score	MCC
10	**0,9589**	**0,9578**	0,9613	**0,9595**	**0,9179**
20	0,9528	0,9414	**0,9670**	0,9540	0,9059
40	0,9557	0,9479	0,9656	0,9567	0,9115
80	0,9535	0,9446	0,9648	0,9546	0,9072
160	0,9542	0,9446	0,9663	0,9553	0,9087
320	0,9499	0,9498	0,9627	0,9511	0,9

Berdasarkan percobaan yang telah dilakukan nilai *accuracy* tertinggi sebesar 95,89% pada yang *epoch* 10. *Metrix precision* memiliki nilai paling

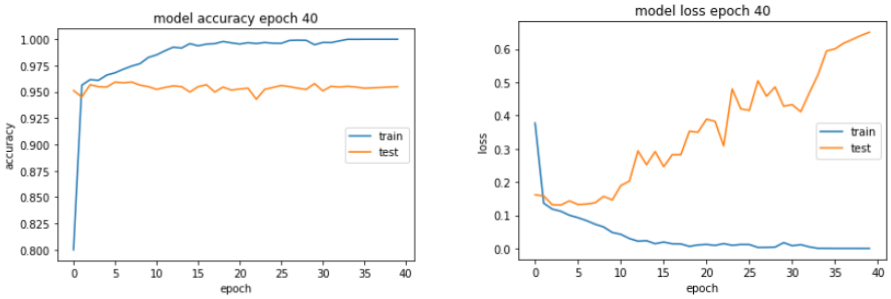
tinggi 0,9578 pada *epoch* 10. *Metrix recall* memiliki nilai paling tinggi 0,9670 pada *epoch 20*. *Metrix f1-score* memiliki nilai paling tinggi 0,9595 pada *epoch* 10. *Metrix* MCC memiliki nilai paling tinggi 0,9179 pada *epoch 10*. Berdasarkan nilai tertinggi maka arsitektur BaselineNet memiliki model berperforma paling optimal pada epoch 10. Grafik *history accuracy* dan *loss* pada epoch 10, 20, 40, 80, 160, dan 320 dapat dilihat pada gambar 5.11, gambar 5.12, gambar 5.13, gambar 5.14, gambar 5.15 dan gambar 5.16.



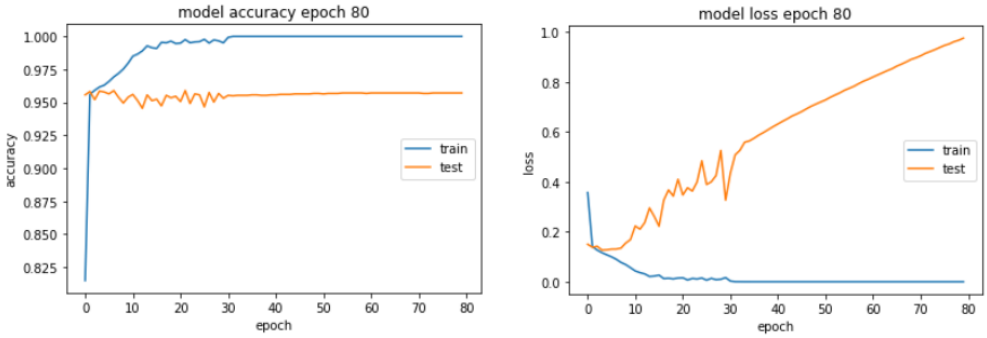
Gambar 5.11 Grafik *history accuracy*, *val accuracy*, *loss*, dan *val loss* model pada percobaan epoch 10



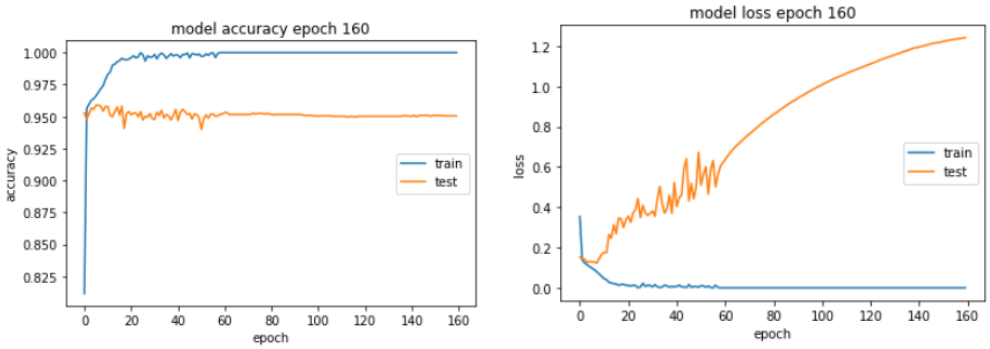
Gambar 5.12 Grafik *history accuracy*, *val accuracy*, *loss*, dan *val loss* model pada percobaan epoch 20



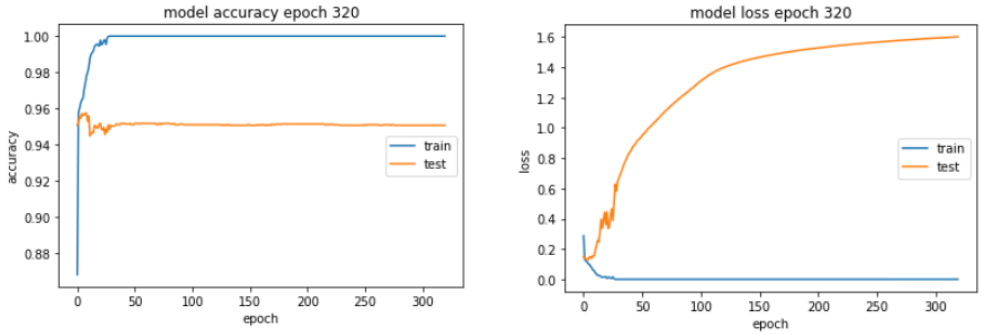
Gambar 5.13 Grafik *history accuracy*, *val accuracy*, *loss*, dan *val loss* model pada percobaan epoch 40



Gambar 5.14 Grafik *history accuracy*, *val accuracy*, *loss*, dan *val loss* model pada percobaan epoch 80



Gambar 5.15 Grafik *history accuracy*, *val accuracy*, *loss*, dan *val loss* model pada percobaan epoch 160



Gambar 5.16 Grafik *history accuracy*, *val accuracy*, *loss*, dan *val loss* model pada percobaan epoch 320

Berdasarkan grafik *history accuracy* dan *loss*, dapat disimpulkan ketika model mengalami *overfitting* dengan menambahkan nilai epoch akan membuat model terus mengalami *overfitting*. Oleh karena itu, pengurangan nilai epoch akan membuat model dari *overfitting* menuju optimal, akan tetapi dalam beberapa kasus mengurangi epoch tidak cukup untuk membebaskan

model dari *overfitting* sehingga harus dilakukan proses lain seperti melakukan regularisasi atau melakukan tuning pada hyperparameter lain seperti bacth-size.

5.9.3 Pengaruh batch-size pada model CNN arsitektur BaselineNet

Percobaan *batch-size* pada model BaselineNet dengan menguji pengaruh *batch-size* menggunakan nilai dengan range (16 – 512). Percobaan yang akan dilakukan dapat dilihat pada tabel 5.8.

Tabel 5.8. Hyperparameter percobaan *batch-size* menggunakan BaselineNet

No	Arsitektur	Input Resolution	Learning Rate	Optimizer	Batch-Size	Epochs
1	BaselineNet	64 x 64	0,001	ADAM	16	20
2	BaselineNet	64 x 64	0,001	ADAM	32	20
3	BaselineNet	64 x 64	0,001	ADAM	64	20
4	BaselineNet	64 x 64	0,001	ADAM	128	20
5	BaselineNet	64 x 64	0,001	ADAM	256	20
6	BaselineNet	64 x 64	0,001	ADAM	512	20

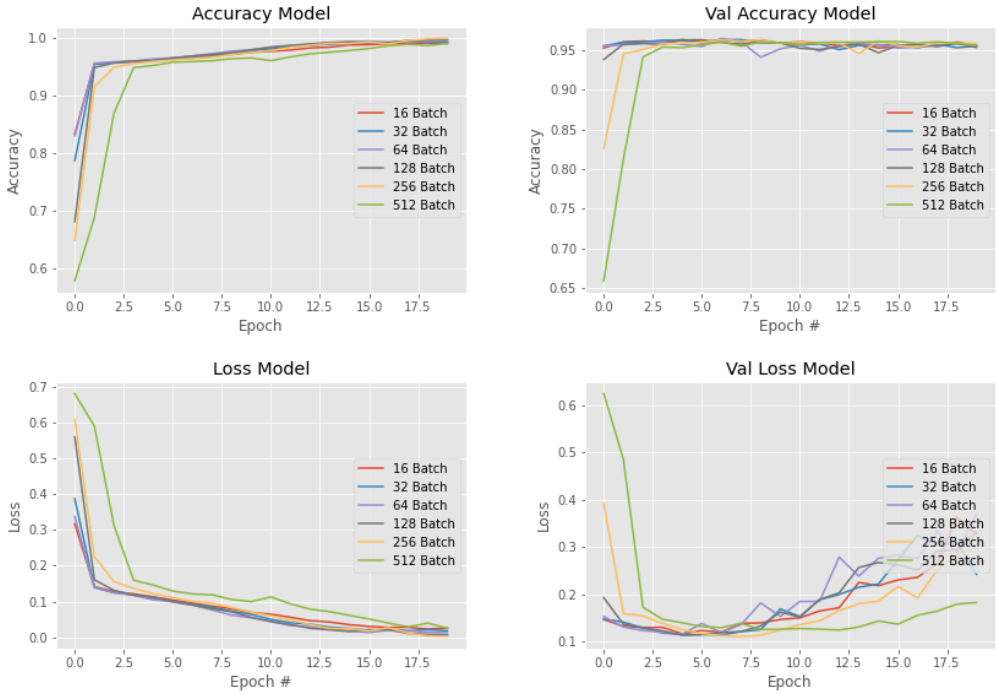
Kemudian percobaan dievaluasi dengan *confusion matrix* yang kemudian menghasilkan nilai *accuracy*, *precision* , *recall, f1-score*, dan MCC. Hasil percobaan dapat dilihat pada tabel 5.9.

Tabel 5.9. Hasil *accuracy, precision, recall, f1-score,* dan MCC dari *confusion matrix* pada percobaan *batch-size*

Batch-size	Accuracy	Precision	Recall	F1-score	MCC
16	0,9582	0,9558	0,9606	0,9582	0,9165
32	**0,9597**	0,9592	**0,9699**	**0,9596**	**0,9194**
64	**0,9597**	0,9592	**0,9699**	**0,9596**	**0,9194**
128	0,9531	0,9600	0,9454	0,9526	0,9064
256	0,9539	0,9541	0,9534	0,9537	0,9078
512	0,9546	**0,9636**	0,9446	0,954	0,9094

Berdasarkan percobaan yang telah dilakukan nilai *accuracy* tertinggi sebesar 95,97% pada *batch-size 32 dan 64*. *Metrix precision* memiliki nilai paling tinggi 0,9636 pada *batch-size 512*. *Metrix recall* memiliki nilai paling tinggi 0,9699 pada *batch-size* 32 dan 64. *Metrix f1-score* memiliki nilai paling tinggi 0,9596 pada *batch-size* 32 dan 64. *Metrix* MCC memiliki nilai paling tinggi 0,9194 pada *batch-size* 32 dan 64. Berdasarkan nilai tertinggi

maka arsitektur BaselineNet memiliki model berperforma paling optimal pada *batch-size* 32 dan 64. Grafik *history accuracy* dan *loss* dapat dilihat pada gambar 5.17.



Gambar 5.17 Grafik *history accuracy, val accuracy, loss,* dan *val loss* model pada percobaan batch-size

Berdasarkan *history loss* model bergerak dari *overfitting* ke model optimal, sehingga dapat disimpulkan apabila model mengalami *overfitting* maka nilai *batch-size* dapat ditingkatkan sehingga mendapatkan model optimal.

5.9.4 Pengaruh optimizer pada model CNN arsitektur BaselineNet

Percobaan *optimizer* pada model BaselineNet dengan menguji pengaruh *optimizer* menggunakan ADAM, SGD, NADAM, ADAMAX, RMSProp, dan ADAGRAD. Percobaan yang akan dilakukan dapat dilihat pada tabel 5.10.

Tabel 5.10. Hyperparameter percobaan *optimizer* menggunakan BaselineNet

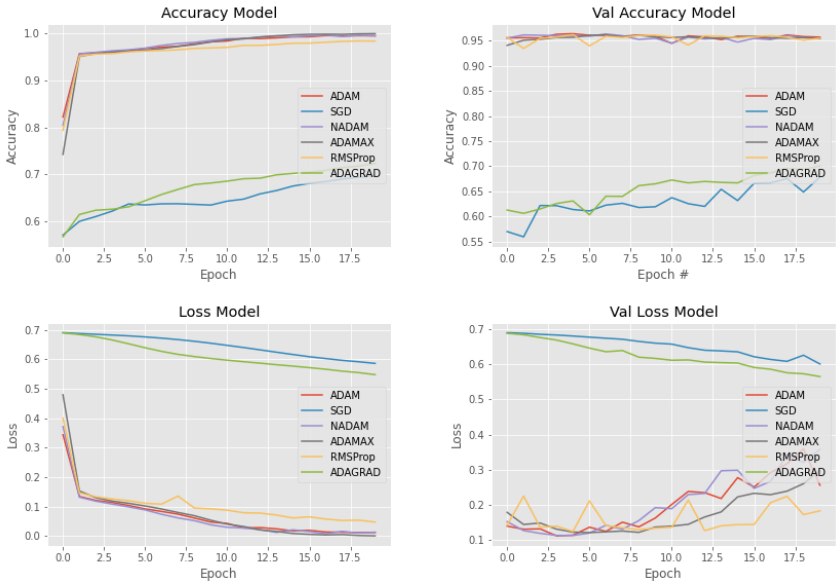
No	Arsitektur	Input Resolution	Learning Rate	Optimizer	Batch-Size	Epochs
1	BaselineNet	64 x 64	0,001	ADAM	64	20
2	BaselineNet	64 x 64	0,001	SGD	64	20
3	BaselineNet	64 x 64	0,001	NADAM	64	20
4	BaselineNet	64 x 64	0,001	ADAMAX	64	20
5	BaselineNet	64 x 64	0,001	RMSProp	64	20
6	BaselineNet	64 x 64	0,001	ADAGRAD	64	20

Kemudian percobaan dievaluasi dengan *confusion matrix* yang kemudian menghasilkan nilai *accuracy*, *precision* , *recall*, *f1-score*, dan MCC. Hasil percobaan dapat dilihat pada tabel 5.11.

Tabel 5.11. Hasil *accuracy*, *precision, recall, f1-score*, dan MCC dari *confusion matrix* pada percobaan *optimizer*

Optimizer	Accuracy	Precision	Recall	F1-score	MCC
ADAM	0,96	**0,964**	0,9556	**0,9597**	**0,9202**
SGD	0,6955	0,7087	0,6746	0,6912	0,3944
NADAM	**0,9682**	0,9558	0,9606	0,9582	0,9165
ADAMAX	0,955	0,9426	**0,9687**	0,9554	0,9103
RMSProp	0,9542	0,9581	0,9497	0,9539	0,9085
ADAGRAD	0,7206	0,695	0,7831	0,7364	0,444

Berdasarkan percobaan yang telah dilakukan nilai *accuracy* tertinggi sebesar 96,82% pada *optimizer* NADAM. *Metrix precision* memiliki nilai paling tinggi 0,9640 pada *optimizer* ADAM. *Metrix recall* memiliki nilai paling tinggi 0,9687 pada *optimizer* ADAMAX.. *Metrix f1-score* memiliki nilai paling tinggi 0,9597 pada *optimizer* ADAM.. *Metrix* MCC memiliki nilai paling tinggi 0,9202 pada *optimizer* ADAM.. Berdasarkan jumlah nilai tertinggi maka arsitektur BaselineNet memiliki model berperforma paling optimal pada *optimizer* ADAM. Grafik *history accuracy* dan *loss* dapat dilihat pada gambar 5.18.



Gambar 5.18 Grafik *history accuracy, val accuracy, loss,* dan *val loss* model pada percobaan *optimizer*

Berdasarkan *history accuracy* dan *loss*, *optimizer* SGD dan ADAGRAD mengalami *underfitting* dan terjebak pada *local optimum*. Sedangkan *optimizer* ADAM, NADAM, ADAMAX dan RMSProp memiliki pergerakan *accuracy* yang stabil.

.

Bab VI
Penutup

6.1 Kesimpulan

Berdasarkan penelitian yang dilakukan, penelitian ini telah berhasil mendapatkan *best hyperparameter* pada setiap akurasi dan penggunaan hyperparameter tuning menggunakan *Grid Search* memiliki pengaruh pada peningkatan akurasi. Pencarian *hyperparameter* tidak mempertimbangkan mengalami *overfitting*, atau *underfitting*. Sehingga harus dilakukan evaluasi pada *best hyperparameter* sebelum digunakan di semua model Deep Learning CNN pada arsitektur BaselineNet dan Rajaraman. Hasil penelitian ini, mengemukakan bahwa *hyperparameter tuning* dapat meningkatkan akurasi 0,36% dan 0,94% pada arsitektur BaselineNet dan Rajaraman. Berdasarkan uji *Signifikansi* menghasilkan nilai Sig 0.013 < 0.05 sehingga *Hyperparamater Tuning* memiliki pengaruh signifikan pada *accuracy* model CNN. Nilai *hyperparameter* terbaik yang didapatkan dari *Grid Search* pada Arsitektur BaselineNet *optimizer* SGD, *learning-rate* 0,1, *batch-size* 32, dan *epoch* 20 dan Rajaraman *optimizer* ADAMAX, *learning-rate* 0,001, *batch-size* 32, dan *epoch* 30 menghasilkan *accuracy 0,9578* dan *0,9658.*

Berdasarkan percobaan tambahan yang telah dilakukan dapat disimpulkan bahwa ketika model mengalami *overfitting* maka *epoch* harus diturunkan dan *batch-size* harus dinaikkan. Akan tetapi, pengurangan *epoch* dan peningkatan *batch-size* tidak selalu memperbaiki model yang mengalami *overfitting* sehingga perbaikan mode bisa menggunakan metode lain seperti regularisasi. *Learning-rate* membantu model keluar dari *local optimum* akan tetapi ketika nilai *learning-rate* melewati *global optimum* maka model akan terjebak kedalam *local optimum* dan mengalami *overfitting*.

6.2 Saran

Saran untuk Penelitian selanjutnya dapat melakukan pengujian pada hyperparameter yang berbeda seperti filter, stride, dan momentum. Pencarian dengan rentang area pencarian yang berbeda dapat membantu memvalidasi penelitian ini, sekaligus menambahkan pengetahuan tambahan yang dapat memperkuat hipotesis mengenai pengaruh *hyperparamater* tuning menggunakan

Grid Search. Selain itu, pengoptimalan model bisa menggunakan metode pencarian yang lain seperti Bayesian Optimization, atau menambahkan algoritma lain pada arsitektur CNN. Peningkatan akurasi pada Deep Learning untuk peneliti selanjutnya dapat menggunakan metode lain seperti Data Augmentation, Transfer Learning, dan Regularisasi.

Untuk yang ingin menduplikasi hasil penelitian ini, ataupun untuk yang mencoba percobaan yang sama disarankan menggunakan akses TPU pada Google Colab Pro+ atau yang alternatif lain dengan batasan runtime diatas 12 Jam. Penggunaan Grid Search Cross Validation pada model keras tidak disarankan dikarenakan batasan runtime pada Google Colab menggunakan TPU dan kegagalan runtime pada Google Colab menggunakan GPU. Dengan data lebih dari 20.000 membuat pencarian hyperparameter menggunakan Grid Search Cross Validation membutuhkan waktu lebih dari 10 jam. Sehingga apabila memiliki sumber daya yang mampu menjalankan Grid Search Cross Validation tanpa mengalami kegagalan runtime disarankan untuk menggunakan library Halving Grid Search Cross Validation yang mampu mengurangi waktu pencarian diatas 30%.

DAFTAR PUSTAKA

Algorit.ma, 2019. *Memahami Convolutional Neural Network menggunakan TensorFlow.* [Online] Available at: https://algorit.ma/blog/convolutional-neural-networks-tensorfflow/ [Diakses 29 November 2021].

ALOMEDIKA, 2020. *Analisis mikroskopis apus darah tepi.* [Online] Available at: https://www.alomedika.com/tindakan-medis/hematologi-dan-onkologi/apus-darah-tepi/analisis-mikroskopis [Diakses 3 Desember 2021].

Bian, B. J., 2005. Diagnosis from the blood smear. *NEJM,* Volume 353, pp. 498-507.

Calderaro, A. et al., 2021. Malaria diagnosis in non-endemic setting : The European exprerience in the last 22 years. *Microorganisms,* 9(11), p. 2265.

Dahwan, D., 2021. *Jenis-jenis machine learning.* [Online] Available at: https://medium.com/sysinfo/jenis-jenis-machine-learning-50f918453336 [Diakses 3 Desember 2021].

Google Cloud AI Platform, 2021. *Overview of hyperparameter tuning.* [Online] Available at: https://cloud.google.com/ai-platform/training/docs/hyperparameter-tuning-overview [Diakses 29 11 2021].

Harahab, M. et al., 2021. Implementation of Convolutional Neural Network in the classification of red blood cell have affected of malaria. *Sinkron : jurnal dan penelitian informatika,* 5(2), pp. 199-207.

Jaiswal, M., Sahu, A. & Zafar, M. T., 2020. Effectively diagnosing malaria by optimizing the hyperparameters of CNN using genetic algorithm on the multi core GPU. *IJRT: International Journal of Recent Technology and Engineering,* 8(6).

Koehrsen, W., 2018. *a conceptuan explanation of bayesian hyperparameter optimization for machine learning.* [Online] Available at: https://towardsdatascience.com/a-conceptual-explanation-of-

bayesian-model-based-hyperparameter-optimization-for-machine-learning-b8172278050f
[Diakses 4 Desember 2021].

Kurniawan, K., 2018. *Bagaimana anda menjelaskan stochastic gradient descent dan kaitannya dengan regresi linier.* [Online] Available at: https://id.quora.com/Bagaimana-anda-menjelaskan-Stochastic-Gradient-Descent-dan-kaitannya-dengan-regresi-linear
[Diakses 10 Desember 2021].

Machine Learning Mastery, 2021. *Gradient descent optimization with adamax from scratch.* [Online] Available at: https://machinelearningmastery.com/gradient-descent-optimization-with-adamax-from-scratch/
[Diakses 4 Desember 2021].

Malik, F., 2020. *What is grid search? explaining how to obtain optimal hyperparameter values.* [Online] Available at: https://medium.com/fintechexplained/what-is-grid-search-c01fe886ef0a
[Diakses 29 November 2021].

Pattanaik, P. A., Mittal, M., Khan, M. Z. & Panda, S. N., 2020. Malaria derection using deep residual networks with mobile microscopy. *Journal of King Saud University - Computer and Information Sciences.*

Poostchi, M., 2018. Image analysis and machine learning for detecting malaria. *ELSEVIER*, Volume 194, pp. 36-55.

Rajaraman, S., Jaeger, S. & Antani, , S. K., 2019. *Performance evaluation of deep neural ensembles toward malaria detection in thin-blood smear images.* s.l., PeerJ.

Rajaraman, S. et al., 2018. Understanding the learned behavior of custumized convolutional neural network toward malaria parasite detection in thin blood smear images. *Journal of Medical Imaging,* 5(3), p. 034501.

Sena, S., 2017. *pengenalan deep learning part 7 : Convolutional Neural Network (CNN).* [Online] Available at: https://medium.com/@samuelsena/pengenalan-deep-learning-

part-7-convolutional-neural-network-cnn-b003b477dc94
[Diakses 29 November 2021].

Sena, S., 2017. *Pengenalan Deep Learning Part 7 : Convolutional Neural Network (CNN) | by Samuel Sena | Medium.* [Online] Available at: https://medium.com/@samuelsena/pengenalan-deep-learning-part-7-convolutional-neural-network-cnn-b003b477dc94
[Diakses 27 November 2020].

SKILLPLUS, 2019. *Gradient descent and learning rate lanjutan.* [Online] Available at: https://skillplus.web.id/gradient-descent-dan-learning-rate-lanjutan/
[Diakses 3 12 2020].

Suriya, M., Chandran, V. & Sumithra, M. G., 2019. Enhanced deep convolutional neural network for malaria parasite classification.. *International Journal of Computers and Application.*

Swastika, W., Kristiani, G. M. & Widodo, R. B., 2021. Effective preprocessed thin blood bmear images to improve malaria parasite detection using deep learning. *Journal of Physics: Conference Series,* 1896(1), p. 012092.

Tilawah, S., 2020. *Adam optimizier.* [Online] Available at: https://medium.com/@saritilawah9/adam-optimizer-80cc267522af
[Diakses 3 Desember 2021].

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

Yi, S., Xiaogang, W. & Xiaoou, T., 2016. *Sparsifying neural network connections for face recognition.* Las Vegas: Computer Vision and Pattern Recognition.

Yuhang, D. et al., 2017. Evaluations of deep convolutional neural networks for automatic identification of malaria infected cells. *IEEE EMBS International Conference on Biomedical & Health Informatics (BHI),* pp. 101-104.

Zhao, O. et al., 2020. Convolutional neural networks to atumate the screening of malaria in low-resource countries. *PeerJ,* Issue 8.

Ziaee, M. & Abedi, F., 2014. Severe falciparum malaria in iran: a very rare case from an endemic region. *Jundishapur journal of microbiology,* 7(1), p. e8752.

LAMPIRAN

A. Hasil Percobaan Grid Search Rajaraman

learning-rate	*optimizer*	*batch-size*	*epoch*	*Accuracy*
0,1	ADAM	16	10	0,5022
0,1	ADAM	16	20	0,5022
0,1	ADAM	16	30	0,4978
0,1	ADAM	16	40	0,4978
0,1	ADAM	24	10	0,5022
0,1	ADAM	24	20	0,4978
0,1	ADAM	24	30	0,5022
0,1	ADAM	24	40	0,4978
0,1	ADAM	32	10	0,4978
0,1	ADAM	32	20	0,5022
0,1	ADAM	32	30	0,4978
0,1	ADAM	32	40	0,5022
0,1	SGD	16	10	0,5062
0,1	SGD	16	20	0,5450
0,1	SGD	16	30	0,4978
0,1	SGD	16	40	0,4978
0,1	SGD	24	10	0,5022
0,1	SGD	24	20	0,5000
0,1	SGD	24	30	0,5134
0,1	SGD	24	40	0,5022
0,1	SGD	32	10	0,5978
0,1	SGD	32	20	0,5359
0,1	SGD	32	30	0,4982
0,1	SGD	32	40	0,4978
0,1	ADAMAX	16	10	0,4978
0,1	ADAMAX	16	20	0,4978

0,1	ADAMAX	16	30	0,5022
0,1	ADAMAX	16	40	0,5022
0,1	ADAMAX	24	10	0,4978
0,1	ADAMAX	24	20	0,5022
0,1	ADAMAX	24	30	0,4978
0,1	ADAMAX	24	40	0,4978
0,1	ADAMAX	32	10	0,4978
0,1	ADAMAX	32	20	0,4978
0,1	ADAMAX	32	30	0,4978
0,1	ADAMAX	32	40	0,5022
0,01	ADAM	16	10	0,5044
0,01	ADAM	16	20	0,5044
0,01	ADAM	16	30	0,4956
0,01	ADAM	16	40	0,4956
0,01	ADAM	24	10	0,4956
0,01	ADAM	24	20	0,4956
0,01	ADAM	24	30	0,5044
0,01	ADAM	24	40	0,4956
0,01	ADAM	32	10	0,4956
0,01	ADAM	32	20	0,5044
0,01	ADAM	32	30	0,4956
0,01	ADAM	32	40	0,5044
0,01	SGD	16	10	0,9305
0,01	SGD	16	20	0,9256
0,01	SGD	16	30	0,9481
0,01	SGD	16	40	0,9612
0,01	SGD	24	10	0,9205
0,01	SGD	24	20	0,9434
0,01	SGD	24	30	0,9517
0,01	SGD	24	40	0,9575
0,01	SGD	32	10	0,9046

0,01	SGD	32	20	0,9470
0,01	SGD	32	30	0,9481
0,01	SGD	32	40	0,9554
0,01	ADAMAX	16	10	0,9583
0,01	ADAMAX	16	20	0,9623
0,01	ADAMAX	16	30	0,9630
0,01	ADAMAX	16	40	0,9670
0,01	ADAMAX	24	10	0,604
0,01	ADAMAX	24	20	0,9583
0,01	ADAMAX	24	30	0,9613
0,01	ADAMAX	24	40	0,9644
0,01	ADAMAX	32	10	0,9670
0,01	ADAMAX	32	20	0,9688
0,01	ADAMAX	32	30	0,9681
0,01	ADAMAX	32	40	0,9677
0,001	ADAM	16	10	0,9619
0,001	ADAM	16	20	0,9688
0,001	ADAM	16	30	0,9684
0,001	ADAM	16	40	0,9666
0,001	ADAM	24	10	0,9619
0,001	ADAM	24	20	0,9688
0,001	ADAM	24	30	0,9684
0,001	ADAM	24	40	0,9666
0,001	ADAM	32	10	0,9652
0,001	ADAM	32	20	0,9641
0,001	ADAM	32	30	0,8685
0,001	ADAM	32	40	0,9623
0,001	SGD	16	10	0,5352
0,001	SGD	16	20	0,6658
0,001	SGD	16	30	0,7050
0,001	SGD	16	40	0,7293

0,001	SGD	24	10	0,5394
0,001	SGD	24	20	0,6633
0,001	SGD	24	30	0,7507
0,001	SGD	24	40	0,7950
0,001	SGD	32	10	0,5388
0,001	SGD	32	20	0,6604
0,001	SGD	32	30	0,7921
0,001	SGD	32	40	0,8687
0,001	ADAMAX	16	10	0,9637
0,001	ADAMAX	16	20	0,9626
0,001	ADAMAX	16	30	0,9684
0,001	ADAMAX	16	40	0,9695
0,001	ADAMAX	24	10	0,9623
0,001	ADAMAX	24	20	0,9670
0,001	ADAMAX	24	30	0,9688
0,001	ADAMAX	24	40	0,9692
0,001	ADAMAX	32	10	0,9644
0,001	ADAMAX	32	20	0,9653
0,001	**ADAMAX**	**32**	**30**	**0,9659**
0,001	ADAMAX	32	40	0,9684

B. Hasil Percobaan Grid Search BaselineNet

learning-rate	*optimizer*	*batch-size*	*epoch*	*Accuracy*
0,1	ADAM	16	10	0,5083
0,1	ADAM	16	20	0,4917
0,1	ADAM	16	30	0,4917
0,1	ADAM	16	40	0,5083
0,1	ADAM	24	10	0,5083
0,1	ADAM	24	20	0,5083
0,1	ADAM	24	30	0,5083
0,1	ADAM	24	40	0,5083

0,1	ADAM	32	10	0,4917
0,1	ADAM	32	20	0,4917
0,1	ADAM	32	30	0,5083
0,1	ADAM	32	40	0,5083
0,1	SGD	16	10	0,9604
0,1	SGD	16	20	0,9568
0,1	SGD	16	30	0,9543
0,1	SGD	16	40	0,5084
0,1	SGD	24	10	0,9586
0,1	SGD	24	20	0,9557
0,1	SGD	24	30	0,9575
0,1	SGD	24	40	0,9597
0,1	SGD	32	10	0,9561
0,1	SGD	32	20	0,9612
0,1	SGD	32	30	0,3083
0,1	SGD	32	40	0,9597
0,1	ADAMAX	16	10	0,4917
0,1	ADAMAX	16	20	0,4917
0,1	ADAMAX	16	30	0,6629
0,1	ADAMAX	16	40	0,5083
0,1	ADAMAX	24	10	0,4917
0,1	ADAMAX	24	20	0,4917
0,1	ADAMAX	24	30	0,5083
0,1	ADAMAX	24	40	0,6332
0,1	ADAMAX	32	10	0,4917
0,1	ADAMAX	32	20	0,4917
0,1	ADAMAX	32	30	0,4917
0,1	ADAMAX	32	40	0,6480
0,01	ADAM	16	10	0,4917
0,01	ADAM	16	20	0,5083
0,01	ADAM	16	30	0,4917

0,01	ADAM	16	40	0,4917
0,01	ADAM	24	10	0,5083
0,01	ADAM	24	20	0,5083
0,01	ADAM	24	30	0,5083
0,01	ADAM	24	40	0,4917
0,01	ADAM	32	10	0,5083
0,01	ADAM	32	20	0,4917
0,01	ADAM	32	30	0,4917
0,01	ADAM	32	40	0,4917
0,01	SGD	16	10	0,9532
0,01	SGD	16	20	0,9575
0,01	SGD	16	30	0,9583
0,01	SGD	16	40	0,9604
0,01	SGD	24	10	0,9492
0,01	SGD	24	20	0,9568
0,01	SGD	24	30	0,9568
0,01	SGD	24	40	0,9583
0,01	SGD	32	10	0,9481
0,01	SGD	32	20	0,9597
0,01	SGD	32	30	0,9608
0,01	SGD	32	40	0,9590
0,01	ADAMAX	16	10	0,4917
0,01	ADAMAX	16	20	0,9481
0,01	ADAMAX	16	30	0,9528
0,01	ADAMAX	16	40	0,9601
0,01	ADAMAX	24	10	0,9594
0,01	ADAMAX	24	20	0,9503
0,01	ADAMAX	24	30	0,9586
0,01	ADAMAX	24	40	0,9550
0,01	ADAMAX	32	10	0,9608
0,01	ADAMAX	32	20	0,4917

0,01	ADAMAX	32	30	0,9499
0,01	ADAMAX	32	40	0,9496
0,001	ADAM	16	10	0,9543
0,001	ADAM	16	20	0,9554
0,001	ADAM	16	30	0,9565
0,001	ADAM	16	40	0,9568
0,001	ADAM	24	10	0,9561
0,001	ADAM	24	20	0,9536
0,001	ADAM	24	30	0,9554
0,001	ADAM	24	40	0,9525
0,001	ADAM	32	10	0,9561
0,001	ADAM	32	20	0,9586
0,001	ADAM	32	30	0,9565
0,001	ADAM	32	40	0,9550
0,001	SGD	16	10	0,6266
0,001	SGD	16	20	0,6419
0,001	SGD	16	30	0,6821
0,001	SGD	16	40	0,7997
0,001	SGD	24	10	0,5130
0,001	SGD	24	20	0,6803
0,001	SGD	24	30	0,7039
0,001	SGD	24	40	0,7671
0,001	SGD	32	10	0,6190
0,001	SGD	32	20	0,6110
0,001	SGD	32	30	0,7449
0,001	SGD	32	40	0,7518
0,001	ADAMAX	16	10	0,9554
0,001	ADAMAX	16	20	0,9561
0,001	ADAMAX	16	30	0,9528
0,001	ADAMAX	16	40	0,9561
0,001	ADAMAX	24	10	0,9543

0,001	ADAMAX	24	20	0,9554
0,001	ADAMAX	24	30	0,9510
0,001	ADAMAX	24	40	0,9532
0,001	ADAMAX	32	10	0,9554
0,001	ADAMAX	32	20	0,9543
0,001	ADAMAX	32	30	0,9517
0,001	ADAMAX	32	40	0,9536

C. Confusion Matrix

Percobaan		TP	FP	TN	FN
BaselineNet Tanpa Hyperparameter Tuning	1	1316	82	1301	57
	2	1217	56	1340	83
	3	1286	50	1346	74
	4	1285	42	1354	75
	5	1279	53	1343	81
BaselineNet Best Hyperparameter	1	1299	66	1330	61
	2	1281	38	1358	79
	3	1308	78	1317	52
	4	1280	30	1366	80
	5	1271	30	1366	89
Rajaraman Tanpa Hyperparameter Tuning	1	1271	22	1374	89
	2	1275	25	1371	85
	3	1262	23	1373	98
	4	1253	21	1375	107
	5	1278	31	1365	82
Rajaraman Best Hyperparameter	1	1287	23	1373	73
	2	1305	34	1362	55
	3	1308	38	1358	52
	4	1295	23	1373	65
	5	1293	24	1372	67

D. Percobaan Tambahan

Percobaan		TP	FP	TN	FN
Learning Rate	0,1	1360	1396	0	0
	0,01	0	0	1396	1360
	0,001	1298	67	1329	62
	0,0001	1275	44	1352	85
	0,00001	1143	59	1337	217
	0,000001	843	430	966	517
Batch Size	16	1321	54	1320	61
	32	1326	55	1319	56
	64	1326	55	1319	56
	128	1328	75	1299	54
	256	1319	64	1310	63
	512	1333	76	1298	49
Epochs	10	1301	54	1342	59
	20	1276	46	1350	84
	40	1286	48	1348	74
	80	1281	49	1347	79
	160	1281	47	1349	79
	320	1274	52	1344	86
Optimizers	ADAM	1333	51	1313	49
	SGD	1001	447	927	381
	NADAM	1321	54	1320	61
	ADAMAX	1301	43	1331	81
	RMSProp	1325	69	1305	57
	ADAGRAD	910	298	1076	472

E. Biodata Diri

Nama : Gregorius Guntur Sunardi Putra

NIM : 311810015

Program Studi : Teknik Informatika

Tempat, tanggal lahir : Blitar, 30 Maret 2001

Alamat : Gang Mahoni II, Karangbesuki, Sukun, Malang

No Hp : 085856065782

Email : guntur,pagi@gmail,com

Dosen Pembimbing PKL : Windra Swastika, Ph. D.

F. Lembar Bimbingan PKL

UNIVERSITAS MA CHUNG
FAKULTAS SAINS & TEKNOLOGI UNIVERSITAS MA CHUNG
Villa Puncak Tidar N-01 Malang 65151
fakultas.sains.teknologi@machung.ac.id
+62 341 550171 www.machung.ac.id

FORM PKL-FST09

LEMBAR BIMBINGAN PRAKTIK KERJA LAPANGAN DARING (PKL)

Nama Mahasiswa	:	Gregorius Guntur
NIM	:	311810015
Program Studi	:	Teknik Informatika
Judul Laporan PKL	:	Pengaruh Hyperparameter Tuning pada Deteksi Malaria Menggunakan CNN

No	Hari, tanggal	Topik Bimbingan	TTD Dosen Pembimbing
1	Senin, 13 September 2021	Pengarahan penelitian yang akan dijadikan PKL	
2	Senin, 20 September 2021	Hasil percobaan menggunakan hyperparameter tuning menggunakan Random Search MNIST	
3	Senin, 27 September 2021	Masalah Penggunaan KerasClassifier pada Hyperparameter Tuning	
4	Senin, 4 Oktober 2021	Hasil percobaan ulang penelitian sebelumnya preprocessing tuning	
5	Senin, 18 Oktober 2021	Progress Report Random Search pada Rajaraman dan BaselineNet	
6	Senin, 25 Oktober 2021	Perubahan metode Random Search ke Grid Search	
7	Senin, 15 November 2021	Kegagalan Hyperparameter Tuning Grid Search Arsitektur MicroVGG, dan Batasan TPU pada Google Collab	
8	Senin, 22 November 2021	Progres Report Grid Search tanpa Cross Validation Arsitektur BaselineNet dan Rajaraman	
9	Senin, 6 Desember 2021	Revisi Laporan PKL BAB 1, 2, 3 , dan 4	
10	Senin, 17 Desember 2021	Revisi Laporan PKL BAB 4 dan 5