

SKRIPSI

**APLIKASI PENCARIAN INFORMASI SEBARAN DAN PENGENALAN
COVID-19 MENGGUNAKAN CONVOLUTIONAL NEURAL
NETWORK BERBASIS ANDROID**

Untuk memenuhi syarat memperoleh gelar sarjana (S1)

Program Studi Teknik Informatika



Diajukan Oleh

Moh Rifkan 162025

Syamsul Bahri 162045

UNIVERSITAS DIPA MAKASSAR

2021

**APLIKASI PENCARIAN INFORMASI SEBARAN DAN PENGENALAN
COVID-19 MENGGUNAKAN CONVOLUTIONAL NEURAL
NETWORK BERBASIS ANDROID**

Telah disetujui untuk dipertahankan :

Pembimbing I



Indo Intan, S.T., M.T.
NIDN : 0929127802

Pembimbing II



Suryani, S.Kom., M.T.
NIDN : 0904018701

**APLIKASI PENCARIAN INFORMASI SEBARAN DAN PENGENALAN
COVID-19 MENGGUNAKAN CONVOLUTIONAL NEURAL
NETWORK BERBASIS ANDROID**

Diperiksa dan disahkan oleh Panitia Ujian Sarjana Strata Satu (S1)
Universitas DIPA
Bertempat di Makassar pada tanggal 22 April 2021

Ketua Merangkap Anggota



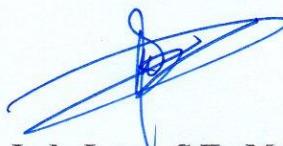
Ir. Irsal, M.T.
NIDN : 0911075701

Anggota:



Dr. Cucut Susanto, S.Kom., M.Si.
NIDN : 0927117301

Anggota:



Indo Intan, S.T., M.T.
NIDN : 0929127802

Anggota:



Suryani, S.Kom., M.T.
NIDN : 0904018701



Plagiarism Checker X - Report

Originality Assessment

Overall Similarity: **20%**

Date: Apr 20, 2021

Statistics: 2929 words Plagiarized / 14994 Total words

Remarks: Moderate similarity detected, you better improve the document (if required).

Judul : APLIKASI PENCARIAN INFORMASI SEBARAN DAN
PENGENALAN COVID-19 MENGGUNAKAN
CONVOLUTIONAL NEURAL NETWORK BERBASIS
ANDROID SKRIPSI

Pembimbing I : Indo Intan, ST., MT.

Pembimbing II : Suryani, S.Kom., MT

Nama Mahasiswa : - Moh Rifkan 162025

- Syamsul Bahri 162045

Di Periksa Oleh,

Nurniwa, S.Kom.

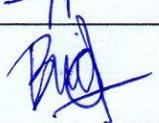
Makassar, 21 April 2021

DiKetahui Oleh,

Ir. Irsal, MT.

PERNYATAAN

Kami yang bertanda tangan dibawah ini menyatakan bahwa, Skripsi ini merupakan karya kelompok kami sendiri (ASLI), dan isi dalam Skripsi ini tidak terdapat karya yang pernah diajukan oleh orang lain atau kelompok lain untuk memproleh gelar akademis disuatu Institusi Pendidikan, dan sepanjang pengetahuan kami juga tidak terdapat karya atau pendapat yang pernah ditulis dan/atau diterbitkan oleh orang lain atau kelompok lain, kecuali yang secara tertulis diacu dalam naskah ini dan disebutkan dalam daftar pustaka.

Nama	Nomor Pokok	Tanda Tangan
Moh Rifkan	162025	
Syamsul Bahri	162045	

Demikian pernyataan yang kami buat, dan apabila dikemudian hari ternyata pernyataan kami ini ditemukan tidak benar, maka kami siap menerima sanksi akademik sesuai dengan peraturan yang ada.

Abstrak

Saat ini Covid-19 sudah menjadi pandemi di seluruh dunia dan diperlukan sebuah metode yang praktis dan efisien untuk membantu menghambat penyebaran virus tersebut dan sebuah platform yang dapat menampilkan informasi Covid-19 secara *realtime*. *Deep Learning* memiliki berbagai macam metode yang dapat digunakan untuk melakukan klasifikasi Covid-19. Metode *Convolutional Neural Network* (CNN) menjadi pilihan dalam penelitian yang dilakukan. Arsitektur CNN dirancang untuk melakukan pelatihan dan pengujian pada infrastruktur *Google Colaboratory*. Dari data hasil pelatihan tersebut kemudian disimpan dan dikonversi ke dalam bentuk *file TensorFlow Lite* dan diimpor ke dalam *project* pada *Android Studio* agar dapat diimplementasikan pada aplikasi *Android*. Berdasarkan hasil pelatihan dan pengujian pada *Google Colaboratory* dengan *dataset* citra *chest ray* yang berukuran 224x224 piksel didapat bahwa pada arsitektur CNN yang dirancang diperoleh tingkat akurasi pelatihan sebesar 0.9671 dan nilai akurasi validasi mencapai 0.9601. Selanjutnya dilakukan pengujian dari arsitektur tersebut pada perangkat *Android*. Hasil pengujian arsitektur CNN pada perangkat *Android* menghasilkan tingkat akurasi sebesar 86% menggunakan kamera dan 100% menggunakan gambar dari galeri.

Kata kunci : Klasifikasi Covid-19, Sebaran Informasi, *realtime*, *Android*, *Google Colaboratory*, CNN, *Citra Chest Ray*.

Abstract

Currently Covid-19 has become a worldwide pandemic and a practical and efficient method is needed to help prevent the spread of the virus and a platform that can display Covid-19 information in real time. Deep Learning has a variety of methods that can be used to classify Covid-19. The Convolutional Neural Network (CNN) method became an option in the research conducted. CNN's architecture is designed to conduct training and testing on the Google Colaboratory infrastructure. The training data is then saved and converted into TensorFlow Lite files and imported into projects in Android Studio so that they can be implemented on Android apps. Based on the results of training and testing at Google Colaboratory with chest ray image dataset measuring 224x224 pixels, it was obtained that in CNN architecture designed obtained a training accuracy rate of 0.9671 and validation accuracy value reached 0.9601. Further testing of the architecture is carried out on Android devices. CNN's architectural test results on Android devices resulted in an 86% accuracy rate using the camera and 100% using images from the gallery.

Keywords : Classification of Covid-19, Information Distribution, *realtime*, *Android*, *Google Colaboratory*, CNN, *Chest Ray Imagery*.

KATA PENGANTAR

Puji syukur peneliti panjatkan ke hadirat Tuhan Yang Maha Esa atas segala nikmat, karunia dan limpahan rahmat-Nya yang telah memberikan kekuatan kepada peneliti sehingga skripsi ini dapat diselesaikan, yang merupakan salah satu persyaratan dalam memenuhi kegiatan beban kerja dosen penyelesaian pendidikan Strata Satu di Universitas DIPA Makassar.

Dalam skripsi ini berbagai hambatan dan keterbatasan dihadapi oleh peneliti mulai dari tahap persiapan sampai dengan penyelesaian tulisan, namun berkat bantuan bimbingan dan kerja sama berbagai pihak, hambatan dan kesulitan tersebut dapat teratasi.

Oleh karena itu perkenankanlah peneliti dengan segala kerendahan hati menyampaikan ucapan terima kasih yang tulus dan penghargaan yang tak terhingga kepada :

1. Dr. Jhony W. Soetikno, S.E,M.M. selaku Rektor Universitas DIPA Makassar.
2. Ir. Irsal, M.T. selaku Ketua Jurusan Teknik Informatika program studi strata satu (S1) Universitas DIPA.
3. Indo Intan, S.T., M.T. selaku Pembimbing I, yang telah banyak berkontribusi, meluangkan waktunya, tenaga, memberikan arahan, motivasi dengan segala ketelitian, dan membimbing dengan sabar, sehingga skripsi ini dapat terselesaikan sampai akhir penulisan.

4. Suryani, S.Kom., M.T. selaku Pembimbing II, yang telah banyak memberikan bimbingan, arahan, dan motivasi kepada peneliti dalam menyelesaikan skripsi ini.
5. Dosen Universitas DIPA Makassar yang telah mendidik dan mengajarkan berbagai disiplin ilmu kepada penulis.
6. Kedua orang tua tercinta yang tak bosan-bosannya memberikan nasehat dan dukungan yang tidak dapat kami nilai dalam bentuk apa pun. Semoga Tuhan senantiasa melimpahkan kesehatan dan kesejahteraan bagi beliau, Amiin.
7. Untuk semua teman-teman tanpa terkecuali yang tidak dapat disebutkan namanya, terima kasih untuk setiap bantuan yang telah kalian berikan kepada peneliti.

Peneliti menyadari bahwa skripsi ini masih banyak terdapat kekurangan. Oleh karena itu peneliti sangat mengharapkan kritik dan saran yang membangun, dan semoga skripsi ini dapat bermanfaat bagi yang membacanya. Akhirnya teriring doa dan harapan semoga segala bantuan yang telah diberikan baik materi maupun moril terdapat imbalan disisi Tuhan YME dan bermanfaat bagi kita semua. Amiin

Makassar, Maret 2021

P e n u l i s

DAFTAR ISI

	Halaman
HALAMAN JUDUL	i
HALAMAN PERSETUJUAN	ii
HALAMAN PENGESAHAN	iii
HALAMAN PLAGIARISM CHECK.....	iv
HALAMAN PERNYATAAN PENULIS	v
ABSTRAK.....	vi
KATA PENGANTAR.....	vii
DAFTAR ISI	ix
DAFTAR TABEL	xiii
DAFTAR GAMBAR.....	xiv
BAB I PENDAHULUAN	1
1.1. Latar Belakang	1
1.2. Rumusan Masalah	2
1.3. Batasan Masalah.....	3
1.4. Tujuan Dan Manfaat Penelitian.....	3
1.4.1. Tujuan Penelitian	3
1.4.2. Manfaat Penelitian	4
1.5. Sistematika Penulisan	4
BAB II TINJAUAN PUSTAKA	6
2.1. Kerangka Pikir.....	6
2.2. Landasan Teori	7

2.2.1.	Klasifikasi.....	7
2.2.2.	Algoritma.....	7
2.2.3.	<i>Deep Learning</i>	8
2.2.4.	<i>Convolutional Neural Network (CNN)</i>	10
2.2.5.	<i>Google Colaboratory</i>	17
2.2.6.	<i>Keras Dan TensorFlow</i>	18
2.2.7.	<i>Tensorflow Lite</i>	21
2.2.8.	Android Studio.....	23
2.2.9.	Aplikasi.....	25
2.2.10.	<i>Kotlin</i>	25
2.2.11.	<i>Python</i>	26
2.2.12.	API	27
2.2.13.	<i>Unified Modeling Language</i>	28
2.3.	Konsep Dasar Pengujian Perangkat Lunak	33
2.4.	Pengujian UI (<i>User Interface</i>).....	34
2.4.1.	Karakteristik Pengujian UI (<i>User Interface</i>)	34
2.5.	Penelitian Terkait.....	35
BAB III	METODOLOGI PENELITIAN.....	39
3.1.	Waktu dan Lokasi Penelitian	39
3.2.	Jenis Penelitian	39
3.3.	Teknik Pengumpulan Data.....	40
3.4.	Arsitektur Sistem.....	41
3.5.	Alat dan Bahan Penelitian.....	42

3.5.1. Alat Penelitian.....	43
3.5.2. Bahan Penelitian	43
3.6. Metode Pengujian	43
3.7. Tahap Penelitian dan Jadwal Penelitian.....	44
3.7.1. Tahap Penelitian.....	44
3.7.2. Rancangan Jadwal Penelitian.....	46
BAB IV ANALISIS DAN IMPLEMENTASI	47
4.1. Rancangan Sistem	47
4.1.1. <i>Use Case Diagram</i>	47
4.1.2. <i>Activity Diagram</i>	48
4.1.3. <i>Sequence Diagram</i>	50
4.1.4. <i>Class Diagram</i>	54
4.2. Perancangan Interface.....	55
4.3. <i>Dataset</i> Penelitian.....	58
4.3.1. <i>Dataset</i>	58
4.3.2. Penyimpanan <i>Dataset</i>	60
4.4. Perancangan Program pada <i>Google Colaboratory</i>	61
4.5. Perancangan Program pada Android Studio	67
BAB V HASIL DAN PENGUJIAN SISTEM	70
5.1. Pengujian Sistem	70
5.1.1. Hasil Pelatihan dan Pengujian pada Google Colaboratory	70
5.1.2. Hasil Pelatihan dan Pengujian pada Perangkat Android	80

5.2. Rekapitulasi Hasil Pengujian	88
BAB VI 91KESIMPULAN DAN SARAN.....	91
6.1. Kesimpulan	91
6.2. Saran	92
DAFTAR PUSTAKA.....	93
BIODATA PENULIS	

DAFTAR TABEL

	Halaman
Tabel 2.1 Simbol <i>Use Case Diagram</i>	29
Tabel 2.2 Simbol <i>Class Diagram</i>	30
Tabel 2.3 Simbol <i>Activity Diagram</i>	32
Tabel 2.4 Simbol <i>Sequence Diagram</i>	33
Tabel 3.1 Perangkat Keras (<i>hardware</i>)	43
Tabel 3.2 Perangkat Lunak (<i>Software</i>)	43
Tabel 3.3 Jadwal Penelitian.....	46
Tabel 5.1 Hasil pengujian arsitektur CNN dengan sumber gambar kamera	81
Tabel 5.2 Hasil pengujian arsitektur CNN dengan sumber gambar galeri	82
Tabel 5.3 Rekapitulasi Hasil Pengujian <i>Interface</i> Menggunakan <i>Espresso</i>	88

DAFTAR GAMBAR

	Halaman
Gambar 2.1 Kerangka Pikir.....	6
Gambar 2.2 Jaringan saraf mendalam untuk klasifikasi digit gambar.....	9
Gambar 2.3 Representasi mendalam yang dipelajari oleh model klasifikasi digit	10
Gambar 2.4 Arsitektur Jaringan CNN	11
Gambar 2.5 Proses Perhitungan Metode CNN.....	12
Gambar 2.6 Konvolusi	13
Gambar 2.7 Perhitungan Konvolusi.....	13
Gambar 2.8 <i>Pooling</i>	14
Gambar 2.9 <i>Fully Connected</i>	15
Gambar 2.10 <i>Rectified Linear Unit (ReLU)</i>	16
Gambar 2.11 Antarmuka <i>Google Colaboratory</i>	18
Gambar 2.12 Minat <i>Framework Deep Learning</i> Berdasarkan Pencarian Google ..	19
Gambar 2.13 Susunan perangkat lunak dan perangkat keras <i>deep learning</i>	21
Gambar 2.14 Arsitektur <i>TensorFlow Lite</i>	22
Gambar 3.1 Arsitektur Sistem	41
Gambar 3.2 <i>Flowchart</i> Pembuatan Model	42
Gambar 4.1 <i>Use Case Diagram</i> Covid-19 Detector	47
Gambar 4.2 <i>Activity Diagram</i> Covid-19 Detector	49
Gambar 4.3 <i>Sequence Diagram</i> Menu Home	50
Gambar 4.4 <i>Sequence Diagram</i> Menu Informasi	51

Gambar 4.5 <i>Sequence Diagram</i> Menu Klasifikasi	52
Gambar 4.6 <i>Sequence Diagram</i> Menu Rujukan	53
Gambar 4.7 <i>Class Diagram</i> Aplikasi Covid-19 Detector	54
Gambar 4.8 Tampilan Halaman <i>Home</i> dan Detail Informasi Covid-19	55
Gambar 4.9 Tampilan Halaman Informasi.....	56
Gambar 4.10 Tampilan Halaman Informasi.....	57
Gambar 4.11 Tampilan Halaman Rujukan.....	58
Gambar 4.12 Tampilan <i>dataset covid-19</i> pada <i>google drive</i>	60
Gambar 4.13 Tampilan program klasifikasi Covid-19 pada <i>Google Colaboratory</i>	61
Gambar 4.14 Rancangan Arsitektur CNN	63
Gambar 4.15 Input Gambar.....	65
Gambar 4.16 <i>Feture Maps</i> Input Gambar	65
Gambar 4.17 Proses <i>Feature Maps</i> pada Lapisan Konvolusi ke-1	66
Gambar 4.18 Proses <i>Feature Maps</i> pada Lapisan Konvolusi ke-2	66
Gambar 4.19 Proses <i>Feature Maps</i> pada Lapisan Konvolusi ke-3	66
Gambar 4.20 Proses <i>Feature Maps</i> pada Lapisan Konvolusi ke-4	67
Gambar 4.21 Proses <i>Feature Maps</i> pada Lapisan Konvolusi ke-5	67
Gambar 4.22 Proses <i>Feature Maps</i> pada Lapisan Konvolusi ke-6	67
Gambar 4.23 Tampilan projek aplikasi Covid-19 Detector pada Android Studio	68
Gambar 4.24 File model (.tflite) beserta labelnya (.txt) pada folder assets	69
Gambar 5.1 Kurva tingkat akurasi hasil pelatihan dan validasi pada arsitektur CNN yang telah dirancang	71

Gambar 5.2 Kurva tingkat kesalahan pelatihan dan validasi pada arsitektur CNN yang telah dirancang	72
Gambar 5.3 Pengujian sample ke-1 pada arsitektur CNN di google colaboratory	73
Gambar 5.4 Pengujian sample ke-2 pada arsitektur CNN di google colaboratory	74
Gambar 5.5 Pengujian sample ke-3 pada arsitektur CNN di google colaboratory	75
Gambar 5.6 Pengujian sample ke-4 pada arsitektur CNN di google colaboratory	76
Gambar 5.7 Pengujian sample ke-5 pada arsitektur CNN di google colaboratory	77
Gambar 5.8 Confusion matrix hasil pengujian pada arsitektur CNN yang telah dirancang	78
Gambar 5.9 <i>Instrumental Testing</i> dengan <i>Espresso</i>	87
Gambar 5.10 Hasil pengujian <i>Instrumental Testing</i> menggunakan <i>Espresso</i>	87

BAB I

PENDAHULUAN

1.1. Latar Belakang

Coronavirus (COVID-19) adalah virus yang diidentifikasi pertama kali di Wuhan, China pada bulan Desember 2019 (Li *dkk.*, 2020). Infeksi virus ini kemudian menyebar ke seluruh China dan negara-negara lain di seluruh dunia yang ditetapkan sebagai pandemi oleh Organisasi Kesehatan Dunia (WHO). Di Indonesia, kasus pertama Covid-19 diidentifikasi pada tanggal 2 Maret 2020. Hingga 17 Maret 2020, ada 134 kasus yang telah terkonfirmasi yang tersebar di delapan provinsi, yaitu Bali, Banten, DKI Jakarta, Jawa Barat, Jawa Tengah, Kalimantan Barat, Sulawesi Utara dan Yogyakarta (KOMPAS, 2020).

Protokol untuk melakukan pengujian apakah subjek terinfeksi Covid-19 harus berdasarkan pada faktor klinis, *epidemiologis* serta penilaian dari kemungkinan infeksi. Salah satu pemeriksaan yang dilakukan di laboratorium sebagai faktor klinis adalah pemeriksaan *radiologis* atau pencitraan. Pemeriksaan ini dilakukan sebelum pemeriksaan PCR dilakukan. Kedua hasil pemeriksaan ini dilakukan secara urut karena keduanya saling menguatkan satu dengan lainnya. Hasil pemeriksaan pencitraan memiliki akurasi sekitar 90% sama baiknya dengan PCR (Shrestha & Shrestha, 2020). Dari sekian banyak kasus yang masuk pada laboratorium, maka tentu saja untuk menentukan diagnosis berdasarkan pencitraan tidak mudah dan tidak selalu lancar, karena ternyata di antara banyak kasus sering terjadi hasil pemeriksaan yang berbeda-beda dari berbagai pemeriksaan. Hal ini

dikarenakan jika pemeriksaan pencitraan dilakukan, maka interpretasi dokter satu dengan lainnya biasa berbeda dan harus mencapai kuorum kesimpulan yang diperoleh. Sementara pada kondisi tertentu, tentu saja hanya satu dokter yang bertugas dan dilibatkan pada bagian pencitraan untuk mendiagnosis kasus sesuai dengan kompetensinya. Kompetensi dan sudut pandang melihat pencitraan yang kadang berbeda inilah memungkinkan hasil yang tidak objektif terhadap kasus pasien.

Untuk mengatasi masalah tersebut maka penelitian ini digagas sebagai langkah untuk memberikan informasi kepada masyarakat sebaran covid-19 berbasis *mobile-android* sekaligus informasi bagi dokter atau tenaga medis dalam mengklasifikasi sebagai penegakan lanjutan terhadap hasil diagnosis.

Metode klasifikasi yang digunakan yaitu *convolution neural network* (CNN). Pemilihan metode ini karena CNN sebagai bagian dari *deep learning* yang menjadi *trend* klasifikasi mutakhir yang diimplementasikan dalam berbagai bidang pengenalan pola karena memiliki akurasi dalam kisaran 95-99% (Islam *dkk.*, 2017) dan (Stephen *dkk.*, 2019).

Implementasi CNN menggunakan bahasa *python* dengan *library tensorflow* yang menghasilkan *output* berupa model, kemudian model tersebut di masukkan ke dalam aplikasi *mobile* sebagai pola untuk memproses data *input* sehingga menghasilkan hasil klasifikasi.

1.2. Rumusan Masalah

Berdasarkan latar belakang masalah, maka perumusan masalah yang akan dibuat dalam penelitian ini adalah :

1. Bagaimana merancang aplikasi untuk mengetahui jumlah sebaran Covid-19 di Indonesia secara *up to date*?
2. Bagaimana merancang sistem pengenalan Covid-19 berdasarkan citra *Chest Ray* menggunakan *Convolutional Neural Network* (CNN)?
3. Bagaimana menguji performansi sistem pengenalan Covid-19 menggunakan *Convolutional Neural Network* (CNN)?
4. Bagaimana menguji perangkat lunak aplikasi menggunakan *library Espresso*?

1.3. Batasan Masalah

Sesuai dari tujuan semula, maka penulis membatasi masalah pada sistem ini pada :

1. Peneliti hanya membahas sebaran Covid-19 di Indonesia berdasarkan provinsi.
2. Peneliti hanya membahas pengenalan penyakit Covid-19 dengan tiga *output* yaitu, normal, Covid-19, dan viral pneumonia.
3. Peneliti merancang aplikasi berbasis android menggunakan bahasa pemrograman *kotlin*.

1.4. Tujuan Dan Manfaat Penelitian

1.4.1. Tujuan Penelitian

Adapun tujuan penelitian ini adalah :

1. Membangun sebuah aplikasi *mobile android* yang dapat menampilkan informasi sebaran Covid-19 di Indonesia berdasarkan provinsi dengan memanfaatkan REST API.
2. Membuat aplikasi pengenalan Covid-19 menggunakan *convolutional neural network* (CNN) dengan menggunakan *library Tensorflow*.

3. Melakukan pengujian performansi sistem pengenalan Covid-19 menggunakan *Convolutional Neural Network* (CNN) dengan cara menghitung kecepatan proses deteksi citra *Chest Ray*.
4. Melakukan pengujian perangkat lunak aplikasi dengan membuat skenario testing dan menguji aplikasi menggunakan *library Espresso*.

1.4.2. Manfaat Penelitian

Manfaat yang diharapkan dalam penelitian ini :

1. Untuk Akademi

Sebagai bahan acuan bagi mahasiswa untuk membuat sebuah program *mobile android* menggunakan *convolutional neural network* (CNN) dalam pengenalan citra *Chest Ray*.

2. Untuk Dokter/Tim Medis

Sebagai media pencarian informasi sebaran Covid-19 di Indonesia berdasarkan provinsi dan sebagai alat bantu secara *mobile* dalam pengenalan Covid-19 tanpa adanya kendala waktu dan tempat.

3. Untuk Peneliti

Sebagai sarana pembelajaran bagi peneliti untuk pembuatan aplikasi *mobile android* dalam pencarian sebaran dan pengenalan Covid-19 menggunakan *convolutional neural network* (CNN).

1.5. Sistematika Penulisan

Sistematika penulisan mengenai pembahasan masalah dan penyelesaiannya akan diuraikan sebagai berikut :

BAB I PENDAHULUAN

Bab ini membahas tentang latar belakang, rumusan masalah, batasan masalah, tujuan, dan manfaat penelitian, serta sistematika penulisan.

BAB II TINJAUAN PUSTAKA

Bab ini membahas kerangka pikir, landasan teori, klasifikasi, algoritma, *deep learning*, *convolutional neural network* (CNN), *google colaboratory*, *keras* dan *tensorflow*, *tensorflow lite*, android studio, aplikasi, *kotlin*, *python*, API, *unified modeling language* (UML), konsep dasar pengujian perangkat lunak, karakteristik pengujian *user interface* (UI), dan penelitian terkait.

BAB III METODE PENELITIAN

Bab ini membahas tentang metode penelitian, waktu dan lokasi penelitian, jenis penelitian, teknik pengumpulan data, arsitektur sistem, alat dan bahan penelitian, metode pengujian, tahap penelitian dan jadwal penelitian.

BAB IV PERANCANGAN SISTEM

Bab ini membahas tentang analisis sistem yang ada, rancangan system dan implementasi.

BAB V HASIL DAN PENGUJIAN SISTEM

Bab ini membahas tentang cara pengujian sistem yang dibuat untuk mengetahui terdapatnya kesalahan atau tidak pada sistem yang dibuat.

BAB VI KESIMPULAN DAN SARAN

Bab ini membahas tentang kesimpulan dan saran.

BAB II

TINJAUAN PUSTAKA

2.1. Kerangka Pikir

Untuk lebih memperjelas kerangka berpikir yang akan disajikan, maka akan digambarkan sebagai berikut :

Trend peningkatan angka pasien covid-19 terus melinjak dan tertinggi di Asia bahkan di dunia. Hal ini menjadi tantangan dalam dunia medis untuk melakukan pemeriksaan dengan menggunakan semua teknologi mutakhir. Salah satu pemeriksaan yang rutin dilakukan yaitu pencitraan atau radiografi paru.



Pemeriksaan yang dilakukan dokter atau tenaga medis di saat yang berbeda terkadang menimbulkan perbedaan informasi terhadap radiografi pasien sehingga membutuhkan perulangan pemeriksaan ataupun dianggap belum valid. Hal ini berimplikasi terhadap hasil diagnosis yang tidak objektif sehingga penegakannya terkadang tidak meyakinkan.



Metode klasifikasi yang digunakan yaitu *convolution neural network* (CNN). Pemilihan metode ini karena CNN sebagai bagian dari *deep learning* yang menjadi *trend* klasifikasi mutakhir yang diimplementasikan dalam berbagai bidang pengenalan pola karena metode ini memiliki tingkat akurasi yang tinggi yaitu kisaran 90–99%.



Diharapkan dengan adanya aplikasi ini dapat menjadi media informasi sebaran covid-19 kepada masyarakat sekaligus informasi bagi dokter atau tenaga medis dalam memberikan hasil pemeriksaan radiografi Covid-19 berbasis *mobile-android*.

Gambar 2.1 Kerangka Pikir

2.2. Landasan Teori

2.2.1. Klasifikasi

Dalam Kamus Besar Bahasa Indonesia (KBBI) klasifikasi adalah pemisahan/pemilahan/pembagian penggolongan menurut kaidah atau standar yang ditetapkan, penyusunan dan penerapan sesuatu ke dalam kelas-kelasnya. Klasifikasi berasal dari kata Latin “*classis*” atau proses pengelompokan, artinya mengumpulkan benda/entitas yang sama serta memisahkan benda/entitas yang tidak sama.

2.2.2. Algoritma

Menurut (Romzi, 2012):

Algoritma sangat lekat dengan kata logika, yaitu kemampuan seorang manusia untuk berpikir dengan akal tentang suatu permasalahan menghasilkan sebuah kebenaran, dibuktikan dan dapat diterima akal, logika sering kali dihubungkan dengan kecerdasan, seseorang yang mampu berlogika dengan baik sering orang menyebutnya sebagai pribadi yang cerdas. Dalam menyelesaikan suatu masalah pun logika mutlak diperlukan.

Algoritma adalah metode efektif yang diekspresikan sebagai rangkaian terbatas. Algoritma merupakan kumpulan perintah untuk menyelesaikan suatu masalah. Perintah-perintah ini dapat diterjemahkan secara bertahap dari awal hingga akhir. Masalah tersebut dapat berupa apa saja, dengan syarat untuk setiap permasalahan memiliki kriteria kondisi awal yang harus dipenuhi sebelum menjalankan sebuah algoritma. Algoritma juga memiliki perulangan proses (iterasi), dan juga memiliki keputusan hingga keputusan selesai.

2.2.3. Deep Learning

Menurut (Miceli dkk., 2018):

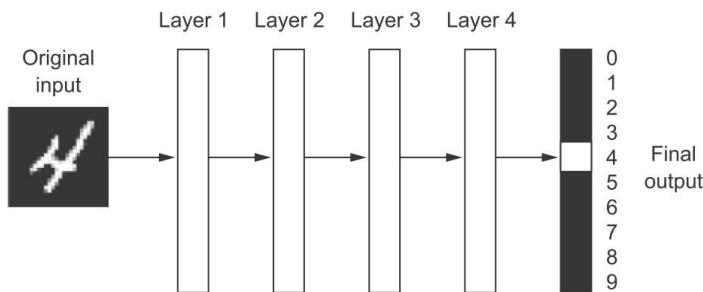
Deep learning is a specific subfield of machine learning: a new take on learning representations from data that puts an emphasis on learning successive layers of increasingly meaningful representations. The deep in deep learning isn't a reference to any kind of deeper understanding achieved by the approach; rather, it stands for this idea of successive layers of representations. How many layers contribute to a model of the data is called the depth of the model. Other appropriate names for the field could have been layered representations learning and hierarchical representations learning. Modern deep learning often involves tens or even hundreds of successive layers of representations and they're all learned automatically from exposure to training data. Meanwhile, other approaches to machine learning tend to focus on learning only one or two layers of representations of the data; hence, they're sometimes called shallow learning.

Deep Learning adalah sub bidang khusus dari pembelajaran mesin (*machine learning*) yaitu pandangan baru tentang representasi pembelajaran dari data yang menekankan pada pembelajaran lapisan berturut-turut dari representasi yang semakin informatif (Miceli dkk., 2018). Pendalaman dalam *Deep Learning* bukan merupakan referensi untuk segala jenis pemahaman yang lebih dalam yang dicapai oleh pendekatan, melainkan ia mewakili gagasan lapisan representasi yang berurutan. Banyak lapisan (layer) yang berkontribusi pada model data disebut kedalaman model. Nama-nama lain yang sesuai untuk bidang ini dapat berupa representasi pembelajaran berlapis dan pembelajaran representasi hierarkis. *Deep Learning* modern sering melibatkan puluhan atau bahkan ratusan lapisan representasi berturut-turut dan mereka semua belajar secara otomatis dari data pelatihan. Sementara itu, pendekatan lain untuk pembelajaran mesin cenderung

fokus pada pembelajaran hanya satu atau dua lapisan representasi data. Oleh karena itu mereka kadang-kadang disebut pembelajaran dangkal.

Dalam *deep learning*, representasi berlapis ini (hampir selalu) dipelajari melalui model yang disebut jaringan saraf, terstruktur dalam lapisan *literal* yang ditumpuk satu sama lain. Istilah jaringan saraf adalah referensi untuk *neurobiologi*, tetapi meskipun beberapa konsep dalam *deep learning* dikembangkan dalam bagian dengan menggambarkan inspirasi dari pemahaman kita tentang otak, model *deep learning* bukanlah model otak. Tidak ada bukti bahwa otak mengimplementasikan sesuatu seperti mekanisme pembelajaran yang digunakan dalam model *deep learning modern*.

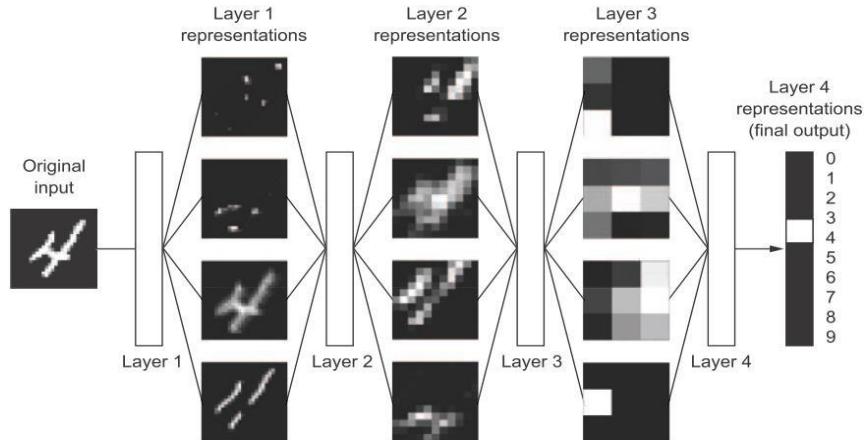
Representasi yang dipelajari pada algoritma *deep learning* dapat dilihat bagaimana jaringan lapisan banyak mengubah gambar digit untuk mengenali digit apa itu seperti pada gambar 2.2.



Gambar 2.2 Jaringan saraf mendalam untuk klasifikasi digit gambar
Sumber: (Miceli dkk., 2018)

Pada gambar 2.3, jaringan mengubah gambar digit menjadi representasi yang semakin berbeda dari gambar asli dan semakin informatif tentang hasil akhir. Kita dapat menganggap jaringan yang dalam sebagai operasi distilasi informasi

yang bertingkat, di mana informasi melewati filter berturut-turut dan keluar semakin dimurnikan (yaitu, berguna berkaitan dengan beberapa tugas).



Gambar 2.3 Representasi mendalam yang dipelajari oleh model klasifikasi digit
Sumber: (Miceli dkk., 2018)

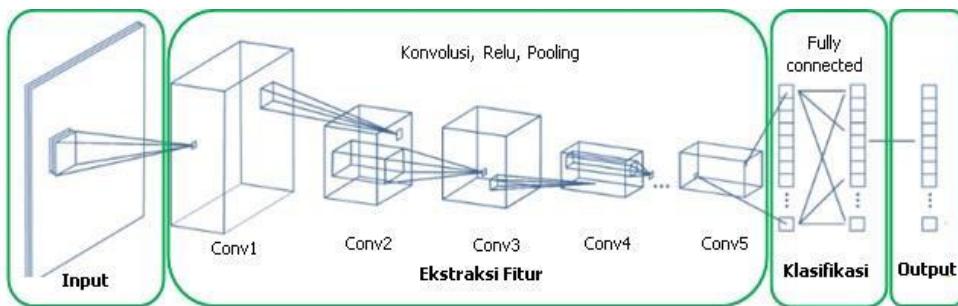
2.2.4. Convolutional Neural Network (CNN)

Menurut (Eka Putra, 2016):

CNN pertama kali dikembangkan dengan nama *NeoCognitron* oleh Kunihiko Fukushima, seorang peneliti dari NHK *Broadcasting Science Research Laboratories*, Kinuta, Setagaya, Tokyo, Jepang. Konsep tersebut kemudian dimatangkan oleh Yann LeChun, seorang peneliti dari AT&T *Bell Laboratories* di Holmdel, New Jersey, USA. Model CNN dengan nama *LeNet* berhasil diterapkan oleh LeChun pada penelitiannya mengenai pengenalan angka dan tulisan tangan. Pada tahun 2012, Alex Krizhevsky dengan penerapan CNN miliknya berhasil menjuarai kompetisi *ImageNet Large Scale Visual Recognition Challenge* 2012. Prestasi tersebut menjadi momen pembuktian bahwa metode *Deep Learning*, khususnya CNN. Metode CNN terbukti berhasil mengungguli metode *Machine Learning* lainnya seperti SVM pada kasus klasifikasi objek pada citra.

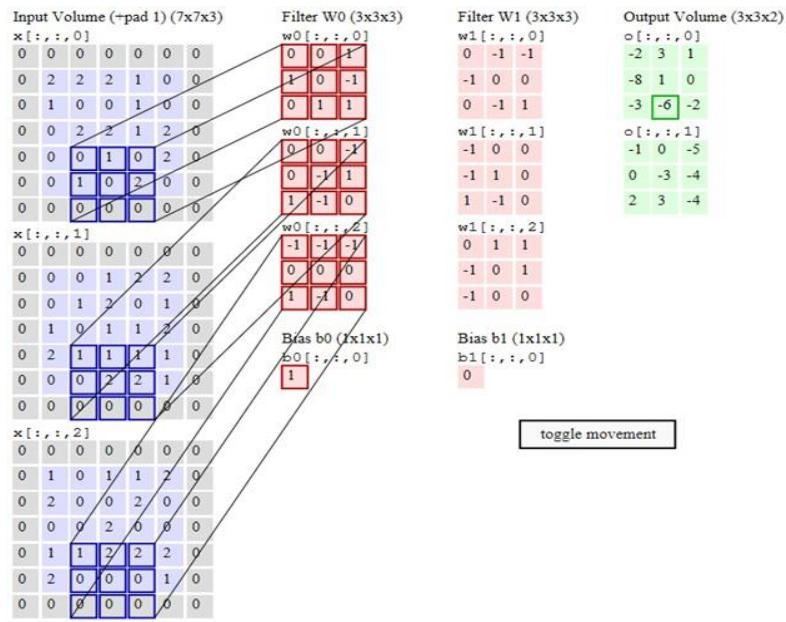
Convolutional Neural Network (CNN) adalah pengembangan dari *Multilayer Perceptron* (MLP) yang didesain untuk mengolah data dua dimensi. CNN termasuk dalam jenis *Deep Neural Network* karena kedalaman jaringan yang tinggi dan banyak diaplikasikan pada data citra. Menurut (Maha dkk., 2019) Sama halnya dengan metode *neural network* pada umumnya, metode CNN mempunyai

beberapa *hidden layer* dari suatu *input* yang memiliki vektor tunggal. Di dalam *input* yang merupakan citra digital yang dijadikan ke dalam vektor tunggal. Di dalam *features extraction layer*, terdapat beberapa *block* yang melakukan *convolution*, *Relu*, *pooling*, dan seterusnya begitu. Dan selanjutnya di *classification layer* mendapatkan *input* dari proses sebelumnya untuk menentukan fitur mana yang paling berkorelasi dengan kelas tertentu. Layer terakhir yang terhubung dengan *hidden layer* sebelumnya disebut dengan *output layer* dan disajikan hasil akhir ke dalam klasifikasi kelas.



Gambar 2.4 Arsitektur Jaringan CNN
Sumber: (Arrofiqoh & Harintaka, 2018)

Berikut adalah gambaran langkah-langkah yang digunakan dalam proses perhitungan dengan menggunakan metode CNN:



Gambar 2.5 Proses Perhitungan Metode CNN

Sumber: (Maha dkk., 2019)

2.2.4.1. Konvolusi

Menurut (Ilahiyah & Nilogiri, 2018):

Konvolusi didefinisikan sebagai proses untuk memperoleh suatu piksel didasarkan pada nilai piksel itu sendiri dan tetangganya dengan melibatkan suatu matriks yang disebut kernel yang merepresentasikan pembobotan. Operasi ini menerapkan fungsi *output* sebagai *Feature Map* dari masukan citra. Masukan dan keluaran ini dapat dilihat sebagai dua argumen bernilai riil.

Konvolusi (*convolution*) adalah sebuah proses di mana citra dimanipulasi dengan menggunakan eksternal *mask* atau sub Windows untuk menghasilkan citra yang baru. Secara matematis menurut (Rohim dkk., 2019) konvolusi adalah jumlah total dari hasil kali antara setiap elemen yang bersesuaian (memiliki posisi koordinat yang sama) dalam dua matriks atau dua vektor, seperti yang ditunjukkan

Gambar 2.6.

Tujuan dilakukannya konvolusi pada data citra adalah untuk mengekstraksi fitur dari citra *input*. Konvolusi akan menghasilkan transformasi linear dari data *input* sesuai informasi spasial pada data. Bobot pada layer tersebut menypesifikasi kernel konvolusi yang digunakan, sehingga kernel konvolusi dapat dilatih berdasarkan *input* pada CNN.

$$\begin{array}{|c|c|c|c|c|c|} \hline
 3 & 0 & 1 & 2 & 7 & 4 \\ \hline
 1 & 5 & 8 & 9 & 3 & 1 \\ \hline
 2 & 7 & 2 & 5 & 1 & 3 \\ \hline
 0 & 1 & 3 & 1 & 7 & 8 \\ \hline
 4 & 2 & 1 & 6 & 2 & 8 \\ \hline
 2 & 4 & 5 & 2 & 3 & 9 \\ \hline
 \end{array} * \begin{array}{|c|c|c|} \hline
 1 & 0 & -1 \\ \hline
 1 & 0 & -1 \\ \hline
 1 & 0 & -1 \\ \hline
 \end{array} = \begin{array}{|c|c|c|c|} \hline
 -5 & -4 & 0 & 8 \\ \hline
 -10 & -2 & -2 & 3 \\ \hline
 0 & -2 & -4 & -7 \\ \hline
 -3 & -2 & -3 & -16 \\ \hline
 \end{array}$$

6×6 3×3 $=$ 4×4

Gambar 2.6 Konvolusi
Sumber: (Rohim dkk., 2019)

$$\begin{array}{|c|c|c|} \hline
 2 & 1 & 3 \\ \hline
 4 & 6 & 5 \\ \hline
 8 & 9 & 7 \\ \hline
 \end{array} * \begin{array}{|c|c|c|} \hline
 10 & 20 & 10 \\ \hline
 20 & 10 & 20 \\ \hline
 10 & 20 & 10 \\ \hline
 \end{array} = C = 2 \times 10 + 1 \times 20 + 3 \times 10 + 4 \times 20 + 6 \times 10 + 5 \times 20 + 8 \times 10 + 9 \times 20 + 7 \times 10 = 640$$

Matriks A Matriks B

Gambar 2.7 Perhitungan Konvolusi
Sumber: (Rohim dkk., 2019)

2.2.4.2.Pooling

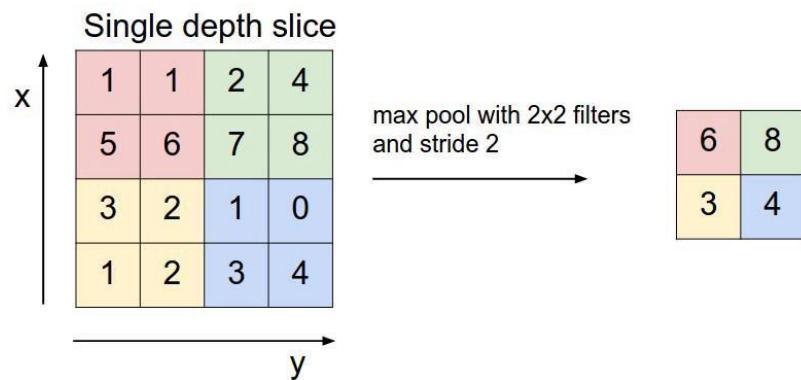
Menurut (Ilahiyah & Nilogiri, 2018):

Pooling Layer adalah lapisan yang menggunakan fungsi dengan *Feature Map* sebagai masukan dan mengolahnya dengan berbagai macam operasi statistik berdasarkan nilai piksel terdekat. Pada model CNN, lapisan *Pooling* biasanya disisipkan secara teratur setelah beberapa lapisan konvolusi. Lapisan *Pooling* yang dimasukkan di antara lapisan konvolusi secara berturut-turut dalam arsitektur model CNN dapat secara progresif mengurangi ukuran volume *output* pada *Feature Map*, sehingga mengurangi jumlah parameter dan perhitungan di jaringan, dan untuk mengendalikan *Overfitting*. Lapisan *Pooling* bekerja di setiap tumpukan *Feature Map* dan mengurangi ukurannya. Bentuk lapisan *Pooling* yang paling umum adalah

dengan menggunakan filter berukuran 2×2 yang diaplikasikan dengan langkah sebanyak 2 dan kemudian beroperasi pada setiap irisan dari *input*. Bentuk seperti ini akan mengurangi *Feature Map* hingga 75% dari ukuran aslinya.

Pooling adalah proses mereduksi ukuran sebuah data citra. Dalam pengolahan citra, *pooling* juga bertujuan untuk meningkatkan *invariansi* posisi dari fitur serta mempercepat komputasi dan mengontrol terjadinya *overfitting*.

Dalam sebagian besar CNN metode *pooling* yang sering digunakan adalah *max pooling*. *Max pooling* membagi *output* dari *convolution layer* menjadi beberapa *grid* kecil lalu mengambil nilai maksimal dari setiap *grid* untuk menyusun matriks citra yang telah direduksi seperti yang ditunjukkan pada Gambar 2.8. *Grid* yang berwarna merah, hijau, kuning dan biru merupakan kelompok *grid* yang akan dipilih nilai maksimumnya. Sehingga hasil dari proses tersebut dapat dilihat pada kumpulan *grid* di sebelah kanannya. Proses tersebut memastikan fitur yang didapatkan akan sama meskipun objek citra mengalami translasi (pergeseran).



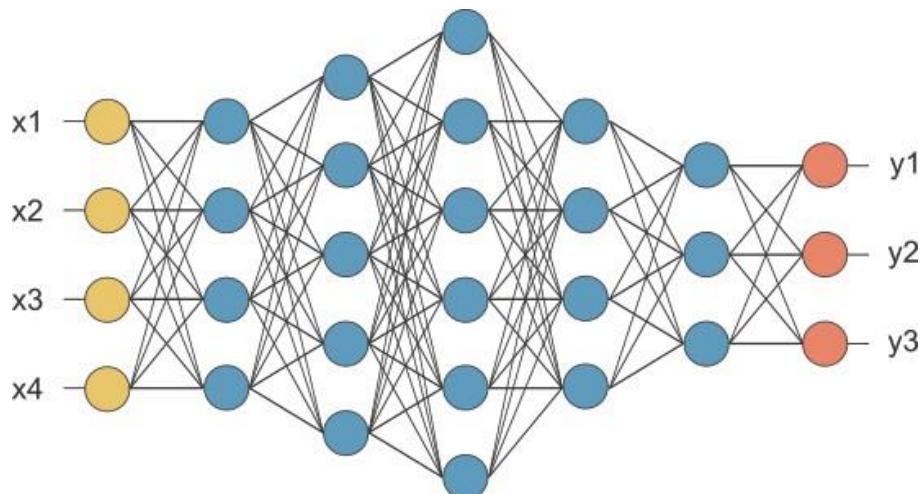
Gambar 2.8 *Pooling*
Sumber: (Rohim dkk., 2019)

2.2.4.3.Fully Connected

Menurut (Arrofiqoh & Harintaka, 2018):

Fully connected merupakan kumpulan dari proses konvolusi. Lapisan ini mendapatkan dari proses sebelumnya untuk menentukan fitur mana yang paling berkorelasi dengan kelas tertentu. Fungsi dari lapisan ini adalah untuk menyatukan semua *node* menjadi satu dimensi.

Setelah melewati proses-proses di *features extraction layer*, hasil dari *pooling layer* digunakan menjadi masukan untuk *Fully connected layer*. Namun, sebelum masuk ke layer ini terdapat lapisan *flatten* yang berfungsi untuk membentuk ulang *feature map* dari lapisan-lapisan yang sudah di ekstraksi fitur sebelumnya menjadi sebuah vektor agar dapat digunakan sebagai masukkan dari *fully-connected layer*. *Fully Connected Layer* ini memiliki kesamaan struktur dengan *Artificial Neural Network* (ANN) pada umumnya yaitu memiliki layer masukkan, *hidden layer* dan layer keluaran yang masing-masing memiliki *neuron-neuron* yang saling terhubung dengan *neuron-neuron* di layer tetangganya. Gambar 2.9 ini merupakan contoh *Fully Connected Layer*.



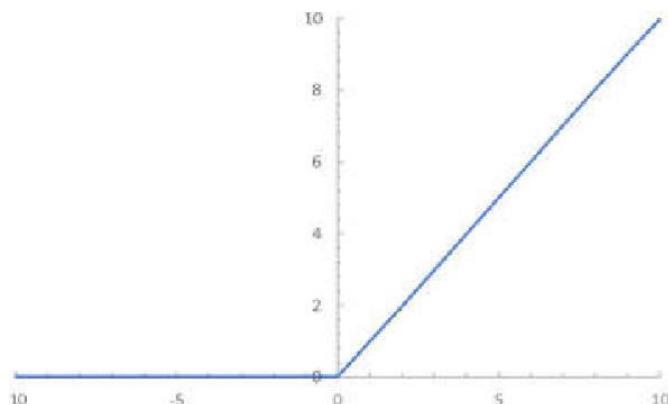
Gambar 2.9 *Fully Connected*
Sumber: (Hidarlan, 2020)

2.2.4.4. *ReLU* Aktivasi

Menurut (Ilahiyah & Nilogiri, 2018):

Aktivasi *ReLU* (*Rectified Linear Unit*) merupakan lapisan aktivasi pada model CNN yang mengaplikasikan fungsi $f(x)=\max(0,x)$ yang berarti fungsi ini melakukan *thresholding* dengan nilai nol terhadap nilai piksel pada *input* citra. Aktivasi ini membuat seluruh nilai piksel yang bernilai kurang dari nol pada suatu citra akan dijadikan 0.

ReLU merupakan fungsi aktivasi yang akan menghasilkan nilai nol apabila $x < 0$ dan kemudian linier dengan kemiringan 1 ketika $x > 0$. *ReLU* akan menghilangkan penghilangan/penyusutan gradien (*vanishing gradient*) dengan cara menerapkan fungsi aktivasi elemen sebagai $f(x)=\max[0](0,x)$ yaitu aktivasi elemen akan dilakukan saat berada di ambang batas 0. Berikut bentuk dari fungsi aktivasi *ReLU* disajikan pada gambar 2.10



Gambar 2.10 *Rectified Linear Unit* (*ReLU*)
Sumber: (Hidarlan, 2020)

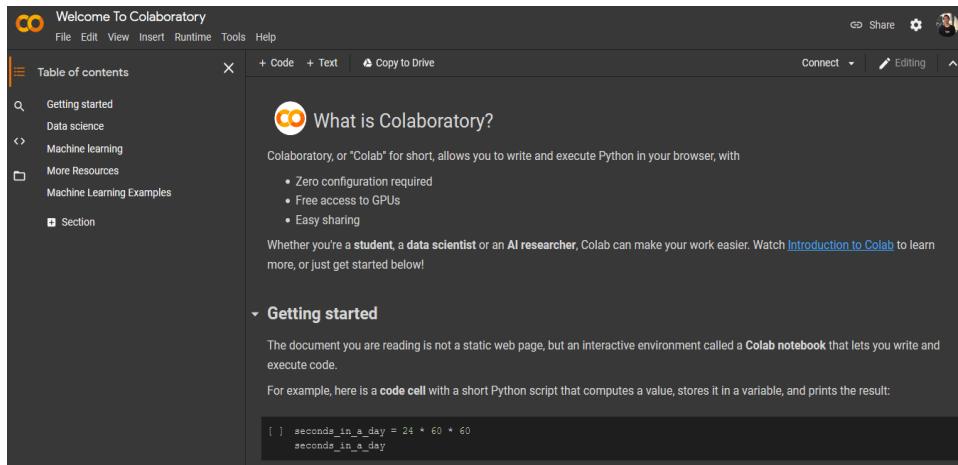
Fungsi ini memiliki kelebihan yaitu dapat mempercepat gradien stokastik di bandingkan dengan fungsi *sigmoid/tan h* karena *ReLU* berbentuk linear serta tidak menggunakan operasi eksponensial seperti *sigmoid/tan h*, sehingga bisa melakukan dengan pembuatan matriks aktivasi saat ambang batas berada pada nilai 0.

Namun terdapat kelemahan yang dapat rapuh saat masa *training* dan mati karena gradien besar yang mengalir melalui *ReLU* menyebabkan pembaruan bobot, sehingga neuron tidak aktif pada data *point* lagi. Jika ini terjadi, maka gradien yang mengalir melalui unit akan selamanya nol dari titik itu. Artinya, unit *ReLU* dapat mati secara *ireversibel* (tidak dapat berubah) selama pelatihan karena mereka dapat melumpuhkan data *manifold*. Misalnya, Anda mungkin menemukan bahwa sebanyak 40% dari jaringan Anda dapat “mati” (yaitu neuron yang tidak pernah aktif di seluruh *dataset* pelatihan) jika tingkat pembelajaran ditetapkan terlalu tinggi. Dengan pengaturan tingkat pembelajaran yang tepat, ini lebih jarang menjadi masalah.

2.2.5. Google Colaboratory

Berdasarkan laman <https://colab.research.google.com/notebooks/welcome> diakses pada tanggal 31 Desember 2020 Google *Colaboratory* adalah layanan berbasis *cloud* Google yang mereplika *Jupyter Notebook* *di-cloud*.

Dengan menggunakan Google *Colaboratory* tidak perlu menginstal apa pun di sistem untuk menggunakannya. Dalam sebagian besar hal, penggunaan *Colaboratory* hampir sama seperti yang dilakukan pada instalasi desktop *Jupyter Notebook*. Google *Colaboratory* ditujukan untuk para pembaca yang menggunakan sesuatu selain dari pengaturan desktop standar untuk mengerjakan contoh-contoh. Berikut tampilan antarmukanya pada gambar 2.9.



Gambar 2.11 Antarmuka *Google Colaboratory*

Dalam menggunakan *Colaboratory*, diwajibkan memiliki akun Google untuk mengakses *Colaboratory* agar semua fitur yang ada pada *Colaboratory* dapat berfungsi dengan baik.

Seperti halnya dengan *Jupyter Notebook*, dengan menggunakan *Colaboratory* dapat melakukan tugas-tugas tertentu dalam paradigma berorientasi sel. Tampilan antarmuka antara *Jupyter Notebook* dengan *Colaboratory* sangat mirip. Pada *Colaboratory* dapat membuat berbagai jenis sel dan menggunakannya untuk membuat buku catatan.

2.2.6. Keras Dan TensorFlow

Menurut (Miceli dkk., 2018):

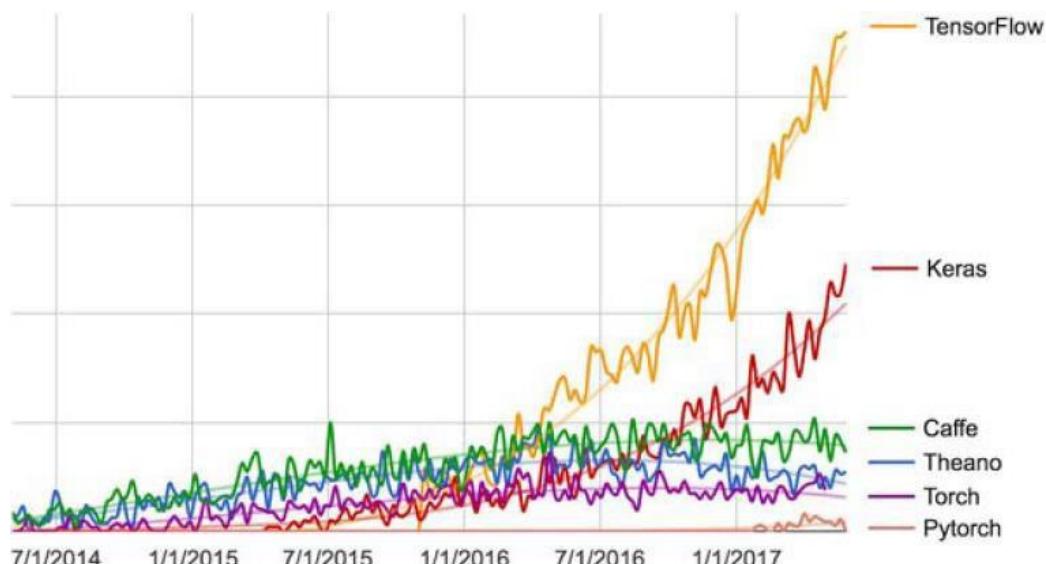
Keras is a deep-learning framework for Python that provides a convenient way to define and train almost any kind of deep-learning model. Keras was initially developed for researchers, with the aim of enabling fast experimentation.

Keras adalah sebuah *framework deep learning* untuk *Python* yang menyediakan cara mudah untuk mendefinisikan dan melatih hampir semua jenis

model *deep learning*. Keras awalnya dikembangkan untuk para peneliti, dengan tujuan memungkinkan eksperimen cepat. Keras memiliki fitur utama berikut:

- a. Keras memungkinkan kode yang sama berjalan mulus di CPU atau GPU.
- b. Memiliki API (*Application Programming Interface*) yang *user-friendly* yang membuatnya mudah untuk cepat prototipe model *deep learning*.
- c. Memiliki dukungan bawaan untuk *convolutional networks* (untuk visi komputer), *recurrent network* (untuk pemrosesan urutan), dan kombinasi keduanya.
- d. Mendukung arsitektur jaringan yang diinginkannya: model *multi-input* atau *multi-output*, berbagi lapisan, berbagi model, dan sebagainya. Ini berarti Keras tepat untuk membangun model *deep learning* apa pun.

Keras didistribusikan di bawah lisensi MIT permissif, yang berarti dapat digunakan secara bebas dalam proyek komersial. *Framework* ini kompatibel dengan versi *Python* apa pun mulai 2,7 hingga 3,6.



Gambar 2.12 Minat *Framwork Deep Learning* Berdasarkan Pencarian Google
Sumber: (Miceli dkk., 2018)

Menurut (Miceli dkk., 2018):

Keras is a model-level library, providing high-level building blocks for developing deep-learning models. It doesn't handle low-level operations such as tensor manipulation and differentiation. Instead, it relies on a specialized, well-optimized tensor library to do so, serving as the backend engine of Keras. Rather than choosing a single tensor library and tying the implementation of Keras to that library, Keras handles the problem in a modular way. thus several different backend engines can be plugged seamlessly into Keras. Currently, the three existing backend implementations are the TensorFlow backend, the Theano backend, and the Microsoft Cognitive Toolkit (CNTK) backend. In the future, it's likely that Keras will be extended to work with even more deep-learning execution engines.

Keras adalah *library* tingkat model, menyediakan blok-blok tingkat tinggi untuk mengembangkan model *deep learning*. Keras tidak dapat menangani operasi tingkat rendah seperti manipulasi *tensor* dan diferensiasi. Sebaliknya, ia bergantung pada *library tensor* khusus yang dioptimalkan dengan baik untuk melakukannya, berfungsi sebagai mesin *backend* dari Keras. Daripada memilih *library tensor* tunggal dan mengimplementasikan Keras ke *library* tersebut, Keras menangani masalah dengan cara modular sehingga beberapa mesin *backend* yang berbeda dapat dihubungkan dengan lancar ke Keras. Saat ini, tiga implementasi *backend* yang ada adalah *backend TensorFlow*, *backend Theano*, dan *backend Microsoft Cognitive Toolkit (CNTK)*. Di masa depan, kemungkinan Keras akan diperluas untuk bekerja lebih dengan mesin eksekusi *deep learning*. Gambar 2.13 merupakan susunan perangkat lunak dan perangkat keras dari *deep learning*.



Gambar 2.13 Susunan perangkat lunak dan perangkat keras *deep learning*
Sumber: (Miceli dkk., 2018)

Melalui *TensorFlow* (atau *Theano*, atau *CNTK*), Keras dapat berjalan dengan mulus di CPU dan GPU. Saat berjalan pada CPU, *TensorFlow* sendiri melibatkan *library* tingkat rendah untuk operasi *tensor* yang disebut *Eigen*. Pada GPU, *TensorFlow* melibatkan *library* operasi *deep learning* yang dioptimalkan dengan baik yang disebut *Library NVIDIA CUDA Deep Neural Network* (cuDNN).

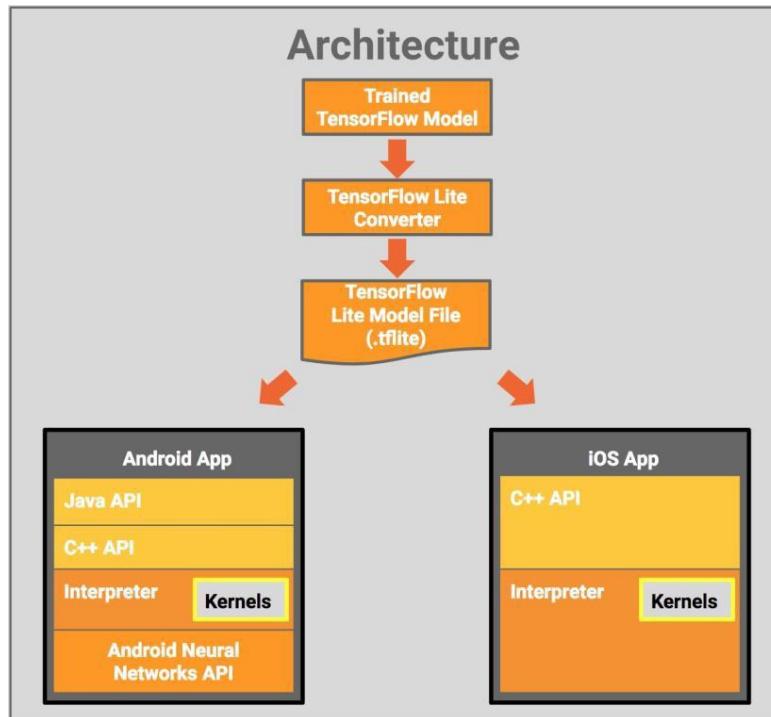
2.2.7. *Tensorflow Lite*

Menurut (Fachriyan & Dharmayanti, 2019):

TensorFlow Lite adalah solusi *TensorFlow* untuk *mobile phone* dan perangkat tertanam. *TensorFlow* akan menjalankan model mesin pencari di perangkat *mobile* dengan penglihatan rendah. Jadi kamu dapat mengambil keuntungan dari klasifikasi yang ada. Pemulihan atau apa pun yang mungkin diinginkan tanpa harus berkunjung terus menerus ke server.

TensorFlow Lite adalah solusi ringan *TensorFlow* untuk perangkat seluler dan tertanam . *TensorFlow* itu sendiri adalah sebuah *end-to-end open source* platform yang digunakan untuk *machine learning*. Ini memungkinkan Anda menjalankan model dari mesin yang dipelajari (*machine-learned*) pada perangkat seluler dengan latensi yang rendah, sehingga dapat dimanfaatkan untuk melakukan klasifikasi, regresi, atau apa pun yang diinginkan tanpa harus melakukan akses

pengiriman dan penerimaan dari atau ke server. Arsitektur dari *TensorFlow Lite* dapat dilihat pada gambar 2.14 berikut.



Gambar 2.14 Arsitektur *TensorFlow Lite*
Sumber: (*TensorFlow.*, 2018)

TensorFlow Lite didukung pada Android dan iOS melalui C++ API (*Application Program Interface*), serta memiliki *Java Wrapper* untuk Pengembang Android. Selain itu, pada Perangkat Android yang mendukungnya, penerjemah (interpreter) juga dapat menggunakan *Android Neural Networks API* untuk akselerasi perangkat keras, jika tidak maka akan diproses dengan standar CPU (*Central Processing Unit*) untuk dieksekusi.

TensorFlow Lite terdiri dari *runtime* yang dapat digunakan untuk menjalankan model yang sudah ada sebelumnya, dan seperangkat alat (*tools*) yang dapat digunakan untuk menyiapkan model yang akan digunakan pada perangkat seluler dan tertanam.

TensorFlow Lite bukan dirancang untuk model pelatihan, akan tetapi untuk melatih modelnya dilakukan pada mesin bertenaga yang lebih tinggi, dan kemudian dikonversi modelnya ke dalam format “.TFLITE”, yang kemudian model tersebut dimuat ke dalam penerjemah seluler (*mobile interpreter*).

2.2.8. Android Studio

Menurut (Andi, 2015):

Android studio adalah IDE (*Integrated Development Environment*) resmi untuk pengembangan aplikasi Android dan bersifat open *source* atau gratis. Peluncuran Android Studio ini diumumkan oleh Google pada 16 Mei 2013 pada *event Google I/O Conference* untuk tahun 2013. Sejak saat itu, Android Studio menggantikan *Eclipse* sebagai IDE resmi untuk mengembangkan aplikasi Android.

Berdasarkan laman <https://developer.android.com/studio/intro> diakses pada tanggal 31 Desember 2020 (Google, 2020), Android Studio merupakan sebuah *Integrated Development Environment* (IDE) resmi yang digunakan untuk pengembangan aplikasi Android, berdasarkan *IntelliJ IDEA*. Selain editor kode dan alat pengembang *IntelliJ* yang baik, Android Studio menawarkan lebih banyak fitur yang meningkatkan produktivitas Anda saat membangun aplikasi Android, seperti:

1. Sistem pembangunan berbasis *Gradle* yang fleksibel,
2. Emulator yang cepat dan kaya fitur,
3. Lingkungan terpadu yang dapat mengembangkan untuk semua perangkat Android,
4. Menerapkan perubahan untuk memasukkan kode dan perubahan sumber daya ke aplikasi yang sedang berjalan tanpa memulai ulang aplikasi tersebut,
5. Code *Templates* dan integrasi dengan *GitHub* yang dapat membantu untuk membangun fitur aplikasi umum dan mengimpor kode sampel,

6. Alat dan kerangka kerja (*framework*) pengujian yang luas,
7. *Lint tools* untuk menangkap kinerja, kegunaan, kompatibilitas versi, dan masalah lainnya,
8. Dukungan C++ dan NDK,
9. Dukungan bawaan untuk Google *Cloud Platform*, membuatnya mudah untuk mengintegrasikan Google *Cloud Messaging* dan *App Engine*.

Salah satu tugas Android sebagai *Integrated Development Environment* adalah menyediakan antarmuka untuk membuat aplikasi dan mengelola manajemen *file* yang bisa dibilang kompleks. Bahasa pemrograman yang digunakan adalah Java. Beberapa hal yang dapat dilakukan di Android Studio ini yaitu dapat menulis, mengedit, dan menyimpan proyek beserta berbagai *file* yang berhubungan dengan proyek itu.

Tidak hanya itu, Android Studio juga memberi akses ke Android *Software Development Kit* (SDK). SDK ini dapat dibilang sebagai ekstensi dari kode Java yang memperbolehkannya untuk berjalan dengan mulus di *Device* Android. Jadi, jika Java dibutuhkan untuk menulis programnya, Android SDK diperlukan untuk menjalankan programnya di Android. Untuk menggabungkan keduanya, maka diperlukannya Android Studio. Selain itu, jika menemukan *bug* pada aplikasi yang dibuat maka kita dapat memperbaikinya dengan Android Studio.

Google sendiri terus mencoba mengoptimasi Android Studio untuk memastikan bahwa perangkat lunak ini dapat membantu dalam membangun Aplikasi Android. Ketika kita sedang melakukan koding, mereka menawarkan *live hints* atau petunjuk langsung. Tidak hanya itu, mereka juga akan memberi saran

jika menurut mereka ada yang salah atau ada hal yang bisa membuat koding kita lebih efisien. Selain itu ketika mulai mengetik sebuah baris koding, Android Studio juga akan memberikan saran kode-kode yang bisa digunakan untuk mempercepat proses koding atau jika kita lupa dengan sebuah kode.

2.2.9. Aplikasi

Pengertian aplikasi menurut Kamus Besar Bahasa Indonesia, “Aplikasi adalah penerapan dari rancang sistem untuk mengolah data yang menggunakan aturan atau ketentuan bahasa pemrograman tertentu”

Secara istilah pengertian aplikasi adalah suatu program yang siap untuk digunakan yang dibuat untuk melaksanakan suatu fungsi bagi pengguna jasa aplikasi serta penggunaan aplikasi lain yang dapat digunakan oleh suatu sasaran yang akan dituju. Menurut kamus *computer* eksekutif, aplikasi mempunyai arti yaitu pemecahan masalah yang menggunakan salah satu teknik pemrosesan data aplikasi yang biasanya berpacu pada sebuah komputasi yang diinginkan atau diharapkan maupun pemrosesan data yang di harapkan.

APK adalah Berkas paket aplikasi Android (*Application Package File*, disingkat APK) adalah format berkas yang digunakan untuk mendistribusikan dan memasang *software* dan *middleware* ke ponsel dengan sistem operasi Android, mirip dengan paket EXE pada Windows atau *Deb* pada OS *Debian*.

2.2.10. Kotlin

Menurut (Indrawan dkk., 2019):

Kotlin merupakan bahasa pemrograman modern yang mudah untuk dipelajari, sederhana dan efisien. *Syntax* di dalam *Kotlin* mudah untuk dibaca dan bisa dibilang bahasanya lebih “manusiawi”. *Kotlin* memiliki konstruksi *object-oriented* dan *functional*. *Kotlin* memiliki beberapa fitur

sehingga bahasa ini menjadi lebih unggul dalam pengembangan aplikasi android yaitu bahasa *kotlin* lebih *concise* (ringkas), *safe* (aman), *Interoperable* (*dapat dioperasikan secara bersilangan*), dan *Tool-friendly* (*dapat digunakan di banyak tool*).

Kotlin merupakan bahasa pemrograman ekspresif dan ringkas yang dapat mengurangi *error* kode umum, serta mudah diintegrasikan ke dalam aplikasi yang sudah ada. Di Google I/O 2017, Google telah mengumumkan bahwa pengembangan Android akan semakin mengutamakan *Kotlin*.

Kotlin adalah bahasa pemrograman pragmatis yang menggabungkan paradigma pemrograman berorientasi objek (OOP) dan pemrograman fungsional. *Kotlin* dikembangkan oleh *JetBrains* yang berjalan di atas *Java Virtual Machine* (JVM). *Kotlin* juga bersifat *interoperabilitas* yang artinya Bahasa pemrograman ini dapat digabungkan dengan Bahasa pemrograman *java* dalam satu Project.

2.2.11. Python

Menurut (Syahrudin & Kurniawan, 2018):

Python adalah bahasa pemrograman interpretatif multiguna dengan filosofi perancangan yang berfokus pada tingkat keterbacaan kode. *Python* diklaim sebagai bahasa yang menggabungkan kapabilitas, kemampuan, dengan sintaksis kode yang sangat jelas, dan dilengkapi dengan fungsionalitas pustaka standar yang besar serta komprehensif. *Python* juga didukung oleh komunitas yang besar.

Python merupakan bahasa pemrograman dinamis yang mendukung pemrograman berbasis objek. *Python* didistribusikan dengan beberapa lisensi yang berbeda dari beberapa versi. Namun pada prinsipnya *Python* dapat diperoleh dan dipergunakan secara bebas, bahkan untuk kepentingan komersial. Karena lisensi *Python* tidak bertentangan baik menurut definisi *Open Source* maupun *General Public License* (GPL).

Nama *Python* berasal dari acara televisi yaitu Monty Python's Flying Circus. Python dikembangkan oleh Guido van Rossum pada tahun 1990-an di CWI, Amsterdam sebagai kelanjutan dari bahasa pemrograman ABC. Versi terakhir yang dikeluarkan CWI adalah 1.2. Tahun 1995, Guido pindah ke CNRI sambil terus melanjutkan pengembangan *Python*. Versi terakhir yang dikeluarkan adalah 1.6. Tahun 2000, Guido dan para pengembang inti *Python* pindah ke BeOpen.com yang merupakan sebuah perusahaan komersial dan membentuk BeOpen PythonLabs. Python 2.0 dikeluarkan oleh BeOpen. Setelah mengeluarkan *Python* 2.0, Guido dan beberapa anggota tim PythonLabs pindah ke DigitalCreations. Saat ini pengembangan *Python* terus dilakukan oleh sekumpulan pemrogram yang di koordinir Guido dan *Python Software Foundation*. *Python Software Foundation* adalah sebuah organisasi non-profit yang dibentuk sebagai pemegang hak cipta intelektual *Python* sejak versi 2.1 dan dengan demikian mencegah *Python* dimiliki oleh perusahaan komersial. Saat ini distribusi *Python* sudah mencapai versi 2.7.13 dan versi 3.6.0.

2.2.12. API

Menurut (Hasan *dkk.*, 2016):

API adalah aplikasi pemrograman yang secara khusus dikembangkan untuk digunakan sebagai perantara komunikasi antara komponen-komponen perangkat lunak. Biasanya, hasil *output* dari API dapat berupa data XML ataupun JSON, tergantung dari situs mana yang menyediakan API tersebut.

API (*Application Programming Interface*) adalah sebuah *interface* yang dapat menghubungkan aplikasi satu dengan aplikasi lainnya. API merupakan perantara antar berbagai aplikasi berbeda, baik dalam satu platform yang sama atau

lintas platform. API juga bisa digunakan untuk menghubungkan bahasa pemrograman yang berbeda.

2.2.13. *Unified Modeling Language*

Menurut (Suendri, 2018):

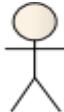
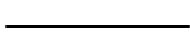
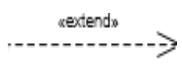
Unified Modeling Language (UML) adalah bahasa spesifikasi standar yang dipergunakan untuk mendokumentasikan, menspesifikasikan dan membangun perangkat lunak. UML merupakan metodologi dalam mengembangkan sistem berorientasi objek dan juga merupakan alat untuk mendukung pengembangan sistem.

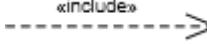
Unified Modeling Language (UML) adalah sebuah bahasa yang berdasarkan grafik atau gambar untuk memvisualisasi, menspesifikasikan, membangun, dan pendokumentasian dari sebuah sistem pengembangan *software* berbasis OO (*Object-Oriented*). UML sendiri juga memberikan standar penulisan sebuah sistem *blue print*, yang meliputi konsep bisnis proses, penulisan kelas-kelas dalam bahasa program yang spesifik, skema *database*, dan komponen-komponen yang diperlukan dalam sistem *software*. Diagram *Unified Modelling Language* (UML) antara lain sebagai berikut:

1. *Use Case Diagram*

Use case menggambarkan *external view* dari sistem yang akan kita buat modelnya. Model *use case* dapat dijabarkan dalam diagram *use case*, tetapi perlu diingat, diagram tidak identik dengan model karena model lebih luas dari diagram. *Use case* harus mampu menggambarkan urutan aktor yang menghasilkan nilai terukur. Berikut adalah simbol-simbol yang ada pada diagram *use case*:

Tabel 2.1 Simbol *Use Case Diagram*

	ACTOR Orang proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari <i>actor</i> adalah gambar orang, biasanya dinyatakan menggunakan kata benda di awal frase nama <i>actor</i> .
	USE CASE Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau <i>actor</i> biasanya dinyatakan dengan menggunakan kata kerja di awal frase nama <i>use case</i> .
	ASOSIASI/ASSOCIATION Komunikasi antara <i>actor</i> dan <i>use case</i> yang berpartisipasi pada <i>use case</i> atau <i>use case</i> memiliki interaksi dengan <i>actor</i> .
	EKSTENSI/EXTEND Relasi <i>use case</i> tambahan ke sebuah <i>use case</i> di mana <i>use case</i> yang ditambahkan dapat berdiri sendiri walaupun tanpa <i>use case</i> tambahan memiliki nama depan yang sama dengan <i>use case</i> yang ditambahkan.
	GENERALISASI/GENERALIZATION Hubungan generalisasi dan spesialisasi (umum-khusus) antara dua buah <i>use case</i> di mana fungsi yang satu adalah

	fungsi yang lebih umum dari lainnya.
	<p>MENGGUNAKAN/INCLUDE</p> <p>Relasi <i>use case</i> tambahan ke sebuah <i>use case</i> di mana <i>use case</i> yang ditambahkan memerlukan <i>use case</i> ini untuk menjalankan fungsional atau sebagai syarat dijalankan <i>use case</i> ini.</p>

Sumber: (Heriyanto, 2018)

2. Class Diagram

Kelas sebagai suatu set objek yang memiliki atribut dan perilaku yang sama, kelas kadang disebut kelas objek. *Class* memiliki tiga area pokok yaitu :

- Nama, kelas harus mempunyai sebuah nama.
- Atribut, adalah kelengkapan yang melekat pada kelas. Nilai dari suatu kelas hanya bisa diproses sebatas atribut yang dimiliki.
- Operasi, adalah proses yang dapat dilakukan oleh sebuah kelas, baik pada kelas itu sendiri ataupun kepada kelas lainnya.

Berikut adalah simbol-simbol yang ada pada diagram *class*:

Tabel 2.2 Simbol *Class Diagram*

GAMBAR	NAMA	KETERANGAN
—	<i>Generalization</i>	Hubungan di mana objek anak (<i>descendent</i>) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (<i>ancestor</i>).
◇	<i>Nary Association</i>	Upaya untuk menghindari asosiasi dengan lebih dari 2 objek.

	<i>Class</i>	Himpunan dari objek-objek yang berbagi atribut serta operasi yang sama.
	<i>Collaboration</i>	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu <i>actor</i>
	<i>Realization</i>	Operasi yang benar-benar dilakukan oleh suatu objek.
	<i>Dependency</i>	Hubungan di mana perubahan yang terjadi pada suatu elemen mandiri (<i>independent</i>) akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri
	<i>Association</i>	Apa yang menghubungkan antara objek satu dengan objek lainnya

Sumber: (Heriyanto, 2018)

3. Activity Diagram

Diagram *activity* menunjukkan aktivitas sistem dalam bentuk kumpulan aksi-aksi, bagaimana masing-masing aksi tersebut dimulai, keputusan yang mungkin terjadi hingga berakhirnya aksi. *Activity diagram* juga dapat menggambarkan proses lebih dari satu aksi salam waktu bersamaan. Diagram *activity* adalah aktivitas-aktivitas, objek, *state*, transisi *state* dan *event*. Dengan kata lain kegiatan diagram alur kerja menggambarkan perilaku sistem untuk aktivitas.

Berikut adalah simbol-simbol yang ada pada diagram *class*:

Tabel 2.3 Simbol *Activity Diagram*

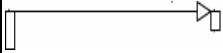
	STATUS AWAL/INITIAL Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal.
	AKTIVITAS/ ACTIVITY Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja.
	PERCABANGAN / DECISION Asosiasi percabangan di mana lebih dari satu aktivitas digabungkan menjadi satu.
	PENGGABUNGAN/ JOIN Asosiasi penggabungan di mana lebih dari satu aktivitas lebih dari satu.
	STATUS AKHIR / FINAL Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status satu.
	SWIMLINE Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi.

Sumber: (Heriyanto, 2018)

4. Sequence Diagram

Secara mudahnya *sequence diagram* adalah gambaran tahap demi tahap, termasuk kronologi (urutan) perubahan secara logis yang seharusnya dilakukan untuk menghasilkan sesuatu sesuai dengan *use case diagram*.

Tabel 2.4 Simbol *Sequence Diagram*

GAMBAR	NAMA	KETERANGAN
	<i>LifeLine</i>	Objek <i>entity</i> , antarmuka yang saling berinteraksi.
	<i>Message</i>	Spesifikasi dari komunikasi antar objek yang memuat informasi-informasi tentang aktivitas yang terjadi
	<i>Message</i>	Spesifikasi dari komunikasi antar objek yang memuat informasi-informasi tentang aktivitas yang terjadi

Sumber: (Heriyanto, 2018)

2.3. Konsep Dasar Pengujian Perangkat Lunak

Salah satu titik krusial dalam sebuah proses pengembangan aplikasi, baik itu pada platform web, desktop, bahkan *mobile* adalah pada proses pengujinya. Pengujian yang baik akan menentukan hasil yang baik pula.

Pengujian aplikasi khususnya Android dapat dilakukan dengan menggunakan cara manual ataupun otomatis (*automation test*). Cara pertama yaitu dengan melakukan pengujian langsung terhadap APK yang di-deploy ke peranti

atau emulator. Pengujian ini membutuhkan ketelitian dari para penguji (*quality assurance*).

Pengujian bisa dilakukan dengan metode *whitebox* dan *blackbox* testing. Pengujian jenis ini akan memakan waktu, membosankan dan rawan akan terjadinya kesalahan atau *human error*. Salah satu pendekatan modern dan efisien adalah dengan menggunakan *automation* testing. Untuk Android sendiri sudah banyak *tools* seperti *Junit*, *Robolectric*, *Mockito*, dan *Espresso* yang mampu melakukan ini. Alat-alat di atas mendukung beragam metode pengujian. Tidak hanya sekedar menguji secara fungsi atau algoritma yang kita gunakan, tetapi sudah bisa masuk ke ranah pengujian otomatis untuk antarmuka.

Teknologi yang digunakan dalam pengujian *user interface* pada penelitian ini adalah *Library Espresso*.

2.4. Pengujian UI (*User Interface*)

Menurut (Merina, 2017):

User Interface test menguji representasi visual aplikasi, seperti bagaimana dialog terlihat atau perubahan UI apa yang dibuat saat sebuah dialog dipecah. Pertimbangan khusus harus dilakukan jika tes melibatkan komponen UI. Hanya *thread* utama yang diizinkan untuk mengubah UI di Android.

Pengujian antarmuka pengguna (UI) memfokuskan pada pengujian aspek antarmuka pengguna dan interaksi dengan pengguna. Mengenali dan menindaklanjuti masukan pengguna merupakan prioritas tinggi dalam pengujian antarmuka pengguna dan validasi.

2.4.1. Karakteristik Pengujian UI (*User Interface*)

Secara karakteristik pengujian antarmuka mencakup beberapa hal sebagai berikut :

1. *User interface* merupakan media visual untuk berinteraksi dengan pengguna yang terdiri dari beragam komponen pembangun *user interface* seperti *Label*, *Button*, *RadioButton*, dan lain sebagainya.
2. Selama proses pengujian, elemen properti yang menempel pada sebuah komponen *user interface* akan memberi pengaruh terhadap kondisi atau ‘*state*’ dari komponen tersebut.
3. Pengujian *user interface* secara otomatis mampu melakukan pemberian *input* dan *event* pada komponen seperti *click()*, *pressKey()* dan lain sebagainya.
4. Pengujian ini lebih menekankan ke perbandingan antara proses yang dilakukan dengan hasil yang diharapkan dalam sebuah skenario penggunaan.
5. Pengujian *user interface* secara otomatis bergantung penuh pada teknologi yang digunakan. Dalam hal ini *framework* atau *tools* yang digunakan akan mempengaruhi hasil dari proses pengujian yang dilakukan.

2.5. Penelitian Terkait

Dalam beberapa jurnal penelitian terkait yang membahas tentang klasifikasi obyek dengan citra dapat dilihat dari beberapa jurnal sebagai berikut :

1. (Rizal dkk., 2019) dengan penelitian yang berjudul “Klasifikasi Wajah Menggunakan Support Vector Machine (SVM)”. Jurnal tersebut meneliti citra wajah manusia dengan menggunakan metode Support Vector Machine (SVM). Sampel wajah yang digunakan dalam penelitian tersebut adalah 200 sampel. Hasil penelitian tersebut menunjukkan tingkat akurasi di dalam klasifikasi wajah menggunakan metode support vector machine (SVM) dengan rata-rata true detection 90% dan false detection 10%.

2. (Simarmata *dkk.*, 2019) dengan penelitian yang berjudul “Klasifikasi Citra Makanan Menggunakan Algoritme Learning Vector Quantization Berdasarkan Ekstraksi Fitur Color Histogram dan Gray Level Co- occurrence Matrix”. Jurnal tersebut meneliti tentang bagaimana Learning Vector Quantization (LVQ) mendeteksi citra makanan menggunakan ekstraksi fitur histogram. Jumlah sampel yang digunakan pada penelitian tersebut adalah 240 sampel dan menghasilkan akurasi paling besar 67.0%.
3. (Hidarlan, 2020) dengan penelitian yang berjudul “Rancang Bangun Klasifikasi Varietas Beras Berdasarkan Citra Menggunakan Metode *Convolutional Neural Network (CNN)* Berbasis Android”. Penelitian tersebut membahas arsitektur CNN yang dapat digunakan untuk melakukan klasifikasi varietas beras berdasarkan citranya. Arsitektur CNN yang digunakan pada penelitian tersebut adalah VGG16Net dan *MobileNet* dengan menggunakan metode ekstraksi fitur (*feature extraction*). Penelitian tersebut menggunakan 3 kelas atau variabel, yaitu Beras Basmathi, Beras IR-64 dan Beras Ketan. Jumlah sampel yang digunakan sebanyak 180 citra dengan 60 citra per variabelnya. Pada penelitian tersebut dihasilkan nilai akurasi yang paling besar di atas 95% .
4. (Hassan *dkk.*, 2020) dengan penelitiannya yang berjudul “*COVID faster R-CNN: A novel framework to Diagnose Novel Coronavirus Disease (COVID-19) in X-Ray images*”. Penelitian tersebut membahas pencegahan Covid-19 dengan gambar rontgen dada yang diusulkan sebagai bantuan bagi tenaga kesehatan untuk memvalidasi penilaian awal mereka terhadap pasien Covid-

19. Penelitian tersebut menggunakan 183 gambar rontgen dada yang positif Covid-19 dan menggunakan metode R-CNN. Penelitian tersebut telah mengusulkan model pembelajaran mendalam untuk mendeteksi kasus Covid-19 dari gambar rontgen dada. Sistem otomatis tersebut dapat melakukan klasifikasi biner tanpa ekstraksi fitur manual dengan akurasi 97,36%.
5. (Ahsan dkk., 2020) dengan penelitiannya yang berjudul “*COVID-19 Symptoms Detection Based on NasNetMobile with Explainable AI Using Various Imaging Modalities*”. Penelitian tersebut mengusulkan dan menguji delapan model berbasis jaringan saraf konvolusional individu yaitu *VGG16*, *InceptionResNetV2*, *ResNet50*, *DenseNet201*, *VGG19*, *MobileNetV2*, *NasNetMobile* dan *ResNet15V2* untuk mendeteksi pasien Covid-19 menggunakan CT scan dan gambar X-ray dada. Penelitian tersebut menggunakan dua kumpulan data berbeda yang berisi gambar X-ray dan CT scan yang masing-masing 400 gambar yang dikumpulkan dari repositori set data *Kaggle open-source*. Penelitian tersebut manghasilkan pada kumpulan data gambar CT scan, *MobileNetV2* dan *NasNetMobile* mengungguli semua model lainnya, sementara *NasNetMobile* adalah model terbaik pada set data gambar X-ray dada saja. *NasNetMobile* mengungguli beberapa model lain kecuali *MobileNetV2* dengan 95% CI pada set data CT scan, dan akurasi berkisar antara 81,5% hingga 95,2%, dan pada set data gambar X-ray dada, akurasinya bervariasi 95,4% hingga 100%.
6. (Fahri, 2020) dengan penelitiannya yang berjudul “Melihat Peta Penyebaran Pasien Covid-19 Dengan Kombinasi Qgis Dan Framework Laravel”.

Penelitian tersebut membahas tentang pemerintah daerah Kabupaten Ketapang kesulitan dalam melacak sebaran pasien Covid-19. Kesulitan ini dikarenakan data yang digunakan masih dalam bentuk tabel dan titik koordinat. Pada Penelitian tersebut melakukan transformasi dari data tabel yang sulit dipahami menjadi data visualisasi mapping sehingga dapat dengan mudah melihat daerah mana saja yang paling banyak terdapat kasus ODP, PDP, dan Positif. Metode yang dilakukan dalam penelitian tersebut adalah pemetaan digitalisasi dengan menggunakan dua aplikasi yaitu QGIS dan framework Laravel.

Berdasarkan penelitian terkait menunjukkan bahwa metode *convolutional neural network* (CNN) memiliki tingkat akurasi yang lebih tinggi dibandingkan dengan metode lainnya dalam hal pengenalan citra. Penelitian yang dilakukan menggunakan dataset *citra chest ray* dan arsitektur CNN murni. Kebaruan yang dilakukan pada penelitian dibandingkan dengan penelitian terkait yaitu menggunakan metode CNN yang dijalankan secara *realtime* di *mobile phone* Android sehingga memberikan fleksibilitas kepada tenaga kesehatan untuk melakukan diagnosis secara fleksibel dan akurat serta data sebaran akan menjadi informasi pelengkap yang menjadi data *trend* kekinian perkembangan Covid-19.

BAB III

METODOLOGI PENELITIAN

3.1. Waktu dan Lokasi Penelitian

Penelitian ini dilakukan di bulan Desember 2020 sampai dengan Februari 2021. Bertempat di STMIK Dipanegara Makassar.

3.2. Jenis Penelitian

Jenis penelitian yang di gunakan adalah :

a. Penelitian Terapan

Penelitian ini merupakan jenis penelitian terapan yaitu menghasilkan produk berupa aplikasi berbasis *mobile android* yang memberikan informasi hasil pengenalan terhadap citra radiografi pasien Covid-19 yang dilengkapi dengan sebaran Covid-19 di Indonesia. Teknik yang digunakan yaitu Teknik *ekperimental* yaitu melakukan pelatihan dan pengujian secara performansi, dengan menguji akurasi dan kecepatan pengenalan sistem.

Penelitian yang dilakukan memiliki beberapa tahapan dalam mengolah citra yaitu pada tahap pertama melakukan penyiapan dan *pre-proses dataset* selanjutnya tahap memuat *dataset* pelatihan dan *dataset* pengujian setelah *dataset* siap, selanjutnya menyimpan label *dataset* yaitu normal, viral pneumonia, dan Covid-19, setelah itu dilakukan perancangan arsitektur CNN, selanjutnya *dataset* pelatihan di *train* menggunakan arsitektur CNN yang sudah di rancang sebelumnya sehingga menghasilkan sebuah model selanjutnya model tersebut di evaluasi menggunakan *dataset* pengujian yang sudah

disiapkan sehingga menghasilkan akurasi dan pada tahap terakhir akurasi tersebut di ukur menggunakan rumus :

$$\% \text{akurasi} = \frac{\text{jumlah prediksi benar}}{\text{jumlah data pengujian}} \times 100\% .$$

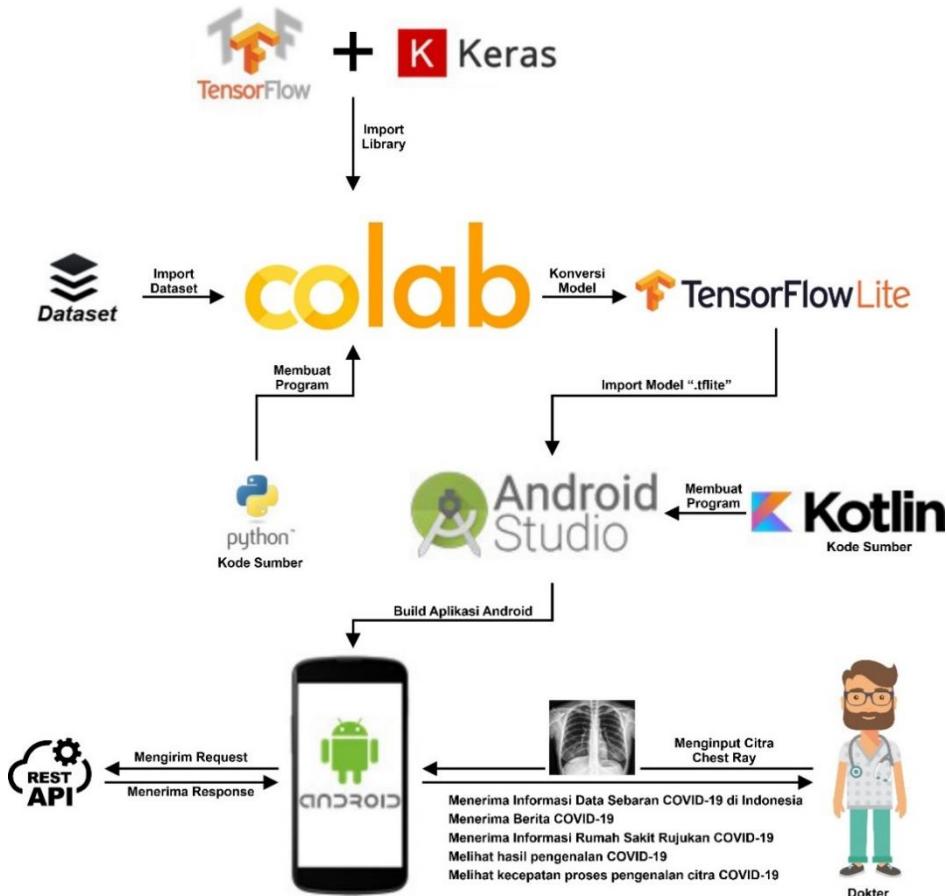
b. Penelitian Kepustakaan

Yaitu penelitian yang dilakukan dengan cara mengumpulkan referensi dari skripsi, jurnal, atau literatur yang terdapat di internet yang mendukung topik penelitian, membandingkan beberapa teori berkaitan dengan Covid-19 dan *convolutional neural network CNN*.

3.3. Teknik Pengumpulan Data

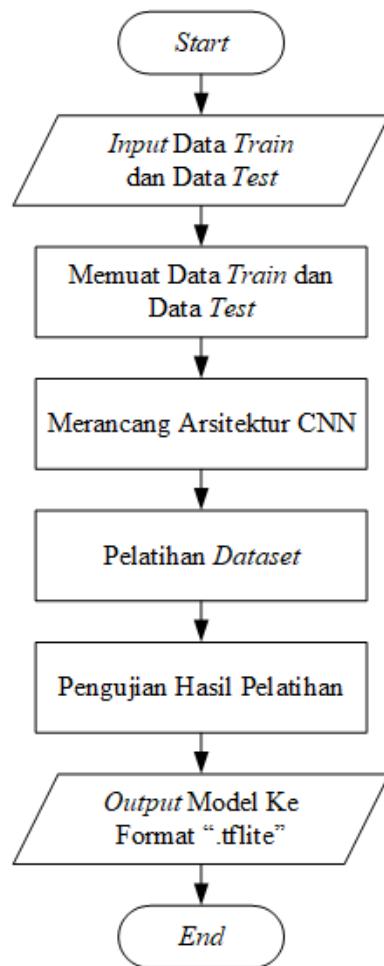
Teknik pengumpulan data dalam penelitian dilakukan dengan cara observasi. Penelitian dilakukan dengan cara mengumpulkan data-data berupa gambar Citra *Chest Ray* yang kemudian menjadi bahan untuk penelitian.

3.4. Arsitektur Sistem



Gambar 3.1 Arsitektur Sistem

Dapat dilihat dari gambar 3.1 bagaimana alur kerja aplikasi, proses awal dimulai dengan melatih *dataset* yang di *import* ke *google colaboratory*. Pembuatan program di *google colaboratory* menggunakan bahasa pemrograman *python* dengan *library tensorflow* dan *keras* untuk membuat model. Selanjutnya model dikonversi menggunakan *tensorflow lite* dan di *import* ke *android studio*. Pembuatan aplikasi android menggunakan bahasa pemrograman *kotlin*. Setelah aplikasi selesai dibuat dokter dapat menginput *citra chest ray* ke aplikasi untuk dikenali, dokter juga dapat melihat informasi sebaran Covid-19 di Indonesia secara *real time*.



Gambar 3.2 Flowchart Pembuatan Model

Dapat dilihat pada gambar 3.2 bahwa pembuatan model dimulai dengan menginput data *train* dan data *test* ke dalam sistem. Selanjutnya semua data di *rescale* menjadi 224x224 piksel dan di *split* sebanyak 0.2°, setelah itu proses perancangan arsitektur CNN, selanjutnya melatih dataset dengan arsitektur yang telah dirancang sehingga menghasilkan model, selanjutnya model tersebut di uji dan disimpan ke format “.tflite”.

3.5. Alat dan Bahan Penelitian

Dalam proses perancangan ini, diperlukan alat dan bahan yang dapat mendukung keberhasilan perancangan. Alat dan bahan yang digunakan adalah :

3.5.1. Alat Penelitian

Tabel 3.1 Perangkat Keras (*hardware*)

Nama Perangkat Keras	Unit	Spesifikasi
Laptop	2	<i>Processor Intel Core i5, RAM 8 GB</i>
Smartphone	2	ROM 64 GB, RAM 6 GB
Kabel Data USB Tipe C	1	1 Meter
Kabel Data Micro USB	1	1 Meter
Wifi Adapter	1	100 Mbps

Tabel 3.2 Perangkat Lunak (*Software*)

Nama Perangkat Lunak	Spesifikasi
Sistem Operasi	Windows 10 64 Bit, Android 9.0 Pie
Editor	Android Studio, Google Colaboratory
Bahasa Pemrograman	<i>Kotlin, Python</i>

3.5.2. Bahan Penelitian

Adapun bahan yang digunakan dalam penelitian berupa *file* gambar yaitu :

1. *File* gambar Citra *Chest Ray* tanpa gejala atau normal.
2. *File* gambar Citra *Chest Ray* dengan gejala Covid-19.
3. *File* gambar Citra *Chest Ray* dengan gejala Viral Pneumonia.

3.6. Metode Pengujian

Metode pengujian yang dilakukan dalam penelitian ini adalah pengujian Instrumental *Test* dengan menggunakan *Library Espresso*. Manfaat utama menggunakan *Espresso* adalah akses ke informasi instrumentasi, seperti konteks

aplikasi, sehingga bisa memantau semua interaksi sistem yang dimiliki sistem Android pada aplikasi. Manfaat utama lainnya adalah secara otomatis menyinkronkan tindakan pengujian dengan UI aplikasi. *Espresso* mendeteksi kapan *thread* utama tidak digunakan, sehingga bisa menjalankan pengujian pada waktu yang tepat, yang akan memperbaiki kendala dalam pengujian. Kemampuan ini juga dapat mempersingkat waktu dalam proses pengujian.

Langkah-langkah pengujian dalam *Espresso*.

1. Membuat skenario pengujian.
2. Melakukan penulisan *code* untuk pengujian.
3. Melakukan pengujian secara otomatis.
4. Jika hasil uji coba sudah sesuai dengan harapan, maka dapat disimpulkan bahwa aplikasi sudah berfungsi dengan baik (bebas dari kesalahan fungsional).

3.7. Tahap Penelitian dan Jadwal Penelitian

3.7.1. Tahap Penelitian

Tahapan yang harus dilalui dalam pembangunan sistem adalah sebagai berikut:

a. Pengumpulan Data

Pada penelitian ini data *citra chest ray* dari pasien berasal dari <https://www.kaggle.com>. Data ini dirilis secara publik yang terdiri dari citra Covid-19, Viral Pneumonia dan Normal. Total data *citra chest ray* yang dikumpulkan adalah 317 data dengan data *train* sebanyak 251 data dan data *test* sebanyak 66 data. *Preprocessing* untuk semua data ini perlu dilakukan sebelum digunakan sebagai *input* pada CNN. Pada penelitian ini,

preprocessing yang dilakukan adalah *rescale* menjadi 224x224 piksel sebagai *standard input*.

b. Analisis Sistem

Pemeriksaan radiologi menjadi tahap awal pengenalan Covid-19 sebelum dilakukan pemeriksaan PCR. Karena proses radiografi hanya membutuhkan waktu yang singkat dan tingkat akurasi yang dihasilkan sama baiknya dengan pemeriksaan PCR.

c. Perancangan Aplikasi

Peneliti merancang suatu aplikasi berbasis android dengan tujuan dapat membantu dokter dalam mengenali Covid-19 dengan mudah dan praktis. Aplikasi yang dibuat bisa dijadikan alternatif dalam proses pengenalan COVID-19 sehingga dapat mempercepat proses pengenalan Covid-19.

d. Penulisan Kode Program

Merupakan proses untuk mentransformasikan rancangan ke dalam kode program menggunakan bahasa pemrograman *Kotlin* dan *Python*.

e. Pengujian Aplikasi

Merupakan proses untuk menguji aplikasi menggunakan *espresso* dan menguji akurasi sistem dalam pengenalan citra.

f. Implementasi.

Implementasi dilakukan secara laboratorium atau internal STMIK Dipanegara sebagai hasil penelitian. Untuk digunakan oleh pakarnya aplikasi yang dibuat masih membutuhkan hasil kepuasan dan validasi dari pihak dokter di rumah sakit.

3.7.2. Rancangan Jadwal Penelitian

Adapun tahapan-tahapan serta jadwal pelaksanaan penelitian yang dilakukan sebagai berikut :

Tabel 3.3 Jadwal Penelitian

BAB IV

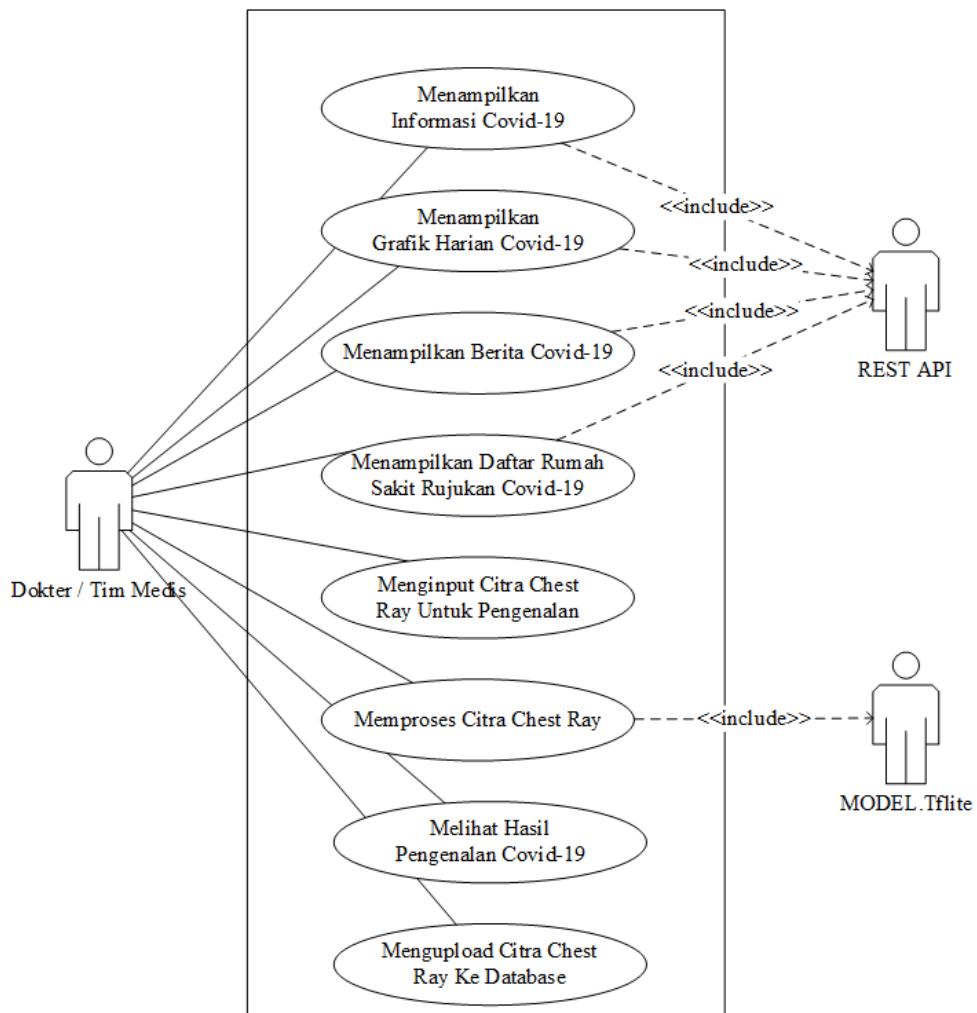
ANALISIS DAN IMPLEMENTASI

4.1. Rancangan Sistem

4.1.1. Use Case Diagram

Use Case Diagram yang dirancang untuk menggambarkan apa yang dilakukan sistem dan siapa saja aktor yang berinteraksi dengan sistem sehingga dapat memahami tentang aplikasi yang akan dibuat.

4.1.1.1. Use Case Aplikasi Covid-19 Detector



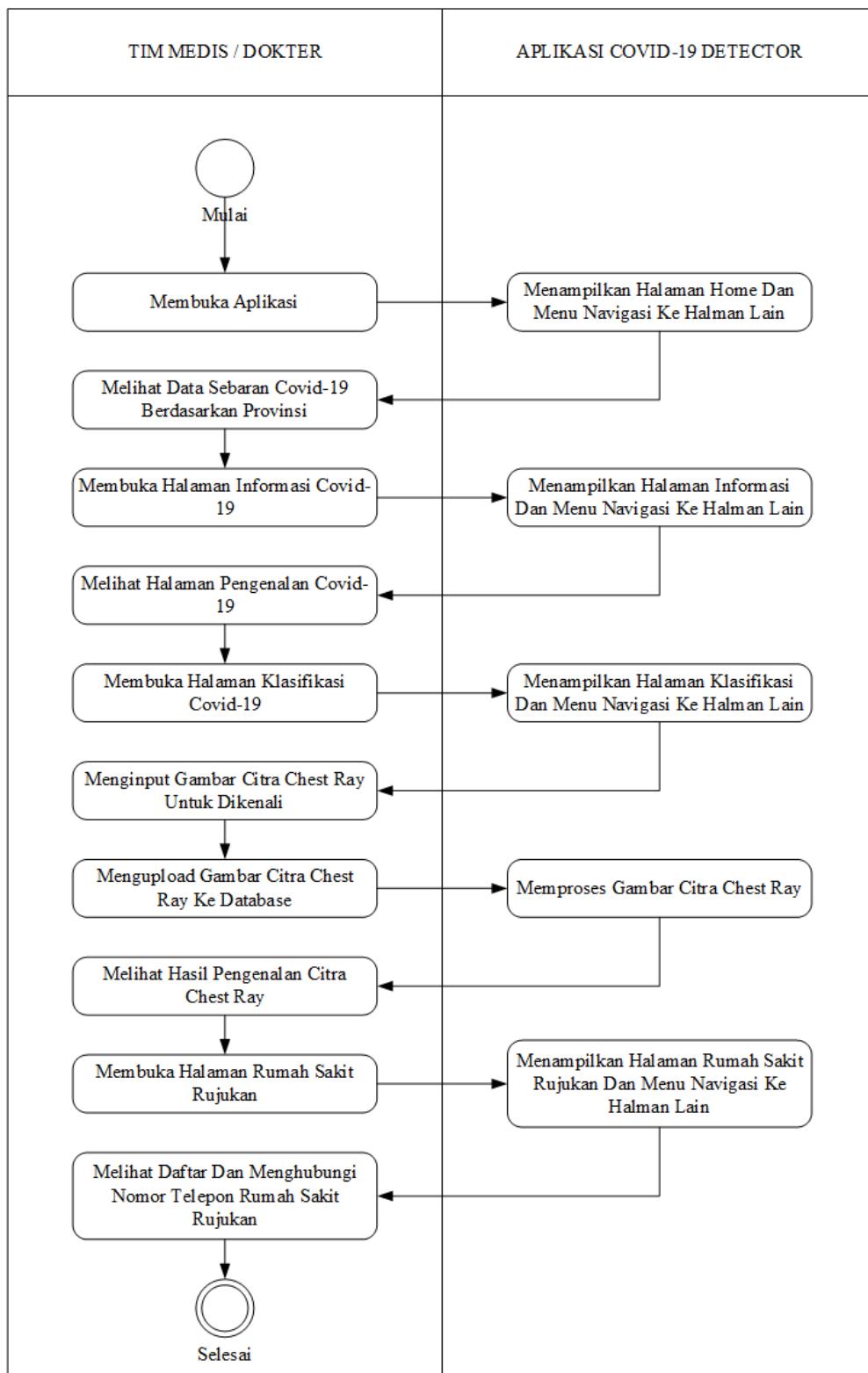
Gambar 4.1 Use Case Diagram Covid-19 Detector

Pada *Use Case Diagram* Covid-19 Detector *user* dapat melihat informasi, grafik harian, berita, dan rumah sakit rujukan Covid-19 secara *real time* yang di dapatkan dari *REST API*. *User* juga dapat *menginput citra chest ray* ke sistem dan mengklasifikasi sehingga *user* dapat melihat hasil klasifikasinya. *User* juga dapat *mengupload citra chest ray* ke *database*.

4.1.2. Activity Diagram

Activity diagram digunakan untuk menggambarkan rangkaian aliran dari aktivitas. *Activity diagram* juga digunakan untuk mendeskripsikan aktivitas yang dibentuk dalam suatu operasi sehingga dapat juga digunakan untuk aktivitas lainnya seperti *use case* atau interaksi.

4.1.2.1. Activity Diagram Covid-19 Detector



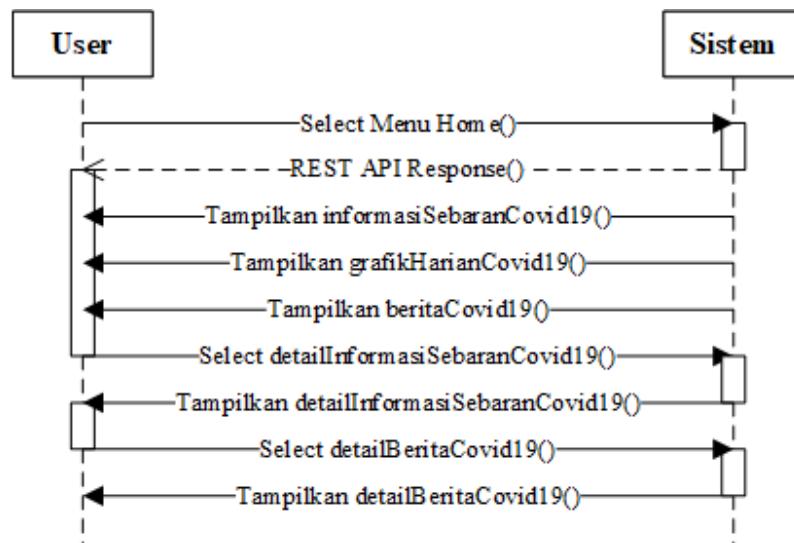
Gambar 4.2 Activity Diagram Covid-19 Detector

Ketika *user* membuka aplikasi, maka sistem akan menampilkan halaman *home* dan menu navigasi ke halaman lain, *user* bisa melihat informasi sebaran Covid-19 berdasarkan provinsi. Selanjutnya *user* memilih menu informasi, maka sistem akan menampilkan halaman informasi tentang pengenalan Covid-19 dan menu navigasi ke halaman lain. Selanjutnya *user* memilih menu klasifikasi, maka sistem akan menampilkan halaman klasifikasi dan menu navigasi ke halaman lain, lalu *user* menginput dan mengupload citra *chest ray* ke aplikasi, maka sistem akan memproses citra *chest ray* tersebut dan menampilkan hasil klasifikasi, selanjutnya *user* memilih menu rumah sakit rujukan, maka sistem akan menampilkan halaman rumah sakit rujukan, *user* dapat menghubungi dan melihat daftar rumah sakit rujukan.

4.1.3. Sequence Diagram

Sequence Diagram merupakan interaksi antara objek-objek dalam suatu sistem dan terjadi komunikasi yang berupa pesan (*message*) serta parameter waktu.

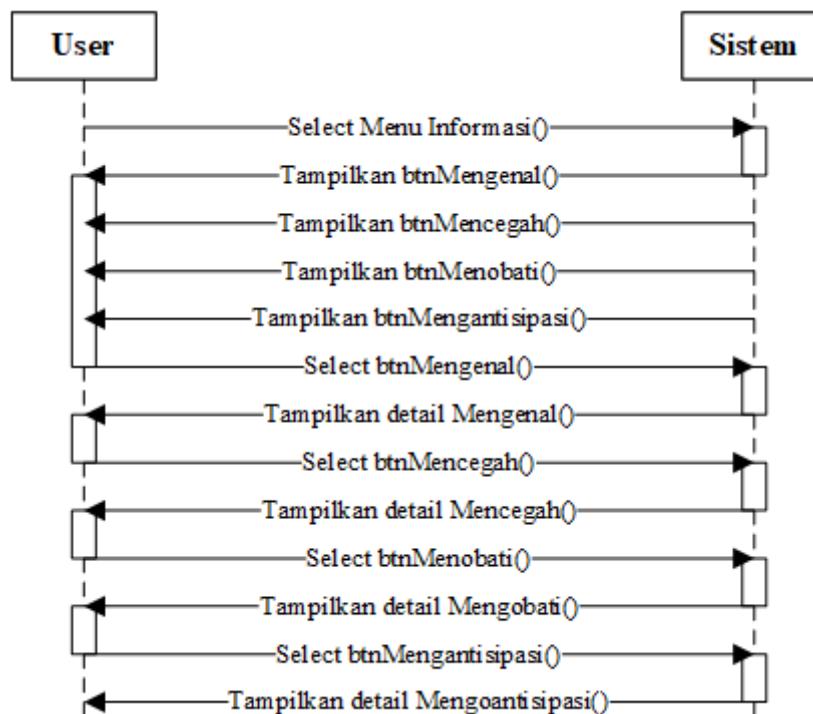
4.1.3.1. Sequence Diagram Menu Home



Gambar 4.3 *Sequence Diagram Menu Home*

Pada *sequence diagram* menu home ketika *user* memilih menu *home* maka sistem akan mengambil data ke REST API secara *asynchronous* dan menampilkan informasi sebaran, grafik harian, dan berita Covid-19. Ketika *user* memilih detail informasi sebaran, maka sistem akan menampilkan detail informasi sebaran Covid-19. Ketika *user* memilih detail berita, maka sistem akan menampilkan detail berita Covid-19.

4.1.3.2. Sequence Diagram Menu Informasi

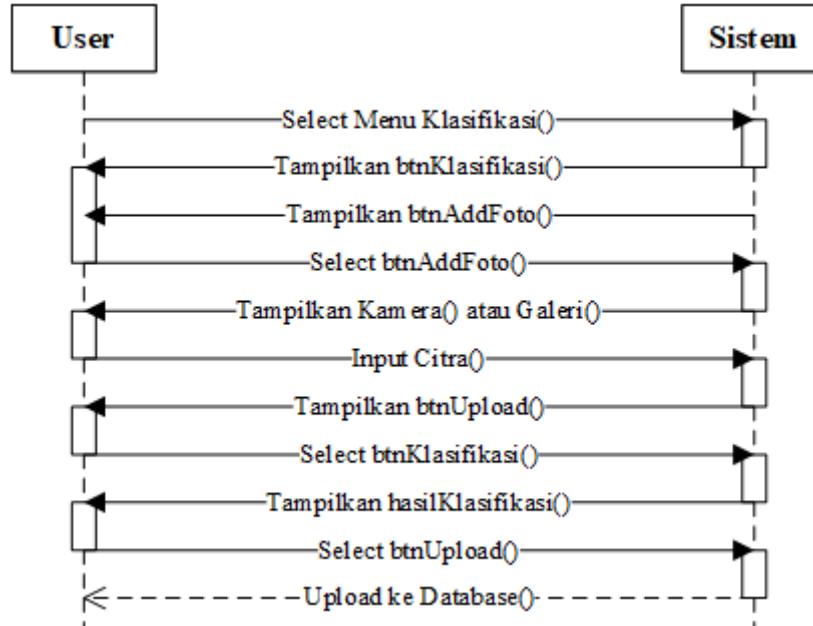


Gambar 4.4 Sequence Diagram Menu Informasi

Pada *sequence diagram* menu informasi ketika *user* memilih menu informasi maka sistem akan menampilkan empat tombol yaitu tombol mengenal, tombol mencegah, tombol mengobati dan tombol mengantisipasi. Ketika *user* memilih tombol mengenal maka sistem akan menampilkan detail mengenal. Ketika *user* memilih tombol mencegah maka sistem akan menampilkan detail mencegah.

Ketika *user* memilih tombol mengobati maka sistem akan menampilkan detail mengobati. Ketika *user* memilih tombol mengantisipasi maka sistem akan menampilkan detail mengantisipasi.

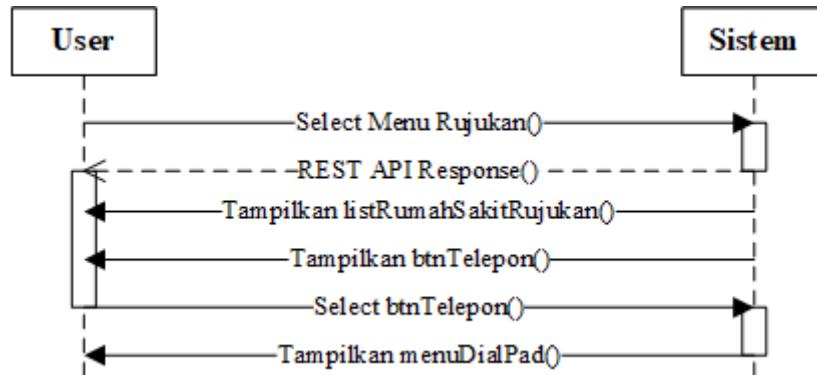
4.1.3.3.Sequence Diagram Menu Klasifikasi



Gambar 4.5 Sequence Diagram Menu Klasifikasi

Pada *sequence diagram* menu klasifikasi ketika *user* memilih menu klasifikasi maka sistem akan menampilkan dua tombol yaitu tombol klasifikasi dan tambahkan foto. Ketika *user* memilih tombol tambahkan foto maka sistem akan menampilkan kamera atau galeri dan *user* bisa *menginput* citra. Ketika *user* memilih tombol klasifikasi maka sistem akan menampilkan hasil klasifikasi. Ketika *user* memilih tombol *upload* maka sistem akan *mengupload* citra ke *database*.

4.1.3.4.Sequence Diagram Menu Rujukan

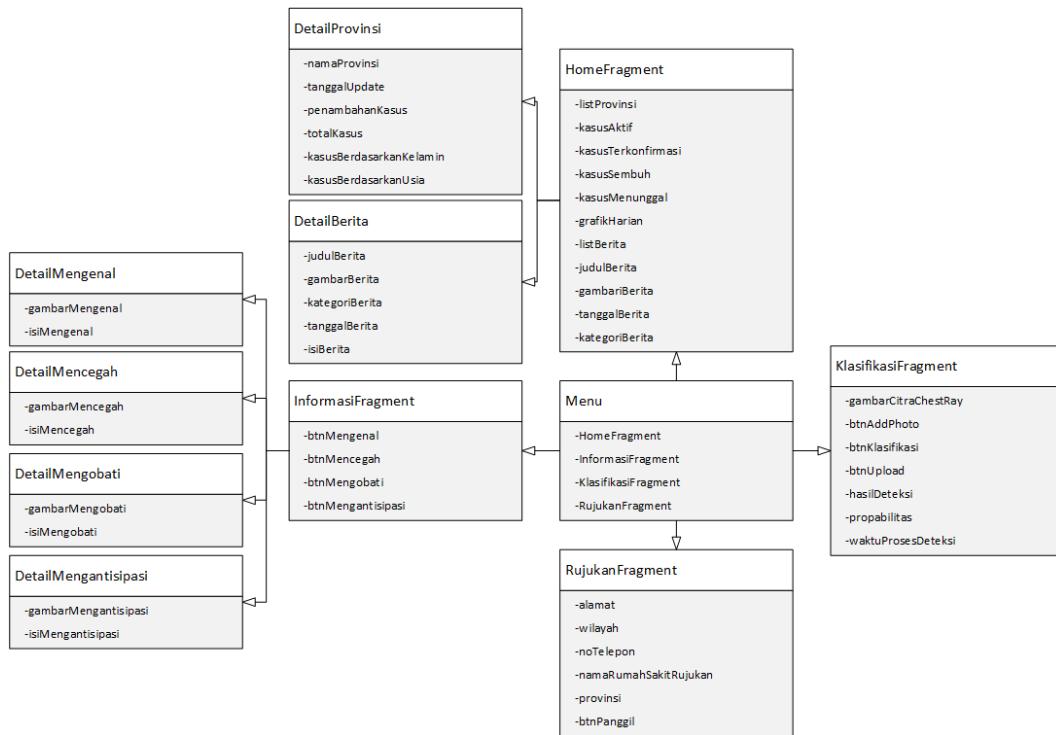


Gambar 4.6 Sequence Diagram Menu Rujukan

Pada *sequence diagram* menu rujukan ketika *user* memilih menu rujukan maka sistem akan mengambil data ke *REST API* secara *asynchronous* dan menampilkan detail *list* rumah sakit rujukan dan tombol panggil, ketika *user* memilih tombol panggil maka sistem akan mengambil nomor rumah sakit rujukan dan membuka *dial pad* di *handphone*.

4.1.4. Class Diagram

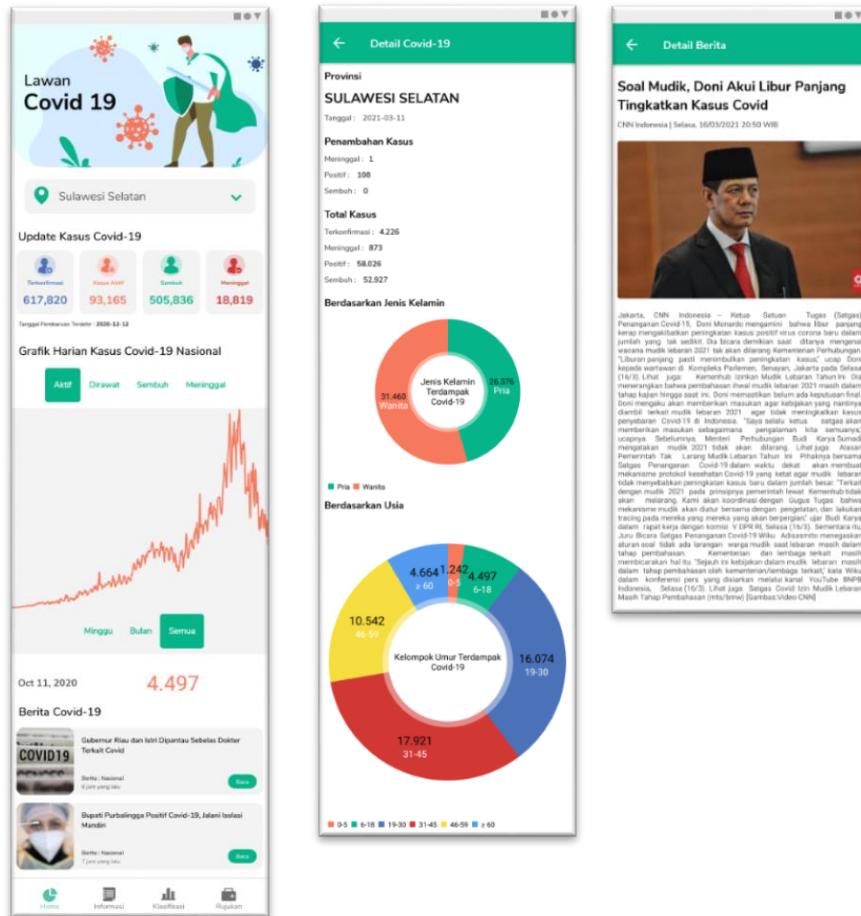
Diagram kelas adalah diagram UML yang menggambarkan kelas-kelas dalam sebuah sistem dan hubungannya antara satu dengan yang lain. Berikut ini adalah tampilan *class diagram* pada aplikasi yang akan dibangun.



Gambar 4.7 *Class Diagram* Aplikasi Covid-19 Detector

4.2. Perancangan Interface

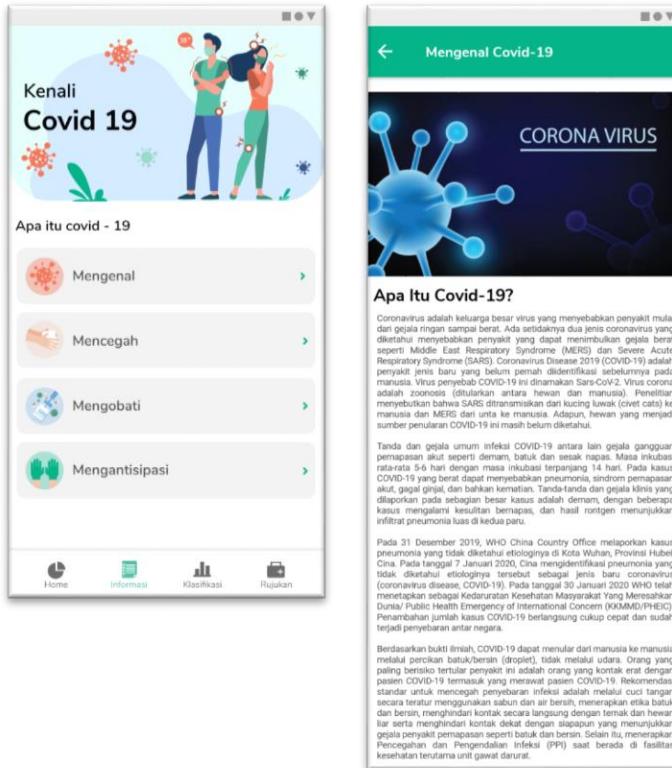
1. Tampilan Home



Gambar 4.8 Tampilan Halaman *Home* dan Detail Informasi Covid-19

Pada tampilan *home* terdapat informasi sebaran Covid-19 berdasarkan provinsi, juga terdapat grafik harian dan berita Covid-19. Pada halaman *home* juga terdapat tombol detail informasi dan detail berita yang akan menampilkan halaman detail informasi dan halaman detail berita.

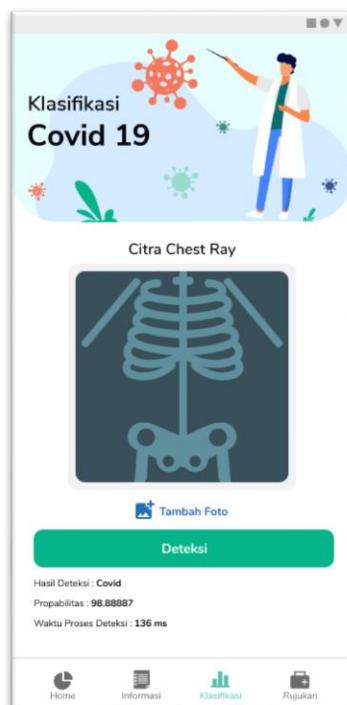
2. Tampilan Informasi



Gambar 4.9 Tampilan Halaman Informasi

Pada rancangan halaman informasi terdapat empat tombol yaitu tombol mengenal, tombol mencegah, tombol mengobati, dan tombol mengantisipasi masing-masing tombol memiliki halaman detail jika di klik untuk lebih jelasnya dapat di lihat pada gambar 4.9.

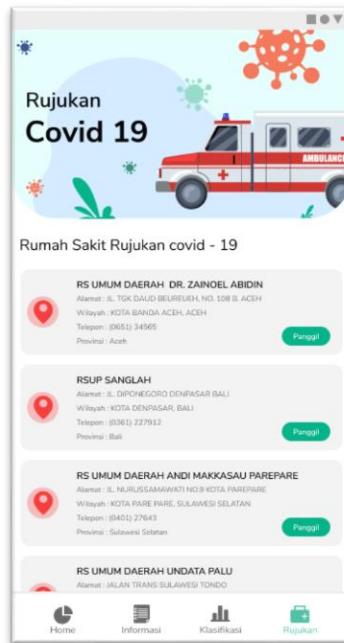
3. Tampilan Klasifikasi



Gambar 4.10 Tampilan Halaman Informasi

Pada rancangan halaman klasifikasi terdapat *imageview*, label untuk menampilkan hasil prediksi dan dua buah tombol yaitu tombol tambah foto dan tombol klasifikasi.

4. Tampilan Rumah Sakit Rujukan



Gambar 4.11 Tampilan Halaman Rujukan

Pada rancangan halaman rumah sakit rujukan terdapat *list* rumah sakit rujukan dan tombol panggil yang berfungsi untuk mengalihkan *user* ke halaman *dial pad*.

4.3. Dataset Penelitian

Penelitian ini menggunakan dua kondisi *dataset* dari ketiga jenis citra *chest ray* yang dikumpulkan. *Dataset* tersebut digunakan untuk melakukan perancangan sistem pada *Google Colaboratory*. Model arsitektur CNN yang diperoleh dari pelatihan pada *Google Colaboratory* disimpan dalam bentuk *TensorFlow Lite* yang nantinya akan digunakan untuk merancang sistem pada perangkat Android.

4.3.1. Dataset

Dataset yang digunakan pada klasifikasi Covid-19 ada tiga macam, yaitu *dataset* pelatihan, *dataset* validasi, dan *dataset* pengujian. *Dataset* pelatihan dan validasi merupakan *dataset* yang akan digunakan untuk melakukan pelatihan untuk

memperoleh model dari arsitektur CNN yang dibuat. Sedangkan *dataset* pengujian digunakan untuk menguji tingkat probabilitas hasil yang benar pada arsitektur CNN tersebut pada *Google Colaboratory*. Seluruh *dataset* tersebut menggunakan gambar berwarna RGB (tiga saluran warna) dan diubah ukurannya menjadi 224x224 *piksel* sesuai dengan masukkan pada arsitektur CNN yang sudah di rancang pada *Google Colaboratory*.

1. Jumlah *Dataset* Pelatihan

Pada penelitian ini *dataset* pelatihan yang digunakan berformat “.jpg” dan *dataset* pelatihan diambil 80% dari folder *train* sehingga jumlah data set pada *chest ray Covid-19* yaitu 368 gambar, pada *chest ray Normal* yaitu 1,012 gambar, dan pada *chest ray Viral Pneumonia* yaitu 2,734 gambar sehingga total *dataset* pelatihan yang digunakan yaitu 4,114 gambar.

2. Jumlah *Dataset* Validasi

Dataset validasi yang digunakan sebanyak 20% dari folder *train* dengan jumlah total 1,018 gambar. Yang masing-masing jumlahnya adalah pada *chest ray Covid-19* yaitu 81 gambar, pada *chest ray Normal* yaitu 253 gambar, dan pada *chest ray Viral Pneumonia* yaitu 684 gambar.

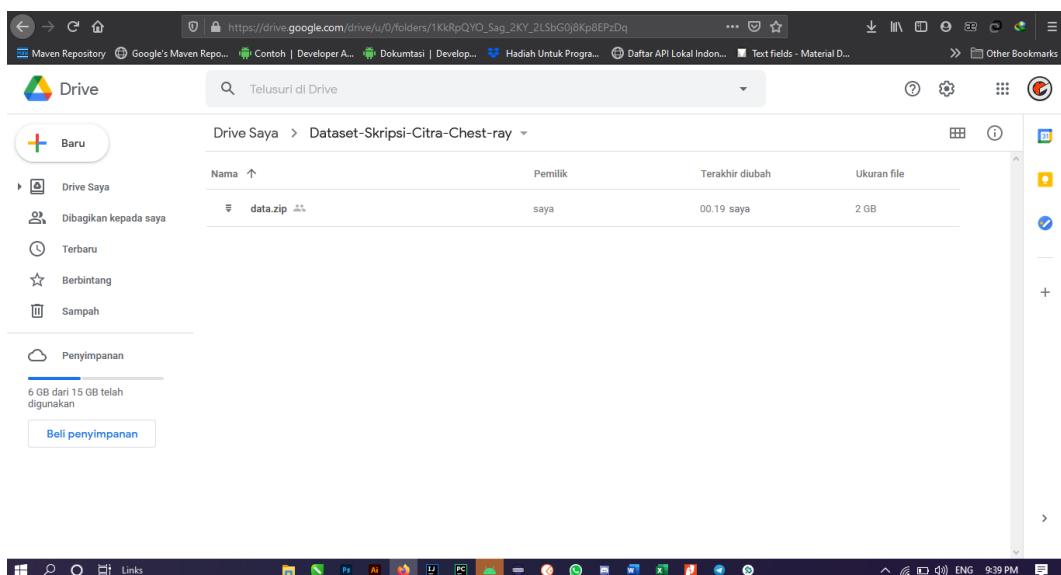
3. Jumlah *Dataset* Pengujian

Jumlah total *dataset* pengujian yaitu 1,288 gambar yang masing-masing pada *chest ray Covid-19* yaitu 116 gambar, pada *chest ray Normal* yaitu 317 gambar, dan pada *chest ray Viral Pneumonia* yaitu 855 gambar. Untuk *dataset* pengujian disimpan dalam folder *test*, terpisah dari *dataset* pelatihan dan

validasi. *Dataset* pengujian ini digunakan untuk menguji hasil pelatihan pada jaringan CNN yang digunakan.

4.3.2. Penyimpanan *Dataset*

Seluruh *dataset* pelatihan, *dataset* validasi, dan *dataset* pengujian diunggah dan disimpan pada layanan *Google Drive*. *Dataset* tersebut dikompres ke *zip file* dengan nama “data.zip” dan disimpan di dalam sebuah folder dengan nama “Dataset-Skripsi-Citra-Chest-ray”. *Dataset* pelatihan dan validasi disimpan di dalam folder “train” dan *dataset* pengujian disimpan di dalam folder “test”. Di setiap folder “train” dan “test” tersebut masing-masing terdapat 3 folder untuk menyimpan masing-masing *dataset* Covid-19, yaitu folder “COVID-19”, “NORMAL”, dan “VIRAL PNEUMONIA”. Tampilan *dataset* dapat dilihat pada gambar 4.7.

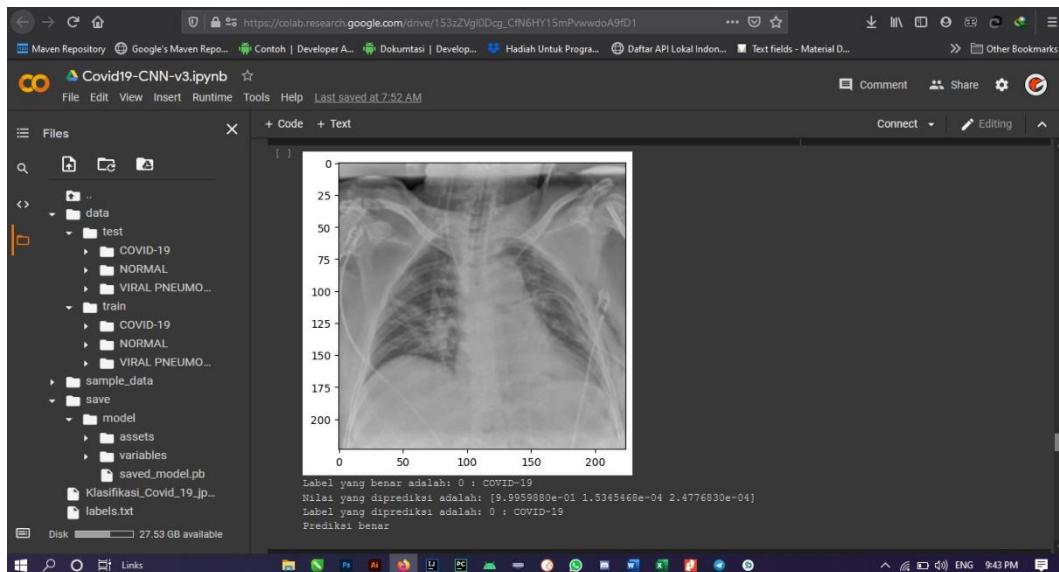


Gambar 4.12 Tampilan *dataset* covid-19 pada *google drive*

Selain untuk menyimpan *dataset*, pada *drive* ini juga digunakan untuk menyimpan program klasifikasi Covid-19 yang dibuat dengan *Google Colaboratory*.

4.4. Perancangan Program pada *Google Colaboratory*

Perancangan program pada *infrastruktur Google Colaboratory* dilakukan dengan membuat beberapa sel/baris *text* untuk memberi judul, keterangan dan penjelasan program di setiap barisnya serta membuat sel/baris kode untuk mengimpor fungsi *library* dan membuat program di setiap baris yang dibuat. Program tersebut dibuat dengan *Framework Keras* dengan *backend TensorFlow*. Berikut tampilan program klasifikasi Covid-19 pada *Google Colaboratory* dapat dilihat pada gambar 4.13.

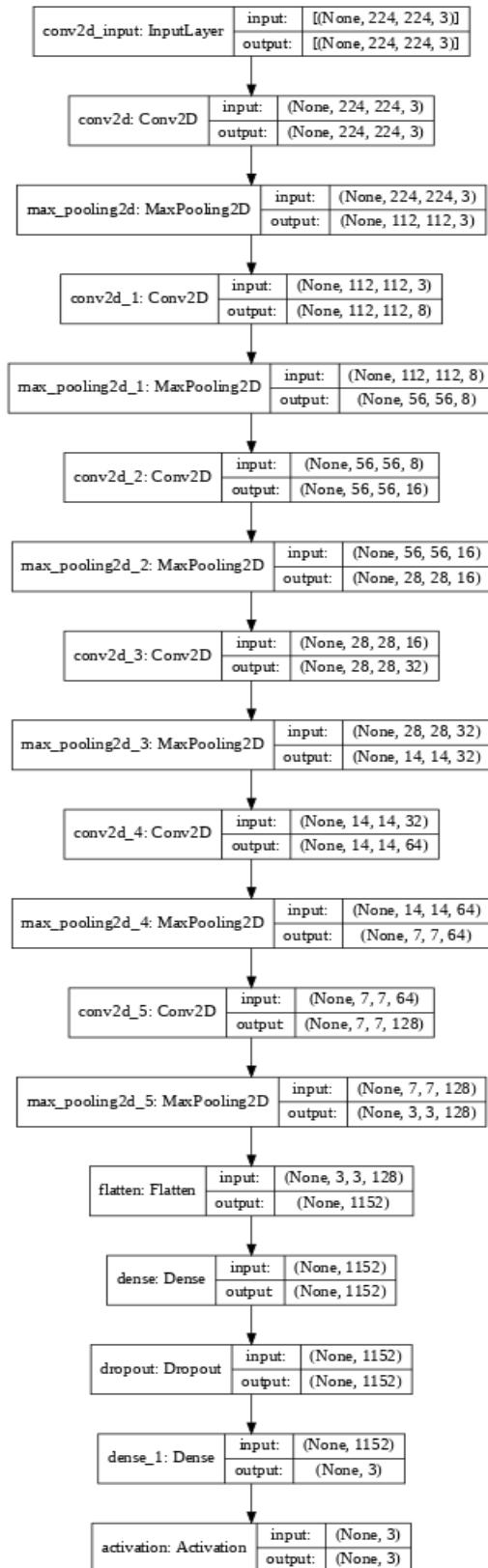


Gambar 4.13 Tampilan program klasifikasi Covid-19 pada *Google Colaboratory*

Pembuatan program yang pertama adalah membuat program untuk mengambil *dataset* dari *Google Drive*. Kemudian memasukkan/mengimpor fungsi *library* yang nantinya akan digunakan untuk menjalankan program yang

membutuhkan *library* tambahan tersebut. Selanjutnya terdapat program yang digunakan untuk mengubah ukuran 224x224 *piksel* sesuai pada masukkan dari arsitektur CNN yang akan dirancang, memecah folder train menjadi dataset pelatihan dan validasi serta memuat seluruh dataset pengujian secara urut. Setelah itu dibuat program untuk menentukan nama label citra chest ray dan menyimpannya dalam bentuk text “.txt”. Setelah itu membuat program untuk menambahkan lapisan klasifikasi Covid-19 dengan menggunakan teknik *Softmax Pooling*. Kemudian membuat program untuk melakukan pelatihan dari kedua arsitektur tersebut dengan epochs sebanyak 100 kali. Selain itu dibuat program untuk menampilkan hasil pelatihan dalam bentuk grafik tingkat akurasi dan tingkat kesalahannya. Sesudah itu terdapat program yang dibuat untuk menguji model hasil pelatihannya dengan dataset pengujian. Selanjutnya dibuat juga program untuk menampilkan hasil pengujian dalam bentuk confusion matrix yang berfungsi untuk mengetahui tingkat keberhasilan hasil pengujian dari setiap citra chest ray. Pada tahap yang terakhir terdapat program yang dibuat untuk menyimpan dan mengkonversi model hasil pelatihan ke dalam bentuk file TensorFlow Lite “.tflite”.

Pada google *Google Colaboratory* dilakukan perancangan arsitektur CNN untuk mengolah dataset citra chest ray yang sudah dikumpulkan. Perancangan arsitektur CNN terdiri dari satu input layer, enam convolusi layer, dan flatten layer. Arsitektur CNN yang dibuat dapat dilihat pada gambar 4.14.



Gambar 4.14 Rancangan Arsitektur CNN

Setelah arsitektur selesai dibuat selanjutnya melatih arsitektur dengan *dataset* yang dikumpulkan dan menghasilkan *output* berupa model yang menyimpan *Feature Maps*.

Berikut jenis-jenis gambar seperti Covid – 19, Viral Pneumonia, dan Normal :

❖ Covid - 19



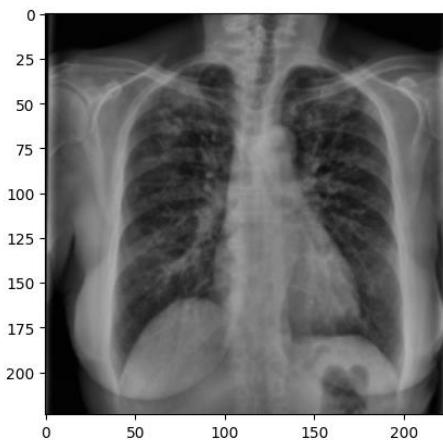
❖ Viral Pneumonia



❖ Normal

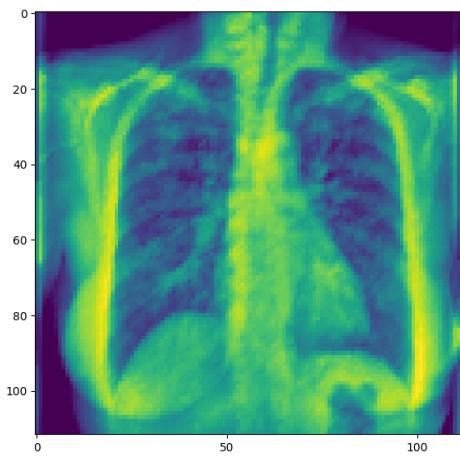


Pada proses pengenalan pola awalnya gambar input di ubah ukurannya menjadi 224 x 224 pixel, dapat dilihat pada gambar 4.15.



Gambar 4.15 Input Gambar

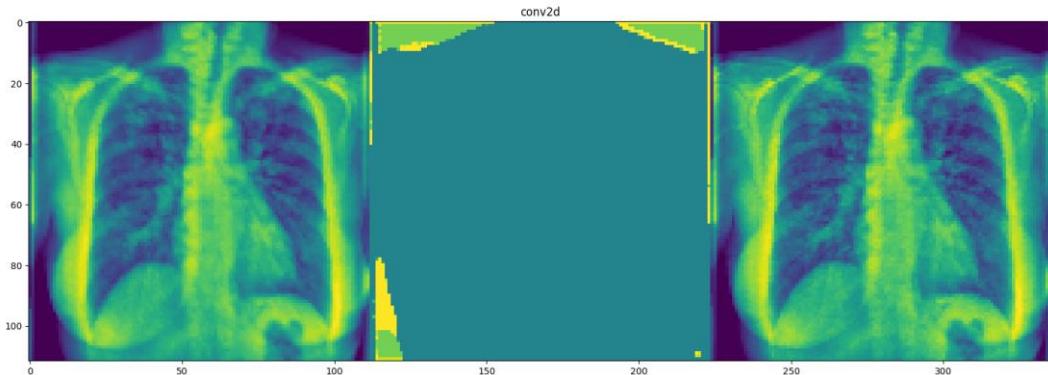
Selanjutnya gambar input diproses ke dalam model yang sudah dihasilkan oleh arsitektur CNN. pada gambar 4.16 dapat dilihat *feature maps* yang dihasilkan dari gambar input.



Gambar 4.16 Fetur Maps Input Gambar

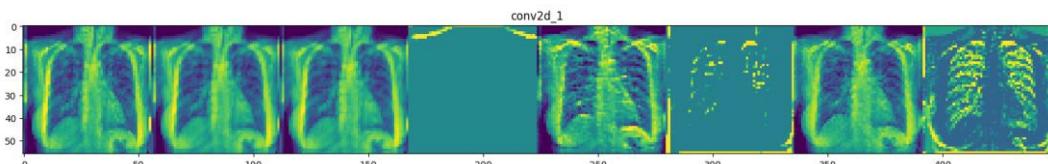
Berikut adalah proses visualisasi *Feature Maps* yang terjadi pada saat proses klasifikasi berlangsung. Seperti pada arsitektur yang dirancang gambar input

diproses di layer konvolusi pertama dengan ukuran 224 x 224 piksel dan menggunakan 3 filter. Proses konvolusi pertama dapat dilihat pada gambar 4.17.



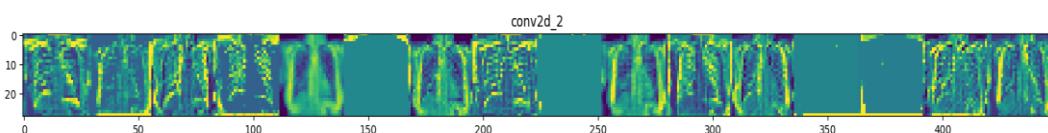
Gambar 4.17 Proses *Feature Maps* pada Lapisan Konvolusi ke-1

Selanjutnya diproses pada layer konvolusi kedua dengan ukuran 112 x 112 piksel dan menggunakan 8 filter. Proses konvolusi kedua dapat dilihat pada gambar 4.18.



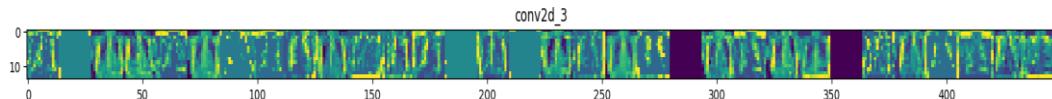
Gambar 4.18 Proses *Feature Maps* pada Lapisan Konvolusi ke-2

Selanjutnya diproses pada layer konvolusi ketiga dengan ukuran 56 x 56 piksel dan menggunakan 16 filter. Proses konvolusi ketiga dapat dilihat pada gambar 4.19.



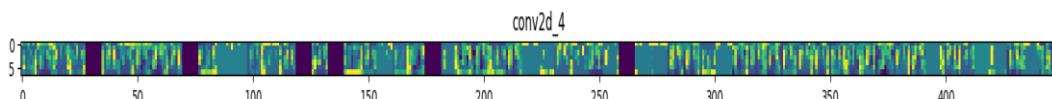
Gambar 4.19 Proses *Feature Maps* pada Lapisan Konvolusi ke-3

Selanjutnya diproses pada layer konvolusi keempat dengan ukuran 28 x 28 piksel dan menggunakan 32 filter. Proses konvolusi keempat dapat dilihat pada gambar 4.20.



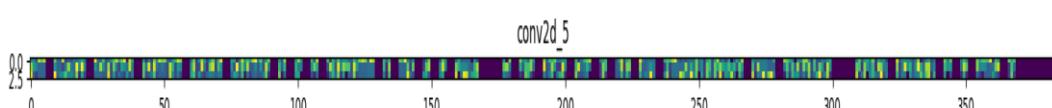
Gambar 4.20 Proses *Feature Maps* pada Lapisan Konvolusi ke-4

Selanjutnya diproses pada layer konvolusi kelima dengan ukuran 14×14 piksel dan menggunakan 64 filter. Proses konvolusi keelima dapat dilihat pada gambar 4.21.



Gambar 4.21 Proses *Feature Maps* pada Lapisan Konvolusi ke-5

Selanjutnya diproses pada layer konvolusi keenam dengan ukuran 7×7 piksel dan menggunakan 128 filter. Proses konvolusi keenam dapat dilihat pada gambar 4.22.



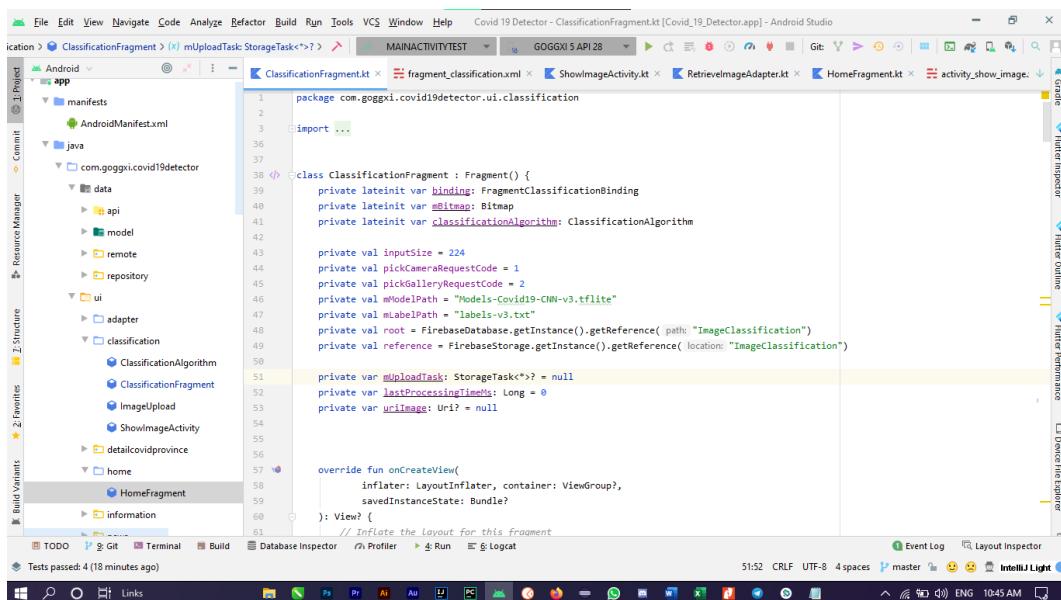
Gambar 4.22 Proses *Feature Maps* pada Lapisan Konvolusi ke-6

Dilakukan proses max pooling pada hasil konvolusi sehingga didapatkan ukuran 3×3 piksel dan 128 filter, selanjutnya di flatten dan didapatkan jumlah fitur yang akan diklasifikasi sebanyak 1152 fitur. Dapat disimpulkan bahwa semakin dalam layer konvolusinya maka semakin kompleks prosesnya dan semakin banyak jumlah fitur yang didapatkan.

4.5. Perancangan Program pada Android Studio

Perancangan program untuk perangkat Android dilakukan setelah membuat program pada Google Colaboratory dan mendapatkan hasil model pelatihannya yang disimpan dalam bentuk “.tflite”. Program dalam Bahasa Kotlin yang dibuat meliputi penggunaan fitur kamera dan galeri pada smartphone Android, mengimpor library TensorFlow Lite, membuat file class untuk klasifikasi Covid-19 yang

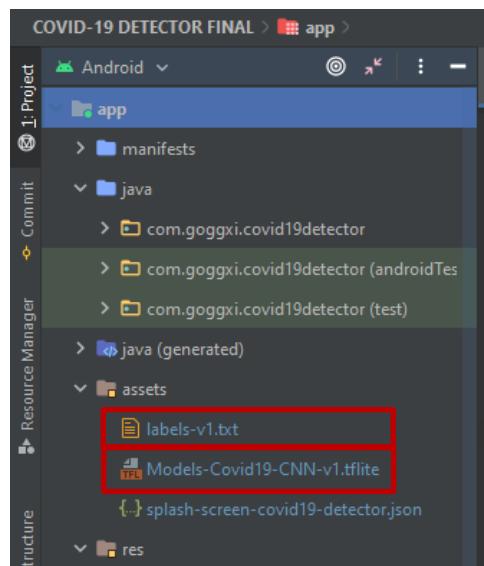
diambil dengan kamera maupun mengimpor gambar dari galeri foto, memasukkan file “.tflite” beserta labelnya dengan format “.txt” ke dalam folder assets serta membuat class lainnya untuk mendukung dalam membuat antarmuka aplikasi Android. Adapun pembuatan layout dalam format “.xml” yang berfungsi untuk membuat tampilan antarmuka pada aplikasi Android dari setiap class yang akan ditampilkan pada aplikasi Android untuk klasifikasi Covid-19. Gambar 4.4 berikut merupakan tampilan dari file project klasifikasi Covid-19 pada Android Studio IDE.



Gambar 4.23 Tampilan projek aplikasi Covid-19 Detector pada Android Studio

Sebelum membuat class untuk klasifikasi Covid-19 dengan kamera dan galeri, maka perlu dimasukkan fitur penggunaan kamera dan perijinan penggunaan kamera pada file “AndroidManifest.xml”. Selanjutnya untuk menggunakan mobile interpreter sebagai penerjemah file “.tflite” maka diharuskan mengimpor library TensorFlow Lite dengan cara memasukan implementasi “org.tensorflow:tensorflow-Lite:2.2.0” di dalam dependencies pada build.gradle. Pada pembuatan

program untuk klasifikasi Covid-19 dibuat dengan 3 file class. Kelas yang pertama digunakan untuk membuat tombol tambahkan foto dan mengambil gambar yang selanjutnya akan diekstrak dalam bentuk array agar dapat diolah oleh model CNN. Kelas selanjutnya dibuat untuk memuat file model (.tflite) dan melakukan klasifikasi Covid-19 serta memotong ukuran gambar RBG menjadi 224x224 piksel, sedangkan kelas yang ketiga untuk mengubah nilai data hasil klasifikasi dalam bentuk persentase. Setelah seluruh program dibuat maka file model CNN yang sudah dilatih dan disimpan dalam bentuk “.tflite” beserta labelnya dalam bentuk “.txt” dimasukan ke dalam folder “assets” di dalam project Android Studio.



Gambar 4.24 File model (.tflite) beserta labelnya (.txt) pada folder assets

BAB V

HASIL DAN PENGUJIAN SISTEM

5.1. Pengujian Sistem

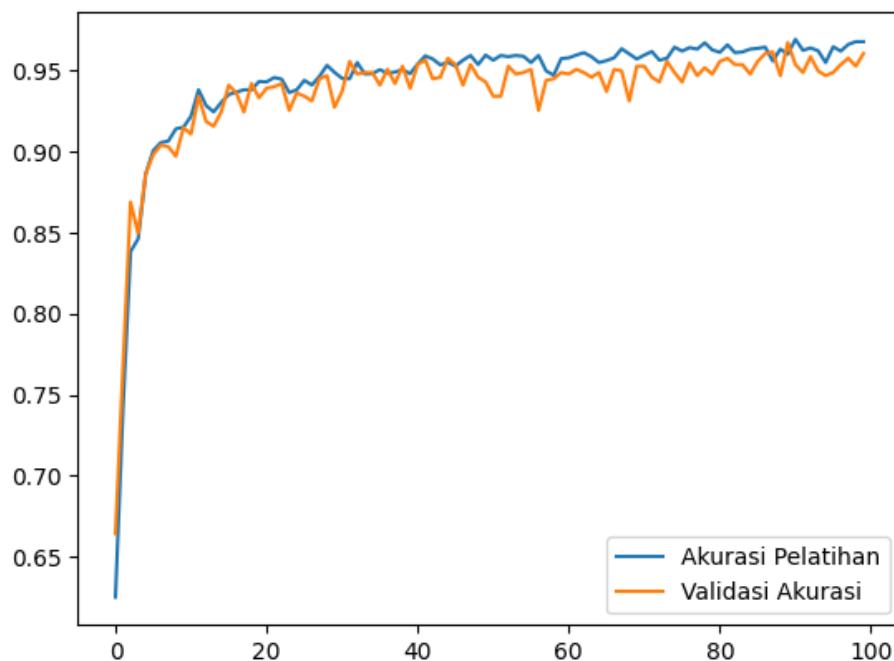
5.1.1. Hasil Pelatihan dan Pengujian pada Google Colaboratory

Pelatihan dan pengujian pada infrastruktur Google Colaboratory berfungsi untuk melatih dataset dan memperoleh modelnya serta melakukan pengujian dari hasil pelatihan tersebut. Pelatihan dan pengujian tersebut dilakukan menggunakan metode CNN. Sedangkan untuk melakukan klasifikasinya menggunakan teknik max pooling. Pelatihan dan pengujian pada Google Colaboratory ini dilakukan dengan menjalankan setiap baris program yang sudah dibuat. Pada saat pelatihan dan pengujian harus dipastikan bahwa laptop yang kita gunakan terhubung ke internet dan juga sudah terkoneksi dengan server Google Colaboratory. Sesudah terhubung ke server maka perlu mengubah pengaturan jenis runtime-nya menjadi GPU agar proses pelatihan dapat dilakukan lebih cepat. Setelah itu semua baris program dijalankan maka akan ditampilkan hasil pelatihan dan pengujianya dalam bentuk kurva tingkat akurasi dan tingkat kesalahan dari pelatihan, validasi, dan pengujianya. Selain itu juga model pada hasil pelatihan tersebut disimpan ke dalam direktori folder sementara pada Google Colaboratory yang nantinya akan diunduh dan digunakan pada aplikasi Android. Pengujian tersebut dilakukan terhadap citra chest ray dan arsitektur yang digunakan.

Pada penelitian ini pengujian yang dilakukan pada infrastruktur Google Colaboratory, yaitu dengan menggunakan dataset pelatihan dan pengujian dengan kualitas gambar yang baik menggunakan arsitektur CNN. Hal tersebut dilakukan

untuk menguji kinerja arsitektur yang digunakan apakah mampu melakukan klasifikasi Covid-19 yang dapat dilihat dari tingkat akurasi keberhasilan dalam memprediksi Covid-19. Pada pengujian tersebut dilakukan sebanyak 100 epochs.

Berikut hasil pelatihan (training) dataset pelatihan dan validasi yang dapat dilihat pada gambar 5.1.

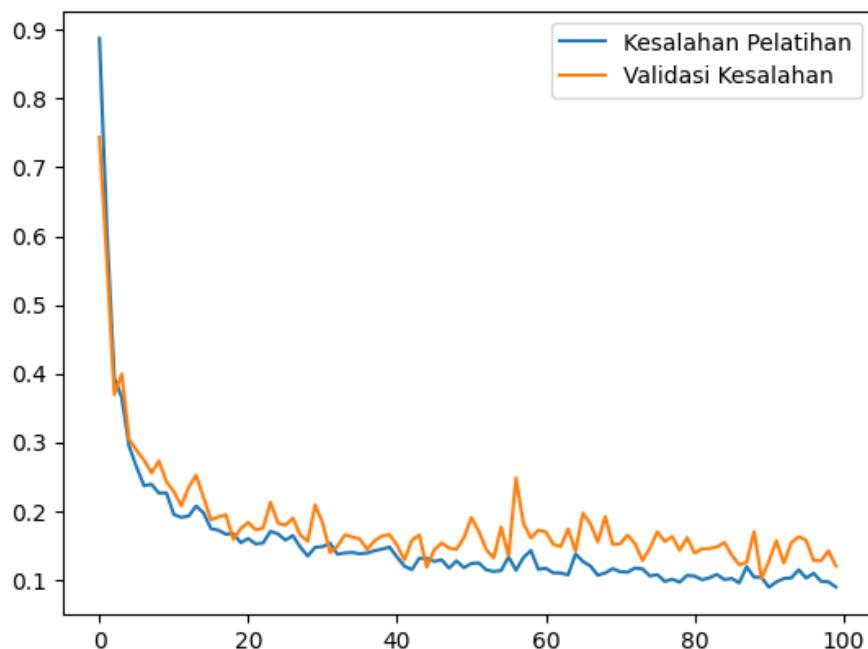


Gambar 5.1 Kurva tingkat akurasi hasil pelatihan dan validasi pada arsitektur CNN yang telah dirancang

Dari hasil pelatihan tersebut dapat dilihat bahwa kurva hasil pelatihan dan validasinya semakin meningkat mendekati nilai 1.0 seiring bertambahnya epochs (lembar kerja). Pada epoch yang terakhir nilai akurasi pelatihan mencapai 0.9671 dan nilai akurasi validasi mencapai 0.9601. Kurva akurasi pelatihan yang berwarna biru dan kurva akurasi validasi yang berwarna oranye merupakan nilai akurasi dari setiap epoch. Jika nilainya semakin mendekati 1 selama bertambahnya epoch maka proses pelatihan dan validasinya dinyatakan berhasil melakukan klasifikasi. Jika

nilai akurasi dari pelatihan dan validasinya tidak meningkat atau menurun maka jaringan CNN yang dilatih tidak dapat melakukan klasifikasi dengan baik. Dari kurva tersebut juga dapat dilihat bahwa kurva akurasi pelatihan hampir sama dengan peningkatannya dengan kurva validasi akurasi. Hal tersebut menunjukkan hasil pelatihan dengan keadaan yang goodfit.

Adapun grafik kesalahan (error) yang dihasilkan selama pelatihan seperti pada gambar 5.2 berikut.



Gambar 5.2 Kurva tingkat kesalahan pelatihan dan validasi pada arsitektur CNN yang telah dirancang

Dari hasil pelatihan tersebut dapat diketahui bahwa grafik dari kesalahan pelatihan dan validasinya semakin menurun mendekati 0.0 seiring bertambahnya epochs. Pada epoch yang terakhir nilai kesalahan pelatihan mencapai 0.0867 dan nilai kesalahan validasinya mencapai 0.1196. Kurva kesalahan pelatihan yang berwarna biru dan kurva kesalahan validasi yang berwarna oranye merupakan nilai

kesalahan yang dihasilkan dari setiap epoch. Jika nilainya semakin mendekati 0 maka jaringan CNN yang dilatih dapat melakukan klasifikasinya dengan baik dan jika nilainya tidak menurun atau bertambah maka tidak dapat melakukan klasifikasi dengan baik. Dari kurva tersebut juga dapat dilihat bahwa kurva kesalahan pelatihan hampir sama penurunannya dengan kurva validasi kesalahan sehingga dapat disimpulkan hasil pelatihannya goodfit.

Berikut adalah hasil pengujian 5 sampel yang dilakukan di google colaboratory.

```
n = 6 #Jangan melampaui (nilai dari gambar test - 1)
plt.imshow(X_test[n])
plt.show()
true_label = np.argmax(y_test2,axis=1)[n]
print("Label yang benar adalah:",true_label,":",labels[true_label])
prediction = model.predict(X_test[n][np.newaxis,...])[0]
print("Nilai yang diprediksi adalah:",prediction)
predicted_label = np.argmax(prediction)
print("Label yang diprediksi adalah:",predicted_label,":",labels[predicted_label])
if true_label == predicted_label:
    print("Prediksi benar")
else:
    print("Prediksi salah")
```



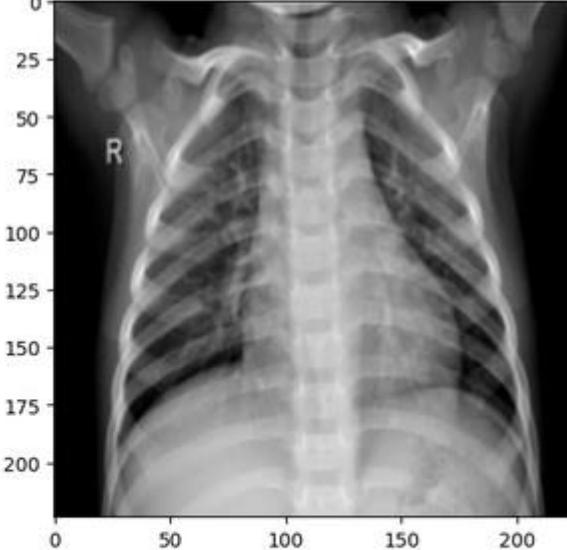
The image is a grayscale chest X-ray. A red arrow points from the bottom right to the text "Covid-19". Another red arrow points from the bottom right to the text "Normal". A third red arrow points from the bottom right to the text "Viral Pneumonia".

Label yang benar adalah: 0 : COVID-19
 Nilai yang diprediksi adalah: 9.970342e-01 7.368542e-05 2.892150e-03
 Label yang diprediksi adalah: 0 : COVID-19
 Prediksi benar

Gambar 5.3 Pengujian sample ke-1 pada arsitektur CNN di google colaboratory

Dapat dilihat pada gambar 5.3 bahwa peneliti membuat code program untuk menguji sample yang di inputkan di google colab. Pada code tersebut di atur n = 6 yaitu gambar ke-6 pada data test. Gambar tersebut adalah chest ray Covid-19 dan propabilitas dari hasil pengujian yang didapatkan adalah Covid-19 sebesar 9.97%, Normal sebesar 7.36% dan Viral Pneumonia sebesar 2.89%. Prediksi yang dihasilkan benar karena propabilitas Covid-19 lebih tinggi dibandingkan dengan citra chest ray yang lainnya.

```
n = 41 #Jangan melampaui (nilai dari gambar test - 1)
plt.imshow(X_test[n])
plt.show()
true_label = np.argmax(y_test2, axis=1)[n]
print("Label yang benar adalah:",true_label,":",labels[true_label])
prediction = model.predict(X_test[n][np.newaxis,...])[0]
print("Nilai yang diprediksi adalah:",prediction)
predicted_label = np.argmax(prediction)
print("Label yang diprediksi adalah:",predicted_label,":",labels[predicted_label])
if true_label == predicted_label:
    print("Prediksi benar")
else:
    print("Prediksi salah")
```



```
0
25
50
75
100
125
150
175
200
0 50 100 150 200
Label yang benar adalah: 1 : NORMAL
Nilai yang diprediksi adalah: [6.983022e-14 9.999769e-01 2.309567e-05]
Label yang diprediksi adalah: 1 : NORMAL
Prediksi benar
```

Gambar 5.4 Pengujian sample ke-2 pada arsitektur CNN di google colaboratory

Dapat dilihat pada gambar 5.4 bahwa pada code tersebut di atur $n = 41$ yaitu gambar ke-41 pada data test. Gambar tersebut adalah chest ray Normal dan probabilitas dari hasil pengujian yang didapatkan adalah Covid-19 sebesar 6.98%, Normal sebesar 9.99% dan Viral Pneumonia sebesar 2.30%. Prediksi yang dihasilkan benar karena probabilitas Normal lebih tinggi dibandingkan dengan citra chest ray yang lainnya.

```

n = 104 #Jangan melampaui (nilai dari gambar test - 1)
plt.imshow(X_test[n])
plt.show()
true_label = np.argmax(y_test2, axis=1)[n]
print("Label yang benar adalah:",true_label,":",labels[true_label])
prediction = model.predict(X_test[n][np.newaxis,...])[0]
print("Nilai yang diprediksi adalah:",prediction)
predicted_label = np.argmax(prediction)
print("Label yang diprediksi adalah:",predicted_label,":",labels[predicted_label])
if true_label == predicted_label:
    print("Prediksi benar")
else:
    print("Prediksi salah")

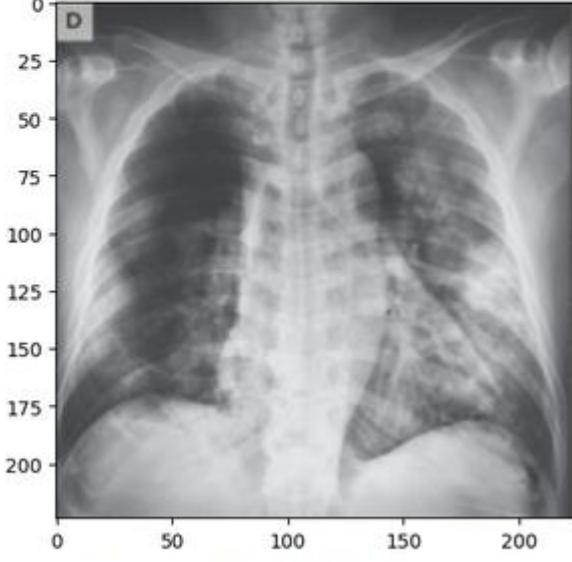
```

Label yang benar adalah: 2 : VIRAL PNEUMONIA
 Nilai yang diprediksi adalah: [6.6702425e-12 1.7064608e-03 9.9829358e-01]
 Label yang diprediksi adalah: 2 : VIRAL PNEUMONIA
 Prediksi benar

Gambar 5.5 Pengujian sample ke-3 pada arsitektur CNN di google colaboratory

Dapat dilihat pada gambar 5.5 bahwa pada code tersebut di atur n = 104 yaitu gambar ke-104 pada data test. Gambar tersebut adalah chest ray Viral Pneumonia dan propabilitas dari hasil pengujian yang didapatkan adalah Covid-19 sebesar 6.67%, Normal sebesar 1.70% dan Viral Pneumonia sebesar 9.98%. Prediksi yang dihasilkan benar karena propabilitas Viral Pneumonia lebih tinggi dibandingkan dengan citra chest ray yang lainnya.

```
n = 16 #Jangan melampaui nilai dari gambar test - 1
plt.imshow(X_test[n])
plt.show()
true_label = np.argmax(y_test2, axis=1)[n]
print("Label yang benar adalah:", true_label, ":", labels[true_label])
prediction = model.predict(X_test[n][np.newaxis, ...])[0]
print("Nilai yang diprediksi adalah:", prediction)
predicted_label = np.argmax(prediction)
print("Label yang diprediksi adalah:", predicted_label, ":", labels[predicted_label])
if true_label == predicted_label:
    print("Prediksi benar")
else:
    print("Prediksi salah")
```



0
25
50
75
100
125
150
175
200
0 50 100 150 200

Label yang benar adalah: 0 : COVID-19
Nilai yang diprediksi adalah: [9.99855876e-01 1.14811926e-04 2.92767745e-05]
Label yang diprediksi adalah: 0 : COVID-19
Prediksi benar

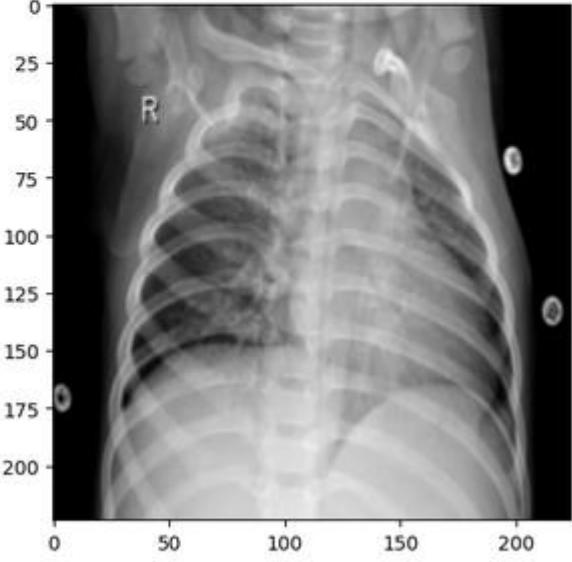
Gambar 5.6 Pengujian sample ke-4 pada arsitektur CNN di google colaboratory

Dapat dilihat pada gambar 5.6 bahwa pada code tersebut di atur $n = 16$ yaitu gambar ke-16 pada data test. Gambar tersebut adalah chest ray Covid-19 dan probabilitas dari hasil pengujian yang didapatkan adalah Covid-19 sebesar 9.99%, Normal sebesar 1.14% dan Viral Pneumonia sebesar 2.92%. Prediksi yang dihasilkan benar karena probabilitas Covid-19 lebih tinggi dibandingkan dengan citra chest ray yang lainnya.

```

n = 47 #Jangan melampaui (nilai dari gambar test - 1)
plt.imshow(X_test[n])
plt.show()
true_label = np.argmax(y_test2, axis=1)[n]
print("Label yang benar adalah:",true_label,":",labels[true_label])
prediction = model.predict(X_test[n][np.newaxis,...])[0]
print("Nilai yang diprediksi adalah:",prediction)
predicted_label = np.argmax(prediction)
print("Label yang diprediksi adalah:",predicted_label,":",labels[predicted_label])
if true_label == predicted_label:
    print("Prediksi benar")
else:
    print("Prediksi salah")

```



```

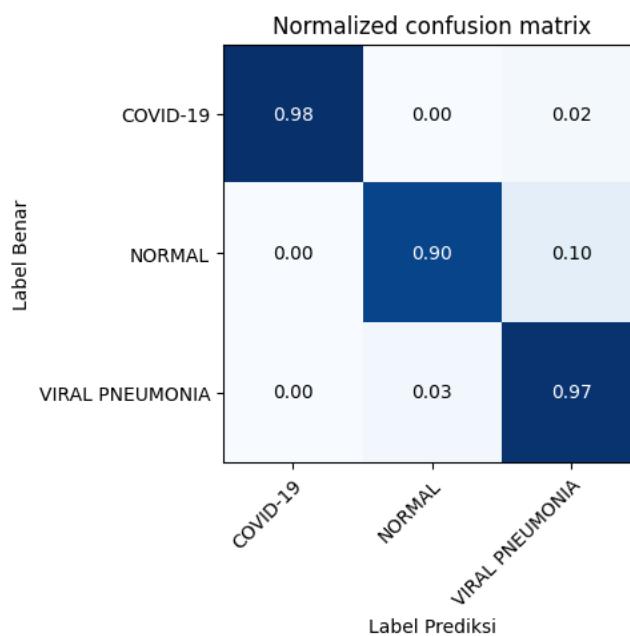
Label yang benar adalah: 1 : NORMAL
Nilai yang diprediksi adalah: [3.6018044e-10 9.4161171e-01 5.8388300e-02]
Label yang diprediksi adalah: 1 : NORMAL
Prediksi benar

```

Gambar 5.7 Pengujian sample ke-5 pada arsitektur CNN di google colaboratory

Dapat dilihat pada gambar 5.4 bahwa pada code tersebut di atur $n = 47$ yaitu gambar ke-47 pada data test. Gambar tersebut adalah chest ray Normal dan propabilitas dari hasil pengujian yang didapatkan adalah Covid-19 sebesar 3.60%, Normal sebesar 9.41% dan Viral Pneumonia sebesar 5.83%. Prediksi yang dihasilkan benar karena propabilitas Normal lebih tinggi dibandingkan dengan citra chest ray yang lainnya.

Kemudian terdapat confusion matrix yang digunakan untuk mengetahui tingkat keberhasilan dan kegagalan pada pengujian yang dilakukan. Dalam confusion matrix ini terdapat label benar dan label prediksi yang ditampilkan dengan masing-masing label citra chest ray yaitu, Covid-19, Normal, dan Viral Pneumonia. Label prediksi merupakan label hasil pengujian pada arsitektur CNN, sedangkan label benar adalah dari dataset pengujian yang sudah ditandai label di setiap citra chest ray.



Gambar 5.8 Confusion matrix hasil pengujian pada arsitektur CNN yang telah dirancang

Berdasarkan confusion matrix dari hasil pengujian tersebut dapat disimpulkan bahwa hasil prediksi dari chest ray Covid-19 bernilai 0.98 pada label benar Covid-19, 0.00 pada label benar Normal dan 0.02 pada label benar Viral Pneumonia. Dari hasil tersebut disimpulkan bahwa nilai 0.98 merupakan tingkat kebenaran hasil prediksi dari dataset pengujian yang diprediksi benar sebanyak 114 dari 116 dataset pengujian pada chest ray Covid-19. Sedangkan nilai 0.02 merupakan hasil prediksi Viral Pneumonia pada label benar Covid-19. Untuk hasil prediksi chest ray Normal bernilai 0.00 pada label benar Covid-19, 0.90 pada label benar Normal dan 0.10 pada label benar Viral Pneumonia. Dari hasil tersebut disimpulkan bahwa nilai 0.90 merupakan tingkat kebenaran hasil prediksi dari dataset pengujian yang diprediksi benar sebanyak 285 dari 317 dataset pengujian pada varietas Normal. Sedangkan nilai 0.10 merupakan hasil prediksi Viral Pneumonia pada label benar Normal. Pada hasil prediksi chest ray Viral Pneumonia bernilai 0.00 pada label benar Covid-19, 0.03 pada label benar Normal dan bernilai 0.97 pada label benar Viral Pneumonia yang dapat disimpulkan bahwa nilai 0.97 merupakan tingkat kebenaran hasil prediksi dari dataset pengujian yang diprediksi benar sebanyak 829 dari 855 dataset pengujian pada chest ray Viral Pneumonia. Sedangkan nilai 0.03 merupakan hasil prediksi Normal pada label benar Viral Pneumonia.

Dari confusion matrix tersebut juga dapat dihitung persentase tingkat akurasi dan tingkat kesalahan dari hasil prediksinya seperti berikut.

$$\% \text{akurasi} = \frac{\text{jumlah prediksi benar}}{\text{jumlah data pengujian}} \times 100\%$$

$$\% \text{akurasi} = \frac{1228}{1288} \times 100\% = 95,5\%$$

$$\% \text{kesalahan} = \frac{\text{jumlah prediksi salah}}{\text{jumlah data pengujian}} \times 100\%$$

$$\% \text{kesalahan} = \frac{60}{1288} \times 100\% = 4,5\%$$

Berdasarkan persentase akurasi dan kesalahan dapat disimpulkan bahwa arsitektur CNN yang telah dirancang mampu melakukan klasifikasi Covid-19 dengan tingkat akurasi yang tinggi dengan nilai kesalahan yang rendah.

5.1.2. Hasil Pelatihan dan Pengujian pada Perangkat Android

Pengujian pada perangkat Android dilakukan setelah melakukan pengujian pada Google Colaboratory. Pengujian ini dilakukan menggunakan model hasil pelatihan dari Google Colaboratory pada arsitektur CNN dan dataset yang digunakan. Model hasil pelatihan tersebut kemudian diimpor ke dalam folder assets pada file proyek Android Studio yang akan dibuat. Setelah diimpor dilanjutkan membuat program untuk aplikasi Android-nya pada Android Studio. Pengujian tersebut dilakukan terhadap citra chest ray dan arsitektur CNN yang digunakan.

5.1.2.1. Hasil Pelatihan dan Pengujian Model Menggunakan Kamera

Pada pengujian gambar dari kamera menggunakan 15 citra chest ray yang masing-masing 5 citra pada setiap citera chest ray nya. Untuk lebih jelasnya dapat di lihat pada tabel berikut.

Tabel 5.1 Hasil pengujian arsitektur CNN dengan sumber gambar kamera

Pengujian ke-	Respon Deteksi	Citra Ches Ray	Hasil Prediksi (Tingkat Probabilitas)	Benar / Salah
1.	91 ms	Covid-19	Covid-19 (100%)	Benar
2.	83 ms	Covid-19	Covid-19 (99.95%)	Benar
3.	34 ms	Covid-19	Covid-19 (100%)	Benar
4.	82 ms	Covid-19	Covid-19 (99.99%)	Benar
5.	32 ms	Covid-19	Covid-19 (100%)	Benar
6.	102 ms	Normal	Normal (96.52%)	Benar
7.	47 ms	Normal	Normal 86.53%)	Benar
8.	52 ms	Normal	Normal (66.20%)	Benar
9.	56 ms	Normal	Viral Pneumonia (99.99%)	Salah
10.	53 ms	Normal	Normal (94.61%)	Benar
11.	37 ms	Viral Pneumonia	Viral Pneumonia (99.87%)	Benar
12.	31 ms	Viral Pneumonia	Viral Pneumonia (89.38%)	Benar
13.	54 ms	Viral Pneumonia	Viral Pneumonia (99.97%)	Benar
14.	63 ms	Viral Pneumonia	Viral Pneumonia (86.38%)	Benar
15.	79 ms	Viral Pneumonia	Covid-19 (100%)	Salah

Dari tabel pengujian tersebut dapat dilihat bahwa hasil prediksi untuk Covid-19, Normal dan Viral Pneumonia. Tingkat probabilitas dari hasil prediksi Covid-19, Normal dan Viral Pneumonia cukup stabil di angka 90%. Setelah diperoleh tabel tersebut maka dapat dihitung presentasi akurasi dan presentasi kesalahan pengujian untuk seluruh citra chest ray sebagai berikut.

$$\% \text{akurasi} = \frac{\text{jumlah prediksi benar}}{\text{jumlah data pengujian}} \times 100\%$$

$$\% \text{akurasi} = \frac{13}{15} \times 100\% = 86\%$$

$$\% \text{kesalahan} = \frac{\text{jumlah prediksi salah}}{\text{jumlah data pengujian}} \times 100\%$$

$$\% \text{kesalahan} = \frac{2}{15} \times 100\% = 13\%$$

Dari pengujian tersebut dapat dilihat faktor cahaya, resolusi gambar dan jarak dari kamera ke obyek mepengaruhi hasil klasifikasi yang dihasilkan. Hasil perhitungan tersebut dengan tingkat akurasi yang cukup tinggi dengan nilai kesalahan yang rendah dapat disimpulkan bahwa arsitektur CNN yang telah dirancang mampu melakukan klasifikasi Covid-19 dengan baik.

5.1.2.2. Hasil Pelatihan dan Pengujian Model Menggunakan Galeri

Pada pengujian gambar dari galeri menggunakan 60 citra chest ray yang masing-masing 20 citra pada setiap citera chest ray nya. Untuk lebih jelasnya dapat dilihat pada table berikut.

Tabel 5.2 Hasil pengujian arsitektur CNN dengan sumber gambar galeri

Pengujian ke-	Respon Deteksi	Citra Chest Ray	Hasil Prediksi (Tingkat Probabilitas)	Benar/Salah
1.	127 ms	Covid-19	Covid-19 (99.99%)	Benar
2.	164 ms	Covid-19	Covid-19 (99.99%)	Benar
3.	165 ms	Covid-19	Covid-19 (99.99%)	Benar
4.	161 ms	Covid-19	Covid-19 (100%)	Benar
5.	169 ms	Covid-19	Covid-19 (99.99%)	Benar
6.	162 ms	Covid-19	Covid-19 (99.98%)	Benar
7.	184 ms	Covid-19	Covid-19 (100%)	Benar

8.	93 ms	Covid-19	Covid-19 (99.93%)	Benar
9.	151 ms	Covid-19	Covid-19 (100%)	Benar
10.	94 ms	Covid-19	Covid-19 (96.18%)	Benar
11.	116 ms	Covid-19	Covid-19 (98.82%)	Benar
12.	143 ms	Covid-19	Covid-19 (99.99%)	Benar
13.	135 ms	Covid-19	Covid-19 (99.99%)	Benar
14.	111 ms	Covid-19	Covid-19 (99.92%)	Benar
15.	113 ms	Covid-19	Covid-19 (99.99%)	Benar
16.	151 ms	Covid-19	Covid-19 (99.05%)	Benar
17.	133 ms	Covid-19	Covid-19 (99.99%)	Benar
18.	115 ms	Covid-19	Covid-19 (99.98%)	Benar
19.	103 ms	Covid-19	Covid-19 (99.99%)	Benar
20.	142 ms	Covid-19	Covid-19 (99.99%)	Benar
21.	97 ms	Normal	Normal (99.97%)	Benar
22.	96 ms	Normal	Normal (93.22%)	Benar
23.	98 ms	Normal	Normal (99.70%)	Benar
24.	113 ms	Normal	Normal (98.45%)	Benar
25.	93 ms	Normal	Normal (99.92%)	Benar
26.	75 ms	Normal	Normal (99.75%)	Benar
27.	84 ms	Normal	Normal (99.96%)	Benar
28.	74 ms	Normal	Normal (96.95%)	Benar
29.	128 ms	Normal	Normal (99.55%)	Benar

30.	207 ms	Normal	Normal (99.88%)	Benar
31.	270 ms	Normal	Normal (99.99%)	Benar
32.	74 ms	Normal	Normal (99.38%)	Benar
33.	75 ms	Normal	Normal (99.92%)	Benar
34.	74 ms	Normal	Normal (99.81%)	Benar
35.	75 ms	Normal	Normal (94.51%)	Benar
36.	122 ms	Normal	Normal (99.62%)	Benar
37.	119 ms	Normal	Normal (94.55%)	Benar
38.	206 ms	Normal	Normal (99.88%)	Benar
39.	77 ms	Normal	Normal (99.96%)	Benar
40.	74 ms	Normal	Normal (99.90%)	Benar
41.	146 ms	Viral Pneumonia	Viral Pneumonia (99.86%)	Benar
42.	76 ms	Viral Pneumonia	Viral Pneumonia (98.42%)	Benar
43.	80 ms	Viral Pneumonia	Viral Pneumonia (99.82%)	Benar
44.	122 ms	Viral Pneumonia	Viral Pneumonia (98.81%)	Benar
45.	169 ms	Viral Pneumonia	Viral Pneumonia (99.99%)	Benar
46.	136 ms	Viral Pneumonia	Viral Pneumonia (99.99%)	Benar
47.	140 ms	Viral Pneumonia	Viral Pneumonia (100%)	Benar
48.	117 ms	Viral Pneumonia	Viral Pneumonia (98.49%)	Benar
49.	154 ms	Viral Pneumonia	Viral Pneumonia (99.99%)	Benar
50.	74 ms	Viral Pneumonia	Viral Pneumonia (98.61%)	Benar
51.	75 ms	Viral Pneumonia	Viral Pneumonia (79.91%)	Benar

52.	126 ms	Viral Pneumonia	Viral Pneumonia (99.99%)	Benar
53.	67 ms	Viral Pneumonia	Viral Pneumonia (99.90%)	Benar
54.	108 ms	Viral Pneumonia	Viral Pneumonia (99.99%)	Benar
55.	75 ms	Viral Pneumonia	Viral Pneumonia (99.99%)	Benar
56.	73 ms	Viral Pneumonia	Viral Pneumonia (99.93%)	Benar
57.	142 ms	Viral Pneumonia	Viral Pneumonia (99.00%)	Benar
58.	114 ms	Viral Pneumonia	Viral Pneumonia (99.96%)	Benar
59.	75 ms	Viral Pneumonia	Viral Pneumonia (99.99%)	Benar
60.	72 ms	Viral Pneumonia	Viral Pneumonia (99.99%)	Benar

Dari tabel pengujian tersebut dapat dilihat bahwa hasil prediksi untuk COVID-19, NORMAL dan VIRAL PNEUMONIA Benar semua. Tingkat probabilitas dari hasil prediksi COVID-19, NORMAL dan VIRAL PNEUMONIA cukup stabil di angkat 95%. Setelah diperoleh tabel tersebut maka dapat dihitung presentasi akurasi dan presentasi kesalahan pengujian untuk seluruh citra chest ray sebagai berikut.

$$\% \text{akurasi} = \frac{\text{jumlah prediksi benar}}{\text{jumlah data pengujian}} \times 100\%$$

$$\% \text{akurasi} = \frac{60}{60} \times 100\% = 100\%$$

$$\% \text{kesalahan} = \frac{\text{jumlah prediksi salah}}{\text{jumlah data pengujian}} \times 100\%$$

$$\% \text{kesalahan} = \frac{0}{60} \times 100\% = 0\%$$

Dari hasil perhitungan tersebut dengan tingkat akurasi yang tinggi dengan nilai kesalahan yang rendah dapat disimpulkan bahwa arsitektur CNN yang telah dirancang mampu melakukan klasifikasi Covid-19 dengan baik.

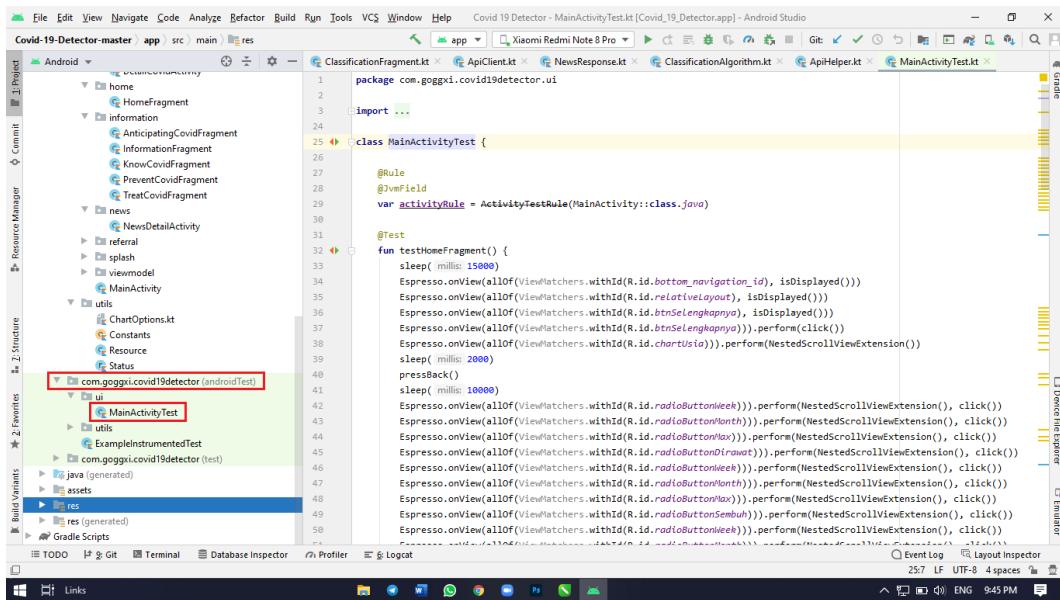
Mengapa secara offline atau pengambilan gambar menggunakan file dari galeri lebih akurat di banding secara real time atau dengan menggunakan camera ?

Karena dengan menggunakan file dari galeri tingkan resolusi gambar stabil dan jelas sehingga hasil yang di peroses mudah dikenali oleh arsitektur CNN. Sedangkan dengan menggunakan real time atau camera, apabila citra difoto kembali tingkat resolusi gambar akan berkurang sehingga sulit untuk dikenali oleh CNN tersebut. Sehingga harus memerlukan cahaya yang cukup dan pengambilan gambar harus diperhatikan agar hasil yang diproleh mudah dikenali.

5.1.2.3. Hasil Pengujian *Interface* pada Perangkat Android Menggunakan

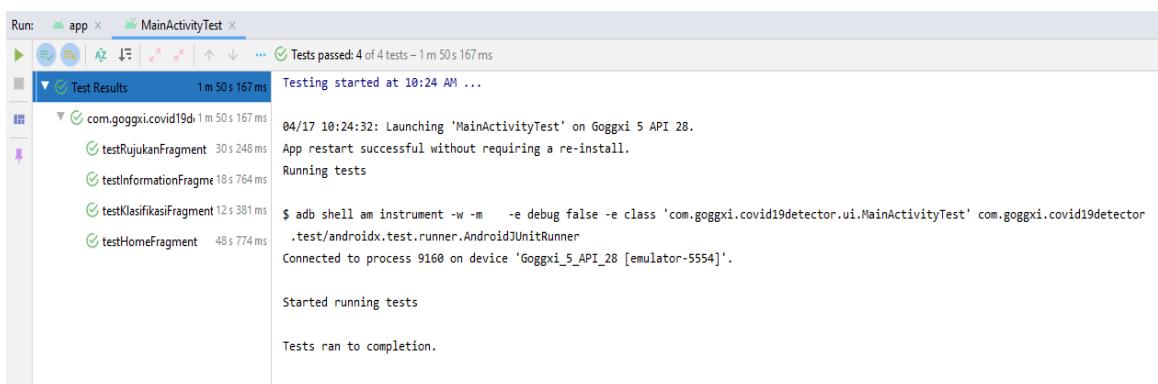
Espresso

Pengujian pada *interface* dilakukan menggunakan *Espresso* selain lebih efektif dan efisien dibanding metode *black box*, pengujian tersebut juga bertujuan untuk meminimalkan terjadinya *human error*. Sebelum menulis kode program tes pada android studio terlebih dahulu membuat skenario testing. Setelah skenarionya telah siap, *Instrumental testing* dibuat dalam folder “*androidTest*” dengan nama file “*MainActivityTest*” untuk lebih jelasnya dapat dilihat pada gambar 5.4.



Gambar 5.9 *Instrumental Testing* dengan Espresso

Pada gambar 5.4 dapat dilihat *instrumental testing* dilakukan pada file *MainActivityTest*. Pengujian tersebut menguji setiap halaman pada aplikasi yaitu *fragmentHome*, *fragmentInformasi*, *fragmentKlasifikasi*, dan *fragmentRujukan*. Setelah pengujian selesai android studio akan memberikan notifikasi tanda (x) berwarna merah jika tesnya gagal, dan memberi tanda (✓) berwarna hijau jika tesnya berhasil. Untuk hasil *instrumental test* pada aplikasi yang dibuat dapat dilihat pada gambar 5.5.



Gambar 5.10 Hasil pengujian *Instrumental Testing* menggunakan Espresso

5.2. Rekapitulasi Hasil Pengujian

Rekapitulasi hasil pengujian dan rincian skenario testing yang sudah dilakukan dapat di lihat pada tabel 5.3.

Tabel 5.3 Rekapitulasi Hasil Pengujian *Interface Menggunakan Espresso*

Pengujian yang dilakukan	Hasil
Pengujian Fragment Home <ul style="list-style-type: none"> Menampilkan informasi sebaran covid-19 Memberi aksi klik pada tombol selengkapnya Menampilkan detail informasi sebaran Covid-19 Menampilkan grafik harian Covid-19 di Indonesia Memberi aksi klik pada tombol Aktif Memberi aksi klik pada tombol Dirawat Memberi aksi klik pada tombol Sembuh Memberi aksi klik pada tombol Meninggal Memberi aksi klik pada tombol Minggu Memberi aksi klik pada tombol Bulan Memberi aksi klik pada tombol Semua Menampilkan list berita Covid-19 Memberi aksi klik pada tombol Baca Menampilkan detail berita 	Berhasil
Pengujian Fragment Informasi <ul style="list-style-type: none"> Menampilkan tombol mengenal 	Berhasil

<ul style="list-style-type: none"> • Menampilkan tombol mencegah • Menampilkan tombol mengobati • Menampilkan tombol mengantisipasi • Memberi aksi klik pada tombol mengenal • Menampilkan detail fragment mengenal • Memberi aksi klik pada tombol mencegah • Menampilkan detail fragment mencegah • Memberi aksi klik pada tombol mengobati • Menampilkan detail fragment mengobati • Memberi aksi klik pada tombol mengantisipasi • Menampilkan detail fragment mengantisipasi 	Berhasil Berhasil Berhasil Berhasil Berhasil Berhasil Berhasil Berhasil Berhasil Berhasil Berhasil Berhasil
Pengujian Fragment Klasifikasi <ul style="list-style-type: none"> • Menampilkan image view • Menampilkan tombol pilih foto • Menampilkan tombol deteksi • Memberi aksi klik pada tombol pilih foto • Memberi aksi klik pada tombol klasifikasi • Menampilkan label hasil klasifikasi • Memberi aksi klik pada tombol upload 	Berhasil Berhasil Berhasil Berhasil Berhasil Berhasil Berhasil
Pengujian Fragment Rujukan <ul style="list-style-type: none"> • Menampilkan list rumah sakit rujukan 	Berhasil

<ul style="list-style-type: none">• Menampilkan tombol panggil• Memberi aksi pada tombol panggil• Menampilkan halaman dial pad	Berhasil Berhasil Berhasil
--	----------------------------------

Pada pengujian fragmentHome terdapat 14 jumlah tes, fragmentInformasi terdapat 12 jumlah tes, fragmentKlasifikasi terdapat 7 tes, fragmentRujukan terdapat 4 tes. Total pengujian yang dilakukan adalah 37 tes dengan hasil benar 37 dan salah 0 dari hasil pengujian tersebut dapat disimpulkan bahwa aplikasi yang dibuat bekerja dengan baik.

BAB VI

KESIMPULAN DAN SARAN

6.1. Kesimpulan

Berdasarkan hasil penelitian yang telah dilakukan maka diperoleh kesimpulannya sebagai berikut.

1. Informasi sebaran, grafik harian, berita, dan rumah sakit rujukan Covid-19 di ambil dari *REST API* Pemerintahan Indonesia secara *real time*.
2. CNN dapat melakukan klasifikasi obyek gambar dengan baik jika obyek yang akan diklasifikasi sangat mirip dengan *dataset* yang sudah dilatih.
3. Proses pelatihan dan pengujian klasifikasi Covid-19 dengan CNN dilakukan pada infrastruktur *Google Colaboratory* agar menghemat waktu pelatihan serta mudah untuk mengimpor *framework Keras* dan *TensorFlow* sehingga tidak perlu memasangnya secara lokal.
4. Pengujian klasifikasi Covid-19 dengan perangkat Android dilakukan dengan cara mengambil citra *chest ray* dengan kamera dan memilih citra *chest ray* melalui galeri.
5. Hasil pengujian akan lebih maksimal jika menggunakan gambar atau file dari galeri di banding dengan menggunakan camera secara *real time*.
6. Faktor cahaya, kualitas gambar, dan jarak pengambilan gambar dapat mempengaruhi hasil klasifikasi jika menggunakan kamera.
7. Pengujian interface aplikasi menggunakan Espresso agar lebih efektif dan efisien.

6.2. Saran

Saran yang perlu dikembangkan untuk tugas akhir ini agar lebih baik sebagai berikut.

1. Dataset pelatihan ditambahkan lagi jumlah dan variasi kondisi berasnya agar hasil prediksinya semakin meningkat.
2. Dataset pengujian perlu ditambahkan lagi agar tingkat akurasi hasil pengujian tiap varietas berasnya tidak terlalu berbeda jauh.
3. Proses pendekripsi Covid-19 pada aplikasi Android dikembangkan lagi agar dapat mendekripsi secara realtime sehingga pengguna tidak perlu menekan tombol “Deteksi” pada aplikasi tersebut.

DAFTAR PUSTAKA

- Ahsan, M., Gupta, K.D., Islam, M.M. & Sen, S. 2020. COVID-19 Symptoms Detection Based on NasNetMobile with Explainable AI Using Various Imaging Modalities. 490–504.
- Andi, J. 2015. Pembangunan Aplikasi Child Tracker Berbasis Assisted – Global Positioning System (A-GPS) Dengan Platform Android. *Jurnal Ilmiah Komputer dan Informatika (KOMPUTA)*, 1(1): 1–8. Tersedia di elib.unikom.ac.id/download.php?id=300375.
- Arrofiqoh, E.N. & Harintaka, H. 2018. Implementasi Metode Convolutional Neural Network Untuk Klasifikasi Tanaman Pada Citra Resolusi Tinggi. *Geomatika*, 24(2): 61.
- Eka Putra, W.S. 2016. Klasifikasi Citra Menggunakan Convolutional Neural Network (CNN) pada Caltech 101. *Jurnal Teknik ITS*, 5(1).
- Fachriyan, M. & Dharmayanti, D. 2019. Pembangunan Aplikasi Pengenalan Objek Terdekat Untuk Penyandang Tunanetra Menggunakan Mlkit Dan Text To Voice Berbasis Android. *Elibrary Unikom*.
- Fahri, M.U. 2020. Melihat Peta Penyebaran Pasien Covid-19 Dengan Kombinasi Qgis Dan Framework Laravel. *Jurnal Teknologi Terpadu*, 6(1): 25–30.
- Google. (2020) “Android Studio User Guide”. [Daring]. Tersedia pada: <https://developer.android.com/studio/intro>. [Diakses: 31-Des-2020].
- Hasan, L.O.M., Sholeh, M. & Iswahyudi, C. 2016. Pemanfaatan Twitter Api Untuk Mengakses Jadwal Bimbingan Dosen Pada Ist Akprind Yogyakarta. *Jurnal SCRIPT*, 3(2): 139–147.
- Hassan, K., Kumar, S., Islam, T. & Rahman, M. 2020. Informatics in Medicine Unlocked COVID faster R – CNN : A novel framework to Diagnose Novel Coronavirus Disease (COVID-19) in X-Ray images. *Informatics in Medicine Unlocked*, 20: 100405. Tersedia di <https://doi.org/10.1016/j.imu.2020.100405>.
- Heriyanto, Y. 2018. Perancangan Sistem Informasi Rental Mobil Berbasis Web Pada PT.APM Rent Car. *Jurnal Intra-Tech*, 2(2): 64–77.
- Hidarlan, V.F. 2020. Rancang Bangun Klasifikasi Varietas Beras Berdasarkan Citra Menggunakan Metode Convolutional Neural Network (CNN) Berbasis

- Android. 1–17.
- Ilahiyah, S. & Nilogiri, A. 2018. Implementasi Deep Learning Pada Identifikasi Jenis Tumbuhan Berdasarkan Citra Daun Menggunakan Convolutional Neural Network. *JUSTINDO (Jurnal Sistem dan Teknologi Informasi Indonesia)*, 3(2): 49–56.
- Indrawan, K., Putra, E., Santoso, N. & Santoso, E. 2019. Pengembangan Aplikasi Pelelangan Ternak Burung Lovebird berbasis Android. 3(7): 6887–6895.
- Islam, M.T., Aowal, M.A., Minhaz, A.T. & Ashraf, K. 2017. Abnormality detection and localization in chest x-rays using deep convolutional neural networks. *arXiv*.
- KOMPAS, 2020. UPDATE: Tambah 17 Pasien, Total Ada 134 Kasus Positif Virus Korona. [online] Tersedia pada: KOMPAS.COM <https://gaya.tempo.co/read/877228/9-juta-orang-di-indonesia-mengalami-depresi/full&view=ok>. Diakses pada tanggal 17 Maret 2020.
- L. Moroney, "Using TensorFlow Lite on Android," Tensorflow Lite, 31 Maret 2018. [Daring]. Available:<https://www.tensorflow.org/lite/guide>.[Diakses:30-Des-2020].
- Li, Q., Guan, X., Wu, P., Wang, X., Zhou, L., Tong, Y., Ren, R., Leung, K.S.M., Lau, E.H.Y., Wong, J.Y., Xing, X., Xiang, N., Wu, Y., Li, C., Chen, Q., Li, D., Liu, T., Zhao, J., Liu, M., Tu, W., Chen, C., Jin, L., Yang, R., Wang, Q., Zhou, S., Wang, R., Liu, H., Luo, Y., Liu, Y., Shao, G., Li, H., Tao, Z., Yang, Y., Deng, Z., Liu, B., Ma, Z., Zhang, Y., Shi, G., Lam, T.T.Y., Wu, J.T., Gao, G.F., Cowling, B.J., Yang, B., Leung, G.M. & Feng, Z. 2020. Early Transmission Dynamics in Wuhan, China, of Novel Coronavirus–Infected Pneumonia. *New England Journal of Medicine*, 382(13): 1199–1207.
- Maha, V., Salawazo, P., Putra, D., Gea, J., Teknologi, F. & Indonesia, U.P. 2019. Implementasi Metode Convolutional Neural Network (CNN) Pada Peneganalan Objek Video Cctv. *Jurnal Mantik Penusa*, 3(1): 74–79.
- Miceli, P.A., Blair, W.D. & Brown, M.M. 2018. *Isolating Random and Bias Covariances in Tracks. 2018 21st International Conference on Information Fusion, FUSION 2018*.
- Rizal, R.A., Girsang, I.S. & Prasetyo, S.A. 2019. Klasifikasi Wajah Menggunakan Support Vector Machine (SVM). *REMIK (Riset dan E-Jurnal Manajemen Informatika Komputer)*, 3(2): 1.

- Rohim, A., Sari, Y.A. & Tibyani 2019. Convolution neural network (cnn) untuk pengklasifikasian citra makanan tradisional. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, 3(7): 7038–7042. Tersedia di <http://j-ptiik.ub.ac.id/index.php/j-ptiik/article/view/5851/2789>.
- Romzi, M. 2012. *Logika dan Algoritma*.
- Shrestha, R. & Shrestha, L. 2020. Coronavirus disease 2019 (Covid-19): A pediatric perspective. *Journal of the Nepal Medical Association*, 58(227): 525–532.
- Simarmata, S.Y.E., Sari, Y.A. & Adinugroho, S. 2019. Klasifikasi Citra Makanan Menggunakan Algoritme Learning Vector Quantization Berdasarkan Ekstraksi Fitur Color Histogram dan Gray Level Co-occurrence Matrix. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, 3(3): 2369–2378.
- Stephen, O., Sain, M., Maduh, U.J. & Jeong, D.U. 2019. An Efficient Deep Learning Approach to Pneumonia Classification in Healthcare. *Journal of Healthcare Engineering*, 2019.
- Suendri 2018. Implementasi Diagram UML (Unified Modelling Language) Pada Perancangan Sistem Informasi Remunerasi Dosen Dengan Database Oracle (Studi Kasus: UIN Sumatera Utara Medan). *Jurnal Ilmu Komputer dan Informatika*, 3(1): 1–9. Tersedia di <http://jurnal.uinsu.ac.id/index.php/algoritma/article/download/3148/1871>.
- Syahrudin, A.N. & Kurniawan, T. 2018. Input Dan Output Pada Bahasa Pemrograman Python. *Jurnal Dasar Pemrograman Python STMIK*, (January): 1–7.

RIWAYAT HIDUP



Moh Rifkan lahir di Kolonodale pada tanggal 5 Agustus 1998, merupakan mahasiswa tingkat akhir di Universitas Dipa

Makassar program studi Teknik Informatika. Tahun 2016 lulus dari SMK Negeri 1 Petasia, tahun 2013 lulus dari SMP Negeri

1 Petasia, dan tahun 2010 lulus dari SDN Inpres 2 Petasia.



Syamsul Bahri lahir di Lara pada tanggal 8 Maret 1998, merupakan mahasiswa tingkat akhir di Universitas Dipa

Makassar program studi Teknik Informatika. Tahun 2016 lulus dari SMK Mutiara Ilmu Makassar, tahun 2013 lulus dari SMP

Negeri 2 Baebunta, dan tahun 2010 lulus dari SD Negeri 045 Lara Utama.