

**DETEKSI PENYAKIT KULIT KUCING  
MENGUNAKAN ALGORITMA CNN DENGAN  
TENSORFLOW LITE BERBASIS ANDROID**

**TUGAS AKHIR**

**diajukan untuk memenuhi salah satu syarat  
memperoleh gelar sarjana  
pada Program Studi Teknik Informatika**



oleh:

**RIYANDI ADITYA FITRAH  
19416255201185**

**PROGRAM STUDI TEKNIK INFORMATIKA  
FAKULTAS ILMU KOMPUTER  
UNIVERSITAS BUANA PERJUANGAN KARAWANG  
2023**

**LEMBAR PERSETUJUAN**

**DETEKSI PENYAKIT KULIT KUCING MENGGUNAKAN ALGORITMA  
CNN DENGAN TENSORFLOW LITE BERBASIS ANDROID**

*The Disease Detection of Cat Skin Employing CNN Algorithm Using  
Tensorflow Lite Based on Android*

Tugas Akhir diajukan oleh :

**RIYANDI ADITYA FITRAH**

**19416255201185**

Program Studi Teknik Informatika

Fakultas Ilmu Komputer

Universitas Buana Perjuangan Karawang

Karawang, 31 Mei 2023

Menyetujui :

Pembimbing I,

Pembimbing II,

**Anis Fitri Nur Masruriyah, M.Kom**

NIDN: 0410049202

**Ayu Ratna Juwita, M.Kom**

NIDN: 0410069301

**LEMBAR PENGESAHAN**

**DETEKSI PENYAKIT KULIT KUCING MENGGUNAKAN ALGORITMA  
CNN DENGAN TENSORFLOW LITE BERBASIS ANDROID**

*The Disease Detection of Cat Skin Employing CNN Algorithm Using  
Tensorflow Lite Based on Android*

oleh:

**RIYANDI ADITYA FITRAH**

**19416255201185**

Tugas akhir ini telah diterima dan disahkan untuk memenuhi

sebagian syarat memperoleh gelar sarjana

pada Program Studi Teknik Informatika

Fakultas Ilmu Komputer

Universitas Buana Perjuangan Karawang

Karawang, 16 Juni 2023

Ketua Penguji,

Anggota Penguji I,

Anggota Penguji II,

**Dr. Ahmad Fauzi, M.Kom**

NIDN: 0419037701

Dekan Fakultas Ilmu Komputer,

**Jamaludin Indra, M.Kom**

NIDN: 0405058208

**Anis Fitri Nur Masruriyah, M.Kom**

NIDN: 0410049202

Koordinator Program Studi,

**Dr. Ahmad Fauzi, M.Kom**

NIDN: 0419037701

**Jamaludin Indra, M.Kom**

NIDN: 0405058208

## **LEMBAR PERNYATAAN**

Saya Riyandi Aditya Fitrah menyatakan dengan sesungguhnya bahwa Tugas Akhir yang saya tulis dengan judul “Deteksi Penyakit Kulit Kucing Menggunakan Algoritma CNN Dengan Tensorflow *Lite* Berbasis Android” beserta dengan seluruh isinya adalah merupakan hasil karya sendiri. Saya tidak melakukan penjiplakan yang tidak sesuai dengan etika keilmuan yang berlaku dalam masyarakat keilmuan.

Sesuai peraturan yang berlaku saya siap menanggung resiko/sanksi yang diberikan jika di kemudian hari ditemukan pelanggaran terhadap etika keilmuan dalam Tugas Akhir ini atau jika ada klaim dari pihak lain terhadap keaslian karya.

Karawang, 31 Maret 2022  
Yang Menyatakan,

**Riyandi Aditya Fitrah**

## **KATA PENGANTAR**

Puji syukur saya panjatkan kepada Allah SWT beserta junjungan-Nya Nabi Muhammad SAW, karena atas berkat dan rahmat-Nya, saya dapat menyelesaikan tugas akhir. Penulisan tugas akhir ini dilakukan dalam rangka memenuhi salah satu syarat untuk mencapai gelar Sarjana Komunikasi Program Studi Teknik Informatika pada Fakultas Ilmu dan Komputer, Universitas Buana Perjuangan Karawang.

Penulis mengucapkan terima kasih kepada:

1. Prof. Dr. H. Dedi Mulyadi, SE., MM, Rektor Universitas Buana Perjuangan Karawang.
2. Dr. Ahmad Fauzi, M.Kom, Dekan Fakultas Ilmu Komputer Universitas Buana Perjuangan Karawang.
3. Jamaludin Indra, M.Kom, Koordinator Program Studi Teknik Informatika Universitas Buana Perjuangan Karawang.
4. Tatang Rohana, M.Kom, Koordinator Tugas Akhir Program Studi Teknik Informatika Universitas Buana Perjuangan Karawang.
5. Anis Fitri Nur Masruriyah, M.Kom, Pembimbing I yang telah memberikan bimbingan pembuatan tugas akhir.
6. Ayu Ratna Juwita, M.Kom, Pembimbing II yang telah memberikan bimbingan tata cara menulis karya ilmiah dengan benar.
7. Drh. Fyarr F. H TA, yang telah membantu dalam usaha memperoleh data yang saya perlukan.
8. Terakhir kepada ke dua orang tua saya Bpk. Sarip Hidayatullah, Ibu. Dina Andini, sebagai sponsor hidup perkuliahan secara finansial dan konsumsi.

Semoga Tugas Akhir ini dapat bermanfaat, baik sebagai sumber informasi maupun sumber inspirasi bagi para pembaca.

Karawang, 31 Mei 2023

Penulis,

**Riyandi Aditya Fitrah**

## ABSTRAK

Penyakit kulit pada kucing dapat menimbulkan dampak negatif bagi pemilik kucing dan hewan tersebut. *Scabies* atau kudis, dan *ringworm* bersifat menular, penularan ini melalui adanya kontak langsung dengan hewan lain yang terkena kudis. Tungau telinga pada kucing bertempat di bawah kulit kucing daerah rongga telinga. Kucing yang terjangkit penyakit kulit dapat menimbulkan kerusakan pada bagian tubuhnya. Pemilik kucing juga dapat kebingungan dalam mengetahui penyakit yang diderita oleh kucingnya. Penelitian ini membantu menunjukkan penyakit kulit yang diderita oleh kucing seperti kudis, *ringworm*, dan tungau. Deteksi penyakit kulit dapat memproses pola bentuk dan ciri untuk mengetahui penyakit kulit yang diderita oleh kucing melalui aplikasi *mobile*. Aplikasi *mobile* berbasis android dibangun untuk mengetahui jenis penyakit kulit pada kucing seperti kudis, *ringworm*, dan tungau. Penelitian ini, merancang sebuah model deteksi objek menggunakan metode *deep learning* algoritma CNN dengan *tensorflow lite*. Arsitektur yang digunakan *MobileNetV2 FPNLite* dalam memproses *training* pada *dataset* yang berjumlah besar, sehingga dapat diterapkan pada aplikasi *mobile*. Berdasarkan penelitian, hasil yang didapat melalui perancangan model nilai mAP sebesar 42% dan nilai mAR 23%. Melalui evaluasi sistem dan validasi pakar diperoleh dengan nilai sebesar 73%.

**Kata Kunci:** *deep learning*, algoritma CNN, *tensorflow lite*

## ABSTRACT

*Skin diseases in cats can have a negative impact on cat owners and these animals. Scabies or mange and ringworm are contagious, this transmission is through direct contact with other animals affected by mange. Ear mites in cats are located under the cat's skin in the ear cavity area. Cats that contract skin diseases can cause damage to parts of their bodies. Cat owners can also be confused about knowing the disease their cat is suffering from. This research helps pinpoint skin diseases suffered by cats such as mange, ringworm, and mites. Skin disease detection can process shape patterns and characteristics to find out skin diseases suffered by cats through a mobile application. An Android-based mobile application was built to find out the types of skin diseases in cats such as scabies, ringworm, and mites. This study designed an object detection model using the CNN algorithm deep learning method with tensorflow lite. The architecture used by MobileNetV2 FPNLite in processing training on large datasets, so that it can be applied to mobile applications. Based on the research, the results obtained through designing a model of a mAP value of 42% and a mAR value of 23%. Through system evaluation and expert validation, a value of 73% was obtained.*

**Keyword:** *deep learning*, algorithm CNN, *tensorflow lite*

## DAFTAR ISI

<b>LEMBAR PERSETUJUAN .....</b>	<b>ii</b>
<b>LEMBAR PENGESAHAN .....</b>	<b>iii</b>
<b>LEMBAR PERNYATAAN .....</b>	<b>iv</b>
<b>KATA PENGANTAR .....</b>	<b>v</b>
<b>ABSTRAK .....</b>	<b>vi</b>
<b>DAFTAR ISI.....</b>	<b>vii</b>
<b>DAFTAR TABEL.....</b>	<b>ix</b>
<b>DAFTAR GAMBAR .....</b>	<b>x</b>
<b>DAFTAR LAMPIRAN .....</b>	<b>xii</b>
<b>BAB I PENDAHULUAN .....</b>	<b>1</b>
1.1 Latar Belakang .....	1
1.2 Rumusan Masalah .....	3
1.3 Tujuan Penelitian .....	3
1.4 Manfaat Penelitian .....	3
<b>BAB II TINJAUAN PUSTAKA.....</b>	<b>4</b>
2.1 Penyakit Kucing .....	4
2.2 <i>System Development Life Cycle</i> .....	6
2.3 Pengolahan Citra Digital .....	6
2.2.1. <i>Color Image</i> atau RGB ( <i>Red, Green, Blue</i> ) .....	8
2.2.2. <i>Black and White</i> .....	8
2.2.3. <i>Binary Image</i> .....	9
2.4 Algoritma CNN .....	9
2.5 <i>Deep Learning</i> .....	11
2.6 <i>MobileNetV2</i> .....	11
<b>BAB III METODE PENELITIAN .....</b>	<b>13</b>
3.1 Objek Penelitian.....	13
3.1.1. Lokasi dan Waktu Penelitian.....	13
3.1.2. Peralatan Penelitian.....	13
3.2 Prosedur Penelitian.....	15
3.2.1 Analisis Masalah .....	15
3.2.2 Pengumpulan Data .....	15
3.2.3 Perancangan .....	16
3.2.4 Implementasi .....	25

3.2.5 Pengujian.....	26
3.3 Evaluasi Hasil.....	27
<b>BAB IV HASIL DAN PEMBAHASAN.....</b>	<b>30</b>
4.1 Perancangan Model.....	30
4.1.1 Akuisisi Citra .....	30
4.1.2 <i>Preprocessing</i> Citra.....	31
4.1.3 Membuat <i>Labelmap</i> .....	33
4.1.4 Membuat <i>TF Record</i> .....	34
4.1.5 Konfigurasi <i>Pipeline</i> .....	34
4.1.6 <i>Training</i> Model Deteksi Objek .....	35
4.2 Desain Sistem Dan Aplikasi.....	37
4.2.1 <i>Use Case</i> Diagram.....	37
4.2.2 Hasil Desain .....	38
4.3 Implementasi Model.....	39
4.4 <i>Whitebox Testing</i> .....	41
4.4.1 Fungsi Mendeteksi Objek Penyakit Kulit Kucing.....	41
4.5 <i>Blackbox Testing</i> .....	43
4.6 Evaluasi Hasil.....	44
<b>BAB V KESIMPULAN DAN SARAN .....</b>	<b>46</b>
5.1 Kesimpulan .....	46
5.2 Saran.....	46
<b>DAFTAR PUSTAKA .....</b>	<b>47</b>
<b>LAMPIRAN.....</b>	<b>50</b>
<b>RIWAYAT PENULIS.....</b>	<b>91</b>



## DAFTAR TABEL

Tabel 3. 1 Tabel Paparan Penelitian.....	13
Tabel 4. 1 Tabel Percobaan Pelatihan Model.....	35
Tabel 4. 2 Kode Fungsi Deteksi Objek Penyakit Kulit Kucing .....	41
Tabel 4. 3 Tabel Pengujian <i>Blackbox Testing</i> .....	43
Tabel 4. 4 Tabel Validasi Pakar .....	44

## DAFTAR GAMBAR

Gambar 2. 1 Kucing Terjangkit Kudis .....	4
Gambar 2. 2 Tungau Telinga Pada Kucing .....	5
Gambar 2. 3 Kucing Terjangkit Kurap .....	5
Gambar 2. 4 Representasi Citra Digital Dalam Dua Dimensi .....	7
Gambar 2. 5 Contoh Color Image Citra .....	8
Gambar 2. 6 Arsitektur Algoritma CNN .....	10
Gambar 2. 7 Arsitektur <i>MobileNetV2</i> .....	12
Gambar 3. 1 Prosedur Penelitian .....	15
Gambar 3. 2 Alur Perancangan Model .....	16
Gambar 3. 3 Alur Arsitektur <i>MobileNetV2</i> .....	19
Gambar 3. 4 Prosedur Ekstraksi Ciri Citra Penyakit Kulit Kucing .....	22
Gambar 3. 5 <i>Flowchart</i> Alur Sistem .....	25
Gambar 3. 6 Ilustrasi Perhitungan IoU .....	28
Gambar 4. 1 Struktur Folder <i>Images</i> .....	30
Gambar 4. 2 Isi Folder <i>Test</i> dan <i>Train</i> .....	30
Gambar 4. 3 Membuka Direktori Folder <i>Test</i> dan <i>Train</i> .....	31
Gambar 4. 4 Pelabelan Citra .....	32
Gambar 4. 5 Menyimpan Hasil Pelabelan .....	32
Gambar 4. 6 Isi File XML .....	33
Gambar 4. 7 Isi File Labelmap .....	33
Gambar 4. 8 File <i>Train Record</i> .....	34
Gambar 4. 9 File <i>Test Record</i> .....	34
Gambar 4. 10 Grafik Total <i>Loss</i> Berdasarkan Jumlah <i>Step</i> .....	36
Gambar 4. 11 Hasil Berkas Pelatihan Model .....	36
Gambar 4. 12 Hasil Berkas Expor Model .....	37
Gambar 4. 13 Hasil Konversi Model Deteksi Tensorflow <i>Lite</i> .....	37
Gambar 4. 14 <i>Use Case Diagram</i> .....	37
Gambar 4. 15 Hasil Desain Tampilan Menu Utama .....	38
Gambar 4. 16 Hasil Desain Tampilan Kamera .....	38
Gambar 4. 17 Hasil Desain Tampilan Detail .....	39
Gambar 4. 18 (A) Deteksi Tungau (B) Deteksi Kudis (C) Deteksi <i>Ringworm</i> .....	40

Gambar 4. 19 Tampilah Hasil Halaman Detail .....	40
Gambar 4. 20 <i>Flow Graph</i> .....	42

## DAFTAR LAMPIRAN

Lampiran 1 Form Bimbingan Proposal Tugas Akhir.....	50
Lampiran 2 Lembar Perbaikan Penguji.....	51
Lampiran 3 Dokumentasi Wawancara Pertama.....	53
Lampiran 4 Konfigurasi <i>Pipeline</i> .....	54
Lampiran 5 Dokumentasi Wawancara Kedua .....	55
Lampiran 6 Evaluasi Sistem.....	56
Lampiran 7 Desain UX Aplikasi Deteksi.....	64
Lampiran 8 Perintah dan Program Model Tensorflow 2.5.0.....	65
Lampiran 9 Perintah dan Program Model Tensorflow 2.8.0.....	68
Lampiran 10 Kode Program Implementasi Model.....	71
Lampiran 11 Lembar Bimbingan Tugas Akhir .....	90

# BAB I

## PENDAHULUAN

### 1.1 Latar Belakang

Berdasarkan Nurajizah dan Saputra (2018) kucing memiliki peran sebagai sahabat bagi pemiliknya. Kebiasaan kucing yang dipelihara oleh manusia selalu berada di rumah dan terkadang juga diluar rumah. Kebiasaan ini tidak menutup kemungkinan kucing dapat terjangkit suatu penyakit yang bisa dilihat secara kasat mata pada tubuh kucing (Nurajizah & Saputra, 2018). Memelihara kucing tidak hanya dari asupan makan, tetapi juga perlu diperhatikan kesehatannya (Sudirman *et al.*, 2022). Pemilik kucing bisa menjadi kebingungan dalam mengetahui penyakit yang diderita oleh kucingnya (Nurajizah & Saputra, 2018). Hal ini dikarenakan kurangnya pengetahuan pemilik kucing tentang penyakit hewan tersebut (Sudirman *et al.*, 2022). Bila hewan peliharaan tersebut menderita penyakit, tentunya bisa berdampak negatif pada hewan peliharaan nya dan juga pemilik binatang (Patria *et al.*, 2021). Ketidaktahuan seorang pemilik kucing tentang informasi jenis penyakit kucing yang dilihat secara kasat mata, serta sulitnya menemukan seorang pakar dalam keadaan mendesak, dan mahal biayanya juga menjadi alasan pemilik hewan tidak membawa peliharaan nya ke dokter hewan (Nurajizah & Saputra, 2018).

Pada penelitian Harjianto dan Latif (2016) membuat penelitian diagnosa penyakit pada kucing dengan metode teorema bayes berbasis android. Penelitian mengolah data penyakit kucing dan data gejala-gejala penyakit kucing yang diperoleh dari *studi literatur* dan para ahli menggunakan metode Teorema Bayes. Berdasarkan pada pengujian, 15 sampel data gejala penyakit menunjukkan bahwa program aplikasi menghasilkan nilai akurasi sebesar 90 %.

Penelitian lainnya membuat sistem pakar diagnosa penyakit kucing menggunakan metode *Case-Based Reasoning* (CBR) (Fidyaningsih *et al.*, 2017). Penelitian ini mengelola data gejala penyakit yang diderita kucing menggunakan algoritma *K-Nearest Neighbor* (KNN) dengan metode *Case-Based Reasoning* (CBR). Dari hasil penelitian, sistem pakar diagnosa penyakit kucing menggunakan metode *Cased-Based Reasoning* (CBR) mempunyai tingkat akurasi sebesar 90%.

Penelitian yang dilakukan oleh Dwiramadhan *et al.* (2022) tentang diagnosa penyakit kulit pada kucing menggunakan metode *Naïve Bayes* menghasilkan sistem

berbasis web. Penelitian tersebut mengolah data gejala-gejala penyakit kulit pada kucing menggunakan algoritma *Naïve Bayes*. Hasil uji yang diperoleh berupa membandingkan hasil diagnosa sistem yang sama dengan diagnosa dokter. Dari pengujian 15 data rekam medis dokter didapat tingkat akurasi sistem pakar diagnosa penyakit kulit pada kucing sebesar 80%.

Dalam penelitian Setyawan *et al.* (2021) yang melakukan penelitian mengenai sistem pakar diagnosa penyakit kucing dengan *Naïve Bayes*. Penelitian ini mengelola data penyakit kucing dari seorang pakar ke dalam suatu sistem beserta data gejala penyebab penyakit kucing menggunakan algoritma *Naïve Bayes*. Hasil pengujian terhadap data uji dengan jumlah penyakit sebanyak 8 dan data gejala sebanyak 19 menghasilkan tingkat keakuratan sistem sebesar 87,5 %.

Penelitian oleh Widyaningsih dan Gunadi (2017) tentang sistem diagnosa gejala penyakit kulit pada kucing. Data yang dikelola berupa gejala penyakit kulit kucing menggunakan metode *Dempster Shafer*. Hasil diagnosa sistem perhitungan *Dempster Shafer* mencapai 99% untuk mengetahui jenis penyakit kulit pada kucing.

Berdasarkan masalah terkait penyakit kucing dan beberapa uraian solusi pada penelitian yang telah dilakukan, penelitian ini membuat sistem untuk mendeteksi penyakit kulit yang diderita oleh kucing. Penelitian ini membangun aplikasi berbasis android yang belum dilakukan oleh penelitian terdahulu, dengan memanfaatkan *framework* tensorflow menggunakan metode *deep learning* dengan algoritma CNN arsitektur *MobileNet*. Pengguna sistem operasi android juga telah banyak digunakan karena bersifat *user friendly* atau mudah dijalankan (Ishak & Pakaya, 2021). Algoritma CNN untuk melakukan klasifikasi data berlabel yang terdapat data latih. Arsitektur *MobileNetV2* cocok digunakan untuk *training* pada *dataset* yang berjumlah besar, sehingga dapat diterapkan pada aplikasi *mobile* (Dharmaputra *et al.*, 2021). Sistem ini memudahkan pemilik kucing agar dapat memahami, mengetahui, dan mengerti jenis penyakit pada kucing yang dipelihara. Pembuatan sistem ini juga, memudahkan pemilik kucing dalam menangani hewannya yang terjangkit penyakit kulit dengan memberikan solusi yang dilakukan pada pemilik kucing seperti obat yang dibutuhkan, pencegahan atau penanganan penyakit hewan tersebut, serta mengetahui penyebab gejala penyakit kulit kucing yang ditimbulkan berdasarkan hasil input citra yang dilakukan pemilik kucing.

## 1.2 Rumusan Masalah

Berdasarkan latar belakang masalah, maka rumusan masalah dalam penelitian ini sebagai berikut :

1. Bagaimana model pendeteksian objek penyakit kucing menggunakan algoritma CNN dan metode *deep learning*?
2. Bagaimana membuat aplikasi *mobile* berbasis android yang menerapkan model deteksi objek penyakit kulit kucing dengan tensorflow *lite* menggunakan algoritma CNN dan metode *deep learning*?
3. Bagaimana hasil evaluasi model deteksi objek penyakit kulit kucing menggunakan metode *deep learning* pada aplikasi android?

## 1.3 Tujuan Penelitian

Tujuan penelitian sebagai berikut :

1. Mengetahui model pendeteksian objek penyakit kucing menggunakan algoritma CNN dengan metode *deep learning*.
2. Membuat aplikasi *mobile* yang menerapkan model deteksi objek penyakit kulit kucing dengan tensorflow *lite* menggunakan algoritma CNN dan metode *deep learning*.
3. Mengetahui hasil evaluasi model deteksi objek penyakit kulit kucing pada aplikasi android.

## 1.4 Manfaat Penelitian

1. Menambah pengetahuan terkait model pendeteksian objek penyakit kucing menggunakan algoritma CNN dengan metode *deep learning*.
2. Membuat model pendeteksian objek penyakit kucing menggunakan metode *deep learning*, diharapkan dapat membuat perkembangan permasalahan *computer vision* lainnya.
3. Hasil penelitian dapat dijadikan acuan untuk pendeteksian objek menggunakan algoritma CNN dengan metode *deep learning*.

## BAB II

### TINJAUAN PUSTAKA

#### 2.1 Penyakit Kucing

Kulit merupakan organ tubuh kucing bagian luar sebagai pembatas (Widyaningsih & Gunadi, 2017). Beragam jenis kucing yang dipelihara manusia, salah satunya yaitu kucing ras seperti (Anggora, dan Persia) serta kucing dom (kucing kampung) (Dwiramadhan *et al.*, 2022). Memelihara kucing harus juga dengan memelihara kesehatannya agar terhindar dari berbagai macam penyakit kucing terutama penyakit kulit pada kucing. Secara umum, terdapat beberapa jenis penyakit kulit pada kucing seperti *scabies* atau kudis, tungau telinga, dan *ringworm* atau kurap.

*Scabies* merupakan salah satu penyakit kulit pada kucing yang sering dijumpai (Susanto *et al.*, 2020). Hal ini disebabkan oleh tungau (*sarcoptes scabiei*) yang ditandai gejala gatal pada bagian kulit kucing (Susanto *et al.*, 2020). Apabila kucing terjangkit, dapat menurunkan produksi daging, kualitas kulit, dan mengganggu Kesehatan (Susanto *et al.*, 2020). kucing yang terjangkit penyakit kulit kudis ditunjukkan pada Gambar 2.1.



Gambar 2. 1 Kucing Terjangkit Kudis  
(Sumber : Observasi)

Tungau telinga (*otodectes cynotis*) merupakan tungau bertempat di bawah kulit kucing daerah rongga telinga (Putri, 2022). Tungau telinga pada kucing dapat menimbulkan luka serta radang pada bagian sarangnya (Putri, 2022). Gejala timbul



tungau telinga melalui tanda bintik-bintik kecil kemerahan pada kulit berbentuk garis-garis (Putri, 2022). Gambar 2.2 merupakan kucing yang terjangkit tungau telinga.



Gambar 2. 2 Tungau Telinga Pada Kucing  
(Sumber : Hewania (2022))

Kurap merupakan salah satu penyakit kulit pada kucing yang bersifat menular (Nasyuha *et al.*, 2022). Kurap disebabkan oleh jamur *keratinofilik* pada permukaan kulit (Nasyuha *et al.*, 2022). Bagian lainnya yang mengandung keratin (bulu, kuku, rambut, dan tanduk) pada hewan dan manusia (Nasyuha *et al.*, 2022). Penyebab kurap terjadi karena cacing dan gejala yang diawali dengan peradangan melalui permukaan kulit (Nasyuha *et al.*, 2022). Jika dibiarkan akan membesar dan membentuk lingkaran seperti cincin (Nasyuha *et al.*, 2022). Terdapat juga kucing yang terjangkit kurap ditunjukkan Gambar 2.3.



Gambar 2. 3 Kucing Terjangkit Kurap  
(Sumber : Fauzi (2022))

## 2.2 System Development Life Cycle

*System Development Life Cycle* (SDLC) merupakan metode pengembangan yang memelihara, dan menghasilkan sistem yang standar (Permana & Romadlon, 2019). Penelitian ini menggunakan metode SDLC model *waterfall*. Menurut Sommerville, mendefinisikan model air *waterfall* sebagai tahapan terpenting pembangunan suatu sistem informasi (How, 2021). Penjelasan tiap tahapan perancangan menggunakan metode *waterfall* menurut Sommerville.

### 1. *Requirement Definition* (Analisis Kebutuhan Perangkat Lunak)

Tahapan analisis ini sebelum perancangan, yaitu pengumpulan beberapa kebutuhan untuk membuat perangkat lunak (How, 2021). Melakukan analisis sifat perangkat lunak yang dibuat sampai dengan antarmukanya (How, 2021).

### 2. *System and Software Design* (Desain)

Apabila analisa sudah dilakukan, selanjutnya membangun desain perangkat lunak (How, 2021). Proses tahapan ini sesuai dengan kebutuhan yang sudah dianalisis sebelum melakukan penerapan kode (How, 2021).

### 3. *Implementation and Unit Testing* (Kode)

Setelah membangun desain, kemudian menerapkannya ke dalam sebuah kode-kode program yang dipilih pengembang (How, 2021).

### 4. *Integration and System Testing*

Lalu pengujian dilakukan untuk memastikan kesalahan yang dibuat, dan hasil perangkat lunak sudah sesuai yang diinginkan (How, 2021).

### 5. *Operation and Maintenance*

Setelah pengujian, pengembang perlu melakukan pemantauan dan jika perlu meningkatkan layanan sistem yang dikelola (How, 2021).

## 2.3 Pengolahan Citra Digital

Pengolahan citra digital atau *digital image processing* merupakan pengetahuan yang mempelajari pengelolaan terkait teknik dalam mengelola citra (Munantri *et al.*, 2020). Pada penelitian ini, citra yang dimaksud sebuah gambar atau foto dan juga gambar bergerak yang berasal dari (Images, 2022). Secara matematika, citra memiliki fungsi kontinu (*continue*) dengan intensitas cahaya terhadap bidang dua dimensi (Munantri *et al.*, 2020).

Dalam pengolahan citra digital, mewakili sebuah matriks dua dimensi  $f(x,y)$  yang terdiri dari  $M$  kolom dan  $N$  baris (Nurhikmat, 2020). Perpotongan antara kolom dan baris disebut piksel (*pixel = picture element*) (Munantri *et al.*, 2020).

Representasi dari citra digital dapat dilihat melalui persamaan 1.

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \dots & f(0,N-1) \\ f(1,0) & f(1,1) & \dots & f(1,N-1) \\ \vdots & \vdots & \ddots & \vdots \\ f(M-1,0) & f(M-1,1) & \dots & f(M-1,N-1) \end{bmatrix} \quad (1)$$

Suatu citra pada  $f(x,y)$  dalam fungsi matematika ditulis melalui persamaan 2.

$$0 \leq x \leq M-1 \quad (2)$$

$$0 \leq y \leq N-1$$

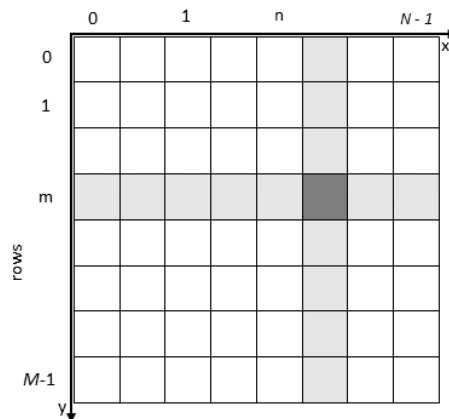
$$0 \leq f(x,y) \leq G-1$$

Dimana :  $M$  = jumlah piksel baris (*row*) pada *array* citra

$N$  = jumlah piksel kolom (*column*) pada *array* citra

$G$  = nilai skala keabuan (*graylevel*)

Representasi nilai fungsi kontinyu berupa nilai-nilai diskrit yang disebut digitalisasi citra seperti ditunjukkan pada Gambar 2.4.



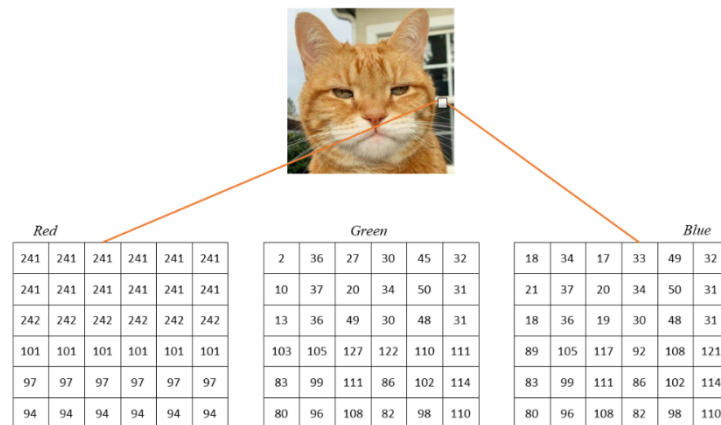
Gambar 2. 4 Representasi Citra Digital Dalam Dua Dimensi  
(Sumber : Nurhikmat (2020))

Melalui representasi nilai citra, objek tertentu dapat terdeteksi menggunakan pengolahan citra digital ini. Metode yang digunakan salah satunya adalah

segmentasi warna. Kelebihan dari segmentasi warna yaitu proses cepat dan efektif pada objek, maupun aplikasi untuk *custom* deteksi objek (Nurhikmat, 2020). Adapun jenis pengolahan citra digital secara umum diantaranya *color image*, *black and white*, dan *binary image*.

### 2.2.1. Color Image atau RGB (Red, Green, Blue)

Pada setiap citra, tentunya memiliki masing-masing piksel dan warna (Nurhikmat, 2020). Warna tersebut mencakup RGB (*Red, Green, Blue*) (Nurhikmat, 2020). Tiap-tiap warna pada citra memiliki *range* dari 0 hingga 255 (Nurhikmat, 2020). *Color image* memiliki tiga matriks yang mewakili nilai-nilai merah, hijau, dan biru untuk setiap pikselnya (Nurhikmat, 2020). Gambar 2.5 merupakan contoh *color image* citra.



Gambar 2. 5 Contoh *Color Image* Citra  
(Sumber : Nurhikmat (2020))

### 2.2.2. Black and White

Pengolahan citra digital *black and white* (*grayscale*) memiliki piksel dengan gradasi warna mulai dari putih sampai hitam (Nurhikmat, 2020). Rentang warna citra *black and white* cocok digunakan untuk pengolahan file gambar (Nurhikmat, 2020). *Black and white* merupakan hasil rata-rata dari *color image* (Nurhikmat, 2020). Menentukan hasil rata-rata dapat ditulis melalui persamaan 3.

$$I_{BW}(x, y) = \frac{I_R(x, y) + I_G(x, y) + I_B(x, y)}{3} \quad (3)$$

Dimana :  $I_R(x, y)$  = nilai piksel *Red* titik  $(x, y)$

$I_G(x, y)$  = nilai piksel *Green* titik  $(x, y)$

$I_R(x, y)$  = nilai piksel *Blue* titik  $(x, y)$

$I_{BW}(x, y)$  = nilai piksel *black and white* titik  $(x, y)$

### 2.2.3. Binary Image

Pada sebuah citra, memiliki piksel yang terdiri dari warna hitam atau putih, karena hanya terdapat dua warna saja untuk tiap piksel (Nurhikmat, 2020). Citra yang direpresentasikan dengan biner, memiliki kecocokan untuk teks dan gambar arsitektur (Nurhikmat, 2020). *Binary image* merupakan pengolahan citra dari *black and white* dengan fungsi yang ditulis melalui persamaan 4.

$$I_{Bin}(x, y) = \begin{cases} 0 & I_{BW}(x, y) < T \\ 255 & I_{BW}(x, y) \geq T \end{cases} \quad (4)$$

Sementara jika bentuk *floating point* dapat ditulis melalui persamaan 5.

$$I_{Bin}(x, y) = \begin{cases} 0 & I_{BW}(x, y) < T \\ 1 & I_{BW}(x, y) \geq T \end{cases} \quad (5)$$

Dimana :  $I_{BW}(x, y)$  = nilai piksel *Gray* titik  $(x, y)$

$I_G(x, y)$  = nilai piksel *Bimary* titik  $(x, y)$

$I_{Bin}(x, y)$  = nilai piksel *Binary* titik  $(x, y)$

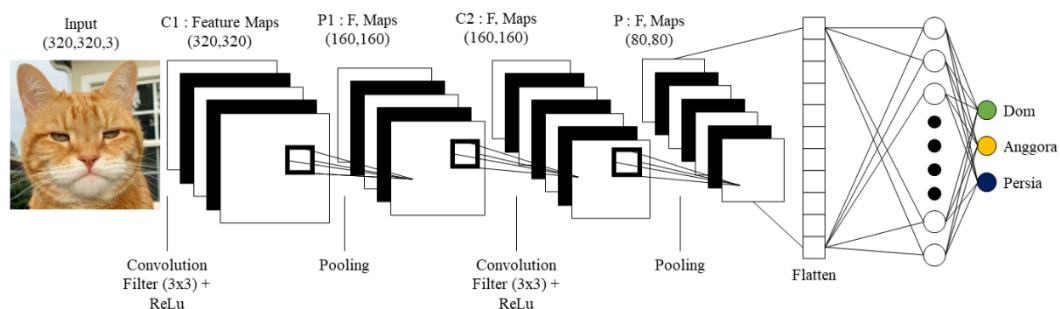
T = nilai *threshold*

## 2.4 Algoritma CNN

Algoritma CNN adalah metode komponen pada *deep learning*, umumnya dimanfaatkan untuk memahami sebuah objek pada citra digital serta melakukan pemrosesan pada citra digital (Felix *et al.*, 2020). Algoritma ini dirancang untuk memproses data piksel dan citra visual (Felix *et al.*, 2020). Kemampuan dari algoritma *Convolutional Neural Network* ini diklaim model terbaik untuk menyelesaikan persoalan *Object Detection* dan *Object Recognition* (Felix *et al.*, 2020).

*Convolutional Neural Network* digunakan untuk melakukan klasifikasi data yang berlabel dengan menggunakan metode *supervised learning*. Metode tersebut memiliki data yang dilatih dan variabel yang ditargetkan sehingga tujuan dari metode ini untuk mengelompokkan data ke data yang sudah ada. CNN sering digunakan untuk mengenali benda atas pemandangan dan melakukan deteksi dan melakukan segmentasi objek. CNN melakukan proses belajar langsung melalui data citra, sehingga dapat menghilangkan ekstraksi ciri dengan cara manual.

CNN juga merupakan saraf yang dikhususkan untuk memproses data yang memiliki struktur kotak (*grid*). Sebagai contoh yaitu berupa citra dua dimensi. Nama konvolusi merupakan operasi dari aljabar linear yang mengalikan matriks dari *filter* pada citra yang akan diproses. Proses ini disebut dengan lapisan konvolusi dan merupakan salah satu jenis dari banyak lapisan yang bisa dimiliki dalam suatu jaringan. Meskipun begitu, lapisan konvolusi ini merupakan lapisan utama yang paling penting digunakan. Jenis lapisan yang lain yang biasa digunakan adalah *pooling layer*, yakni lapisan yang digunakan untuk mengambil suatu nilai maksimum atau nilai rata-rata dari bagian-bagian lapisan piksel pada citra.



Gambar 2. 6 Arsitektur Algoritma CNN  
(Sumber : Mardiyah (2020))

Pada Gambar 2.6 merupakan sebuah alur dari CNN, *hidden layers* sebagai ekstraksi fitur dan klasifikasi. Pada *convolution layer* memiliki tujuan untuk menghapus garis lain pada citra, sehingga yang didapatkan hanya garis vertikal dan horizontal. Selanjutnya, pada *pooling layers* digunakan *max pooling* yang di mana akan membagi output menjadi beberapa bagian kotak dan mengambil nilai maksimum dari kotak yang telah dibagi. Hal ini bertujuan untuk menurunkan ukuran gambar agar bisa diubah dengan *convolutional layer* dengan *stride* yang

sama dengan *pooling layer*. Bentuk dari *pooling* akan meminimalisir *feature map* sampai 75% dari *size* aslinya.

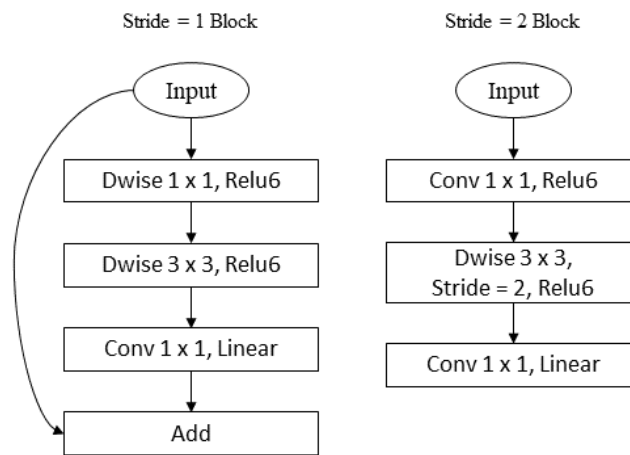
Setelah melalui *convolution layer* dan *pooling layer* (ekstraksi fitur), komponen selanjutnya adalah klasifikasi yang terdiri dari beberapa layer yang terhubung secara penuh (*Fully Connected*). Layer ini hanya dapat menerima data berdimensi 1. Untuk mengkonversi data 3 dimensi menjadi 1 dimensi, kita bisa menggunakan fungsi *flatten*. *Neuron* pada layer yang terhubung secara penuh memiliki koneksi menyeluruh ke semua aktivasi *layer* sebelumnya. Bagian ini pada prinsipnya sama dengan *Neural Network* biasa (Khoiruddin *et al.*, 2022).

## 2.5 Deep Learning

*Deep learning* adalah komponen dari *machine learning* yang dapat memahami kerja pikiran manusia (Arifianto, 2022). *Deep learning* menggunakan jaringan syaraf tiruan berlapis-lapis untuk mengetahui nilai akurasi yang tinggi dalam melakukan *voice recognition*, menerjemahkan bahasa, dan khususnya deteksi objek (Rosalina & Wijaya, 2020). *Deep learning* memiliki arsitektur yang fleksibel karena dapat belajar dari data mentah dan meningkatkan akurasi prediktif bila dimasukan lebih banyak datanya (Arifianto, 2022; Rosalina & Wijaya, 2020).

## 2.6 MobileNetV2

*MobileNet* merupakan arsitektur *Convolutional Neural Network* (CNN) yang digunakan untuk mengatasi perhitungan dengan kebutuhan sumber data yang berlimpah, sehingga *MobileNet* cocok digunakan untuk *training* pada *dataset* yang berjumlah besar (Dharmaputra *et al.*, 2021). *MobileNetV2* dirilis pada tahun 2018 dengan adanya fitur tambahan yaitu *linear bottleneck* menggunakan *depthwise seperable convolution* dan *shortcut connections* antar layer (Dharmaputra *et al.*, 2021) . Gambar 2.7 menunjukkan arsitektur *MobileNetV2*.



Gambar 2. 7 Arsitektur *MobileNetV2*  
(Sumber : Dharmaputra *et al.*(2021))

Pada Gambar 2.7 terdapat dua tipe blok yaitu blok *residual* dengan *stride of 1* dan blok kedua *stride of 2* untuk *downsizing* citra. Dapat dilihat juga ada tiga *layers* pada tiap blok, pertama konvolusi 1x1 dengan ReLU6, *layer* kedua adalah *depth* konvolusi, dan *layer* ketiga adalah konvolusi 1x1.



### BAB III

#### METODE PENELITIAN

##### 3.1 Objek Penelitian

Pada penelitian ini, objek yang digunakan citra terutama dalam membaca objek pada penyakit kulit kucing yang terlihat secara kasat mata. Objek yang diteliti berjumlah 3 kelas yakni *scabies* atau kudis, tungau, dan *ringworm* atau kurap.

##### 3.1.1. Lokasi dan Waktu Penelitian

Lokasi dan waktu penelitian dilakukan di *Zhot Pet Shop & Pet Care* pada tanggal 10 November 2022 sampai dengan 29 Mei 2023. Berikut rinci penelitian berdasarkan Tabel 3.1.

Tabel 3. 1 Tabel Paparan Penelitian

No	Kegiatan	April 2023				Mei 2023			
		1	2	3	4	1	2	3	4
1	Analisis masalah								
2	Pengumpulan data								
3	Perancangan								
4	Implementasi								
5	Pengujian								

##### 3.1.2. Peralatan Penelitian

Peralatan yang digunakan penelitian ini meliputi perangkat keras dan perangkat lunak dengan rincian sebagai berikut :

###### 1. Perangkat Keras

Penelitian ini menggunakan perangkat keras dengan spesifikasi Laptop Intel (R) *Core* (TM) i5-10300H CPU @ 2.50GHz (8 CPUs), ~ 2.5GHz *Memory* 8192MB, RAM. 3, SSD: 512 GB 4. *Windows 10 Pro 64-bit*.

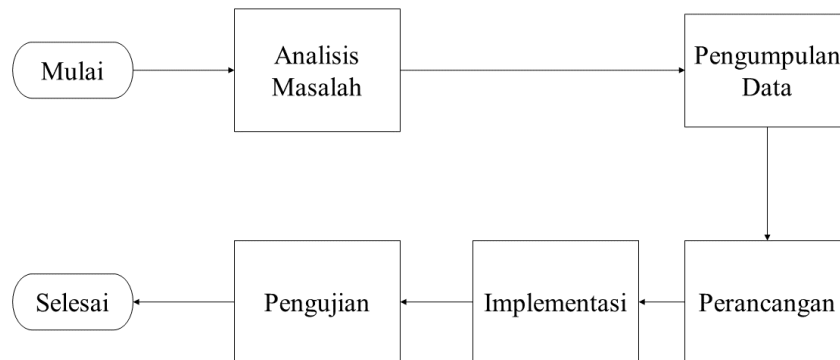
*Smartphone* dengan penyimpanan perangkat 128 GB (Total), RAM 6.00 GB Versi Android 11, *Processor* Qualcomm SDM710 Delapan-  
inti.

## 2. Perangkat Lunak

- 1) Google Colab, untuk membuat program dan menjalankan program yang sudah dieksekusi. Program yang ada juga tetap tersimpan menggunakan bahasa python.
- 2) Google Chrome, untuk mengumpulkan *dataset* yang dicari melalui sumber internet.
- 3) Android Studio, untuk membuat program dan menjalankan program yang terintegrasi. Android Studio juga untuk perancangan tampilan sistem aplikasi android menggunakan bahasa java.
- 4) Opencv, pustaka yang digunakan untuk kebutuhan dalam pemrograman. Khususnya program implementasi membuka kamera *smartphone*.
- 5) Tensorflow, salah satu pustaka yang digunakan untuk pemodelan deteksi objek yang diteliti ketika proses *training dataset*.
- 6) *Command Prompt*, untuk menjalankan program *labelimg* yang berfungsi melabeli data pada gambar, serta mengkonversi file model yang dilatih menjadi tensorflow *lite*.
- 7) Visual Studio *Code*, perangkat lunak ini digunakan untuk melakukan konfigurasi program yang nantinya harus diolah terlebih dahulu.

### 3.2 Prosedur Penelitian

Prosedur penelitian dilakukan secara bertahap untuk memberikan hasil dari suatu permasalahan. Prosedur penelitian terdapat analisis masalah, studi pustaka, pengumpulan data, perancangan, implementasi, pengujian, dan evaluasi hasil. Alur dari prosedur penelitian ditunjukkan pada Gambar 3.1.



Gambar 3. 1 Prosedur Penelitian

#### 3.2.1 Analisis Masalah

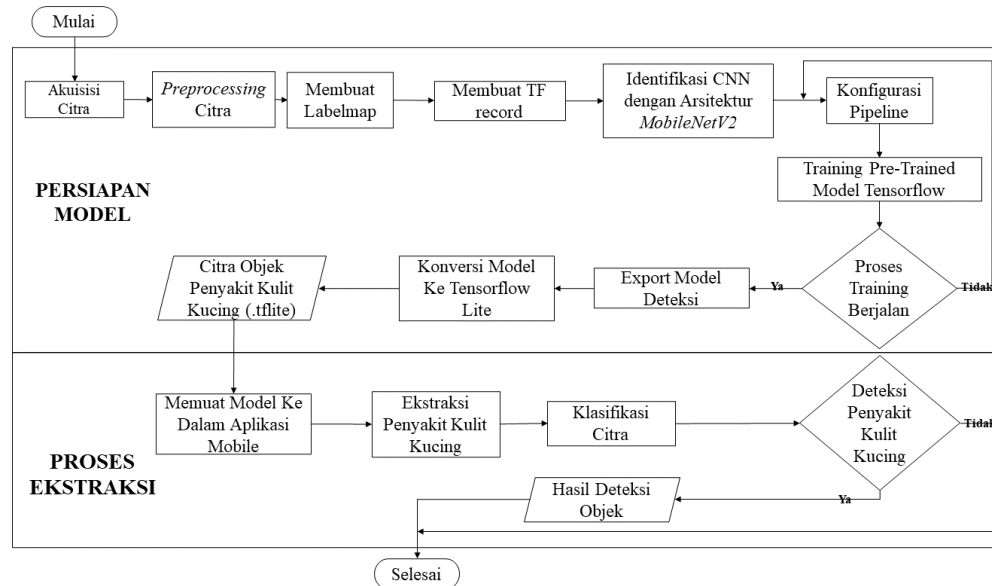
Penelitian dimulai dari tahapan analisis masalah. Permasalahan diselesaikan dengan melakukan analisis masalah citra penyakit kulit kucing. Untuk melakukan deteksi objek citra penyakit kulit kucing. Parameter yang digunakan *scabies* atau kudis, tungau, dan *ringworm*.

#### 3.2.2 Pengumpulan Data

Pengumpulan data dilakukan setelah studi pustaka. Perolehan data didapat dengan data sekunder melalui sumber lain yang sudah ada diinternet. Data sekunder penelitian, berupa foto penyakit kulit pada kucing yang didapatkan dari layanan situs (Images, 2022) berdasarkan sumber yang sudah ada. Data primer diperoleh melalui observasi dan juga wawancara terhadap seorang pakar dokter hewan yaitu Drh. Fyar F. H TA yang memiliki pengetahuan tentang penyakit kulit pada kucing. Teknik pengumpulan data dilakukan secara studi pustaka dengan mempelajari literatur-literatur dari jurnal penelitian yang berkaitan dengan permasalahan yang dibahas.

### 3.2.3 Perancangan

Pada Gambar 3.1 prosedur penelitian berikutnya yaitu perancangan. Perancangan penelitian mencakup perancangan model yang akan beroperasi pada penelitian ini ditunjukkan oleh Gambar 3.2.



Gambar 3. 2 Alur Perancangan Model

Gambar 3.2 menjelaskan alur sistem yang terbagi menjadi 2 tahapan utama. Alur sistem mempersiapkan model deteksi terdiri dari akuisisi, pelabelan citra, pembuatan *labelmap*, dan pembuatan tensorflow *record*.

Kemudian, identifikasi CNN arsitektur *MobileNetV2* untuk mengetahui ukuran citra jenis model yang digunakan. Setelah itu, model deteksi yang sudah dilatih akan tersimpan, dan siap untuk melakukan *export* model deteksi. Selanjutnya, mengkonversinya menjadi model citra objek penyakit kulit kucing dengan eksistensi jenis file tensorflow *lite*. Setelah rangkaian pembuatan model selesai, memuat model deteksi objek tersebut ke dalam aplikasi *mobile*. Sehingga, dapat melanjutkan ke tahap proses ekstraksi dan menampilkan hasil deteksi citra tersebut.

#### 1. Persiapan Model

Model deteksi membutuhkan *datataset* citra yang diberi label sebagai kebutuhan data latih untuk mendeteksi objek atau kelas. Persiapan model

deteksi objek terdiri dari beberapa tahapan yaitu akuisisi citra, *preprocessing* citra, pembuatan *labelmap*, dan pembuatan tensorflow *record*. Pada penelitian ini, *library* tensorflow yang digunakan untuk mendeteksi objek dengan tensorflow 2. Menggunakan deteksi objek API di mana metode *deep learning* model CNN sebagai algoritma dengan arsitektur *MobileNetV2*.

### 1. Akuisisi Citra

Melakukan tahapan akuisisi saat membuat folder yang diberi nama *train\_demo* dibuat untuk menaruh *dataset* yang telah diakuisisi. Selanjutnya, citra akan diambil untuk melatih model deteksi objek yang bertugas untuk mendeteksi penyakit kulit pada kucing. Citra diambil menggunakan layanan situs (Images, 2022) dengan kata kunci penyakit kulit yang dicari.

Seluruh citra tersebut disimpan ke dalam dua folder yang dinamakan *train* dan *test* di dalam folder *images*. Kualitas masing-masing 80% dan 20% berdasarkan jumlah citra yang dijadikan *dataset*. Citra diambil dan digunakan sebagai *data test*. Perlu diketahui, tidak ada aturan tepat dalam menentukan kualitas jumlah citra yang diambil pada folder *train* dan *test*. Namun jumlah citra pada folder *train* tersebut harus lebih banyak dibandingkan folder *test*. Berkas citra dinamai berdasarkan angka untuk efisiensi dalam pembacaan dan pencocokan suatu citra dengan labelnya.

### 2. Pelabelan Citra

Setelah melakukan akuisisi citra ke dalam beberapa folder yang telah dibuat, citra penyakit kulit kucing perlu diberi label untuk menjadi sebuah *dataset*. Penelitian ini menggunakan *tools* yang dijalankan melalui *command prompt* bernama *labelimg*. Saat proses pelabelan, kelas-kelas objek harus ditentukan terlebih dahulu. Jika sudah ditentukan, maka pelabelan citra dilakukan pada tiap citra objek berdasarkan kelas-kelas objek yang sudah ditentukan dengan memberikan *bounding box* atau kotak pembatas.

Jika menamakan citra dengan 1.jpg, maka keluaran hasil nya juga 1.xml. Proses pelabelan citra tersebut berupa jenis *file* xml yang berisi informasi anotasi objek. Selain itu, terdapat nilai titik koordinat x dan y kotak pembatas.

### 3. Membuat *Labelmap*

Tahapan pembuatan *labelmap* diperlukan untuk mengkategorikan setiap nama kelas atau label ke dalam *format tensorflow record*. *Tensorflow record* menyimpan urutan catatan biner. *Labelmap* disimpan dalam format pbtx yang berfungsi untuk mengatur *training* model. File tersebut berfungsi saat proses pelatihan model deteksi objek dan pendeteksian objek. Struktur format pbtx berbentuk *list* item yang terdapat *id* dan *name*. Nilai *id* berupa integer untuk mengklasifikasikan pendeteksian objek. Selain itu juga, *name* mendeskripsikan nama dari objek yang dideteksi harus sama saat pelabelan citra *dataset*.

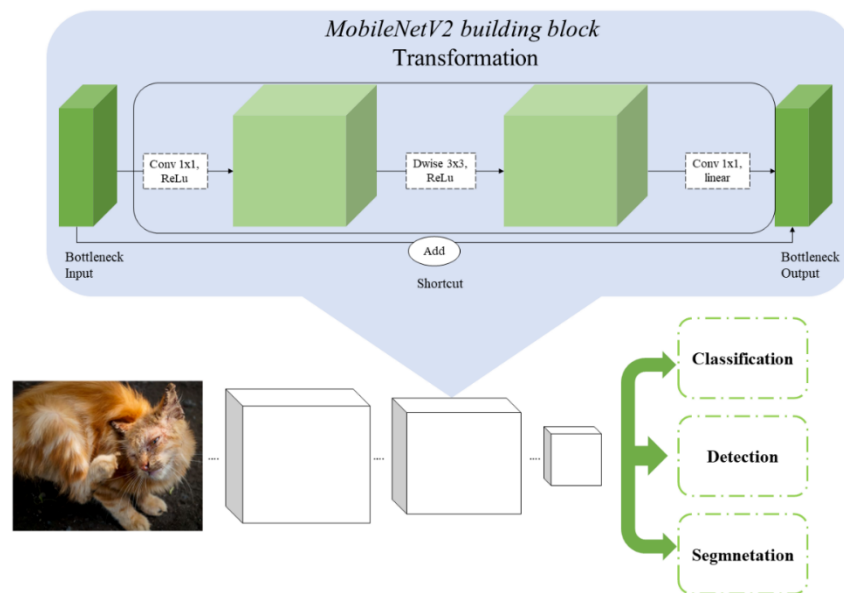
### 4. Membuat TF *Record*

Sebelum tahapan *training*, tensorflow harus membaca data *input* dalam format *record*. Tahapan pembuatan tensorflow *record* terbagi menjadi dua yaitu *record test* dan *record train*. *Tensorflow record* menyimpan rekaman data biner untuk serialisasi data terstruktur *record test* dan *record train*. *Tensorflow object detection API* memiliki fungsi *generate\_tfrecord.py*, untuk mengubah *dataset train* dan *test* ke dalam tensorflow *record*. Proses tahapan ini diperlukan sebelum atau masuk ke dalam proses tahapan pelatihan model deteksi objek yang di mana akan membaca data *record test* dan *record train*.

### 5. Identifikasi CNN Dengan Arsitektur *MobilNetV2*

Tahapan identifikasi CNN dengan arsitektur *MobileNetV2* dilakukan untuk mengklasifikasikan dan mengidentifikasi citra penyakit kulit kucing. Metode ini sering digunakan untuk mendeteksi objek ke dalam aplikasi *mobile*. Letak perbedaan yang mendasari arsitektur *MobileNetV2* dengan CNN yaitu berdasarkan lapisan konvolusi. Proses konvolusi dilakukan oleh *network MobileNet* menggunakan *depthwise separate convolutions*. Proses tersebut membagi *kernel* menjadi dua

bagian *depthwise convolution* dan *pointwise convolutions*. Adapun alur arsitektur *MobileNetV2* pada Gambar 3.3.



Gambar 3. 3 Alur Arsitektur *MobileNetV2*  
(Sumber: Ekoputris (2018))

Berdasarkan Gambar 3.3 alur arsitektur *MobileNetV2* bagian *bottleneck* memiliki dua jenis *input* dan *output* antara model. Sementara itu, *layer* bagian dalam penyatuan data kemampuan model untuk mengubah *input* piksel. Piksel tersebut diubah menjadi lebih rendah ke deskriptor tingkat yang lebih tinggi. Bagian *shortcut* antar *bottleneck* memungkinkan *training* atau pelatihan lebih cepat dan akurasi lebih baik. Model arsitektur *MobileNetV2* yang digunakan untuk penelitian ini yaitu model *Single Shot Detector (SSD) MobileNet* sebagai pendeteksi objek. Tensorflow 2 menyediakan kumpulan model deteksi yang dilatih. Hal ini berguna untuk menginisialisasi model dan kebutuhan pelatihan, serta pengujian berdasarkan kumpulan data.

#### 6. Konfigurasi *Pipeline*

Model objek deteksi memiliki konfigurasi untuk proses *training*. Konfigurasi *pipeline* dilakukan sebelum proses *training*. Konfigurasi *pipeline* memiliki fungsi untuk mengelola beberapa berkas. Berkas terdapat file *pipeline* yang tersedia pada *pre-trained* model yang dipilih.

*Pipeline* memiliki peran untuk mengatur parameter-parameter *training*, ataupun evaluasi seperti proses pelatihan, dan sebagainya. Parameter tersebut seperti jumlah kelas objek yang dideteksi, *batch\_size*, dan *num\_step*,

#### 7. *Training Pre-trained Model Tensorflow*

Tahapan *training* data menggunakan *google colab* produk *Google Research* (Arifianto, 2022). *Google colab* juga, bagian dari fitur *jupyter notebook* berbasis *cloud*. Membolehkan penelitian untuk melatih model *deep learning* pada CPU atau GPU (Arifianto, 2022). *Google Colab* memiliki GPU dan TPU secara gratis. Secara umum, komputasi menggunakan CPU untuk *pre-training* model *deep learning* memakan waktu yang lama. Maka dari itu, GPU atau TPU dibutuhkan agar proses *training* model lebih cepat. Sehingga membantu lebih banyak kontrol terhadap penelitian. Pengerjaan proses *training* model tensorflow terdiri dari 5 langkah.

Langkah pertama, penyiapan model pada tensorflow *object detection* API dengan melakukan *cloning* melalui situs (Tensorflow, 2022) yang sudah disediakan oleh tensorflow. Langkah kedua, menjalankan program *model\_builder\_tf2\_test.py* untuk melakukan pengujian model *object detection* tensorflow.

Langkah ketiga, dengan mengunggah berkas *dataset* yang telah disiapkan dan dibuat sebelumnya. Melalui penelitian ini, berkas *dataset* diunggah ke dalam *github* akun penelitian yang berisi beberapa folder dan kebutuhan program, yang sudah dipisahkan melalui model tensorflow *detection object* API dan mengkloningnya.

Langkah keempat, mengambil data latih model deteksi objek yang telah disediakan oleh tensorflow 2. Pada penelitian ini, model deteksi objek yang digunakan SSD *MobileNet V2 FPN Lite 320x320*. Setelah mengambil data tersebut, maka akan membuat tensorflow *record train* dan *test* dengan menjalankan fungsi dari *generate\_tfrecord.py*.

Kemudian, melakukan konfigurasi *pipeline* model deteksi objek tersebut dengan mengelola beberapa berkas yang digunakan untuk



kebutuhan proses pelatihan model. Langkah terakhir, menjalankan program *model\_main\_tf2.py* sebagai kebutuhan proses pelatihan model.

#### 8. *Export Model Deteksi*

Ketika model sudah dilatih, tahapan selanjutnya mengekspor model deteksi objek. *Export* model berfungsi untuk arsitektur grafik dan operasi jaringan kompatibel. Tensorflow pada perangkat seluler dalam hal ukuran data yang digunakan model, persyaratan proses perangkat kerasnya, serta *size* dan kompleksitas model keseluruhan. Tahapan ini dapat dilakukan dengan menjalankan fungsi dari *export\_tflite\_graph\_tf2.py*. Keluaran dari proses mengekskpor model menyimpan berkas folder *saved\_model* yang berisi folder *assets*, *variables*, dan file *saved\_model.pb*.

#### 9. Konversi Model Menjadi Tensorflow Lite

Ada berbagai macam cara untuk mengkonversi model deteksi menjadi tensorflow *lite*. Namun penelitian ini menggunakan *saved\_model.pb* yang sudah terproses ekspor sebelumnya. Konversi tensorflow *lite* mengambil dari model tensorflow dan menghasilkan model tensorflow *lite*. Memuat *saved\_model.pb* yang sudah tersimpan, maka konversi dapat dijalankan melalui CMD. *Output* hasil berkas model deteksi objek, memiliki jenis file tensorflow *lite* yang siap untuk diterapkan ke dalam aplikasi *mobile*.

## 2. Proses Ekstraksi

#### 1. Memuat Model Ke Dalam Aplikasi *Mobile*

Sebelum memuat model ke dalam aplikasi *mobile*, selanjutnya membuat label dengan jenis file txt. Isi file tersebut berupa nama kelas-kelas objek yang dideteksi.

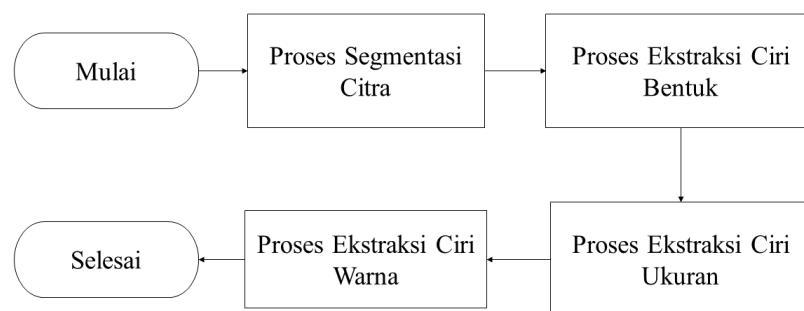
Setelah model dilatih, diekspor, dan dikonversi menjadi tensorflow *lite*. Selanjutnya, memuat model ke dalam aplikasi *mobile* dengan cara melakukan *drag* dan *drop* ke dalam folder *assets tools*.

Selanjutnya memuat model deteksi objek ke dalam android studio. Deklarasi model tersebut melalui program java dengan menggunakan

*library* opencv. Lalu mengisi *size* model deteksi yang digunakan beserta nama file ekstensi txt yang digunakan.

## 2. Ekstraksi Ciri

Ekstraksi ciri memiliki fungsi untuk mengekstrak informasi objek atau data melalui sebuah citra penyakit kulit kucing. Proses ekstraksi terlebih dahulu tersegmentasi untuk mengambil objek dalam citra. Sehingga citra tersebut dapat dikenali dan dibedakan dengan objek lainnya. Ciri citra tersebut dapat digunakan sebagai parameter (warna, bentuk, dan ukuran) pembeda antara objek satu dengan objek lainnya. Adapun prosedur ekstraksi ciri pada Gambar 3.4.



Gambar 3. 4 Prosedur Ekstraksi Ciri Citra Penyakit Kulit Kucing

### a. Proses Segmentasi Citra

Proses segmentasi citra dilakukan dengan teknik ruang warna yang sederhana. Mengkonversi citra dari *Red Green Blue* (Red Green Blue). Warna tersebut dapat direpresentasikan menjadi komponen warna merah, hijau, dan biru. RGB mendeskripsikan warna sebagai tiga komponen, setiap komponen tersebut memiliki nilai antara 0 sampai dengan 255 di mana nilai (0, 0, 0) mewakili hitam, sedangkan (255, 255, 255) mewakili putih. Proses segmentasi citra menggunakan jenis pendekatan *neural network* atau jaringan saraf. Segmentasi ini, berbasis jaringan saraf. Pendekatan tersebut, digunakan dalam mensegmentasi citra penyakit kulit kucing. Kemudian memisahkannya dari latar belakang.

Hasil dari proses segmentasi citra biner objek *foreground* berwarna putih (bernilai 1). Lalu akan dipisahkan dengan *background* yang dihilangkan berwarna hitam (bernilai 0). Sehingga pengenalan pola dapat tersegmentasi, dan proses ekstraksi ciri citra dapat dilakukan.

b. Proses Ekstraksi Ciri Bentuk

Ekstraksi ukuran citra bentuk menggunakan parameter yang disebut *eccentricity*. *Eccentricity* memiliki rentang nilai antara 0 hingga 1. Objek berbentuk memanjang garis lurus, nilai *eccentricity* mendekati angka 1. Sedangkan objek berbentuk bulat, nilai *eccentricity* mendekati 0. Perhitungan *eccentricity* dapat ditunjukkan melalui persamaan 6.

$$e = \sqrt{1 - \frac{a^2}{b^2}} \quad (6)$$

Dimana :

$e = eccentricity$

$a = mayor\ axis$

$b = minor\ axis$

c. Proses Ekstraksi Ciri Ukuran

Proses ekstraksi ciri menggunakan parameter luas dan keliling. Proses ini untuk membandingkan ukuran objek satu dengan objek lainnya, Ciri yang diekstraksi yaitu ciri ukuran (luas dan keliling). Sementara letak atau posisi (koordinat) melalui objek. Peran luas dalam ekstraksi sebagai piksel yang menyusun objek, apabila keliling banyaknya piksel yang berada pada objek.

d. Proses Ekstraksi Ciri Warna

Proses ekstraksi ciri warna untuk membandingkan objek warna menggunakan nilai *hue*. *Hue* merepresentasikan dari warna (merah, jingga, kuning, hijau, biru, dan ungu). Nilai *hue* dapat dikolaborasikan dengan nilai *saturation* dan *value*.

Mendapatkan nilai tersebut diperlukan konversi ruang warna citra RGB (red, green, blue) melalui persamaan 7.

$$R' = R/255 \quad (7)$$

$$G' = G/255$$

$$B' = B/255$$

$$Cmax = \max(R', G', B')$$

$$Cmin = \min(R', G', B')$$

$$\Delta = Cmax - Cmin$$

Perhitungan nilai *hue* dapat dituliskan melalui persamaan 8.

$$H = \begin{cases} 0^\circ & \Delta = 0 \\ 60^\circ \times \left( \frac{G' - B'}{\Delta} \bmod 6 \right), Cmax = R' \\ 60^\circ \times \left( \frac{B' - R'}{\Delta} + 2 \right), Cmax = G' \\ 60^\circ \times \left( \frac{R' - G'}{\Delta} + 4 \right), Cmax = B' \end{cases} \quad (8)$$

Perhitungan nilai *saturation* dapat dituliskan melalui persamaan 9.

$$S = \begin{cases} 0, Cmax = 0 \\ \left( \frac{\Delta}{Cmax} \right), Cmax \neq 0 \end{cases} \quad (9)$$

Perhitungan nilai *value* dapat dituliskan seperti persamaan 10.

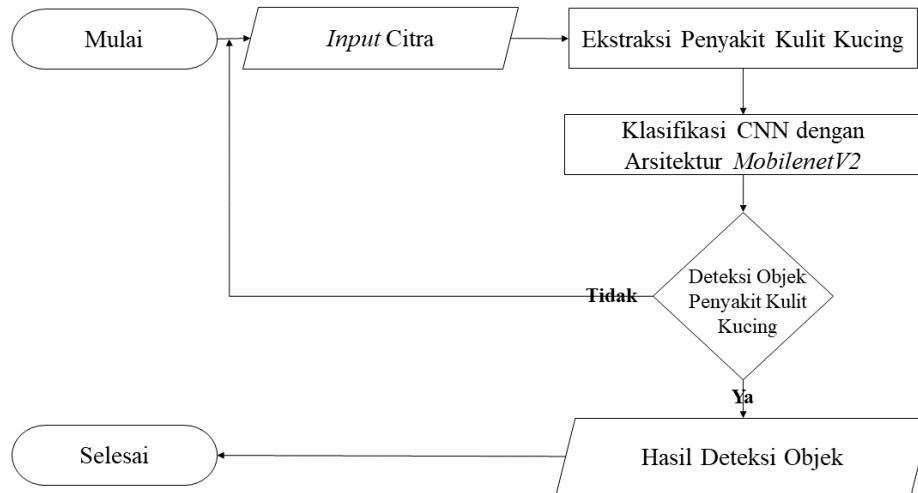
$$V = Cmax \quad (10)$$

### 3. Klasifikasi Citra

Arsitektur yang digunakan *MobileNetV2* untuk mendeteksi objek citra penyakit kulit kucing melalui aplikasi *mobile*. Metode ini digunakan untuk mengklasifikasikan data citra. Proses ekstraksi *MobileNetV2* memiliki beberapa lapisan yang tidak terlihat secara kasat mata. Lapisan blok *residual* dengan *stride of 1* dan blok kedua *stride of 2* untuk *downsizing* citra. Terdapat juga tiga *layers* pada tiap blok. Pertama konvolusi 1x1 dengan ReLU6, *layer* kedua adalah *depth* konvolusi, dan *layer* ketiga adalah konvolusi 1x1.

### 3.2.4 Implementasi

Pembuatan model deteksi objek untuk menyelesaikan masalah dengan memanfaatkan *image processing* dalam pengolahan citra. Berdasarkan Gambar 3.5 dapat dilihat alur implementasi metode sistem.



Gambar 3. 5 Flowchart Alur Sistem

#### 1. Input Citra

Tahapan *input* citra untuk menggunakan kamera *smartphone* yang telah dikelola khusus. Selanjutnya pada aplikasi *mobile* memasukan citra untuk mengambil objek penyakit kulit kucing menggunakan kamera *smartphone* android. Peluncuran kamera *smartphone* android menggunakan *library* opencv, dan pengambilan citra penyakit kulit kucing dapat dilakukan secara *real-time*.

#### 2. Ekstraksi Penyakit Kulit Kucing

Setelah menerima *input* citra pengambilan objek melalui kamera *smartphone*. Sistem akan melakukan ekstraksi ciri objek citra penyakit kulit kucing yang terdeteksi. Proses ekstraksi citra berfungsi untuk mengekstrak informasi objek melalui sebuah citra penyakit kulit kucing sehingga citra tersebut dapat dikenali.

### 3. Klasifikasi CNN Dengan Arsitektur *MobileNetV2*

Ketika sistem sudah mengekstraksi informasi objek citra penyakit kulit kucing. Sistem akan melakukan klasifikasi menggunakan algoritma CNN dengan arsitektur *MobileNetV2*.

### 4. Deteksi Penyakit Kulit Kucing

Kemudian jika mendeteksi maka sistem akan menampilkan hasil klasifikasi deteksi objek citra penyakit kulit kucing. Hasil deteksi objek citra menampilkan nilai presisi dan mendeskripsikan kelas-kelas objek. Jika tidak mendeteksi maka sistem kembali menerima *input* citra.

### 3.2.5 Pengujian

Pada penelitian ini, jenis pengujian dilakukan untuk menguji model deteksi *training* model tensorflow, dan menguji sistem aplikasi *mobile*. Pengujian model deteksi objek untuk menguji hasil model *training* menggunakan tensorboard. Apabila sudah bisa mendeteksi penyakit kulit kucing melalui beberapa gambar uji. Data hasil *training* model berupa grafik *learning rate*, *total loss* melalui *loss function* dan model *object detection*.

*Learning rate* mendeskripsikan grafik parameter *training* untuk dihitung nilai koreksi bobot pada waktu proses pelatihan berjalan (Arifianto, 2022). *Learning rate* memiliki nilai berada pada *range* 0 hingga 1. Meningkatnya nilai *learning rate*, maka proses *training* berjalan semakin cepat (Arifianto, 2022). Jika *learning rate* semakin kecil, ketelitian jaringan akan semakin besar atau bertambah. Resiko yang terjadi *training* model *object detection* memakan waktu semakin lama (Arifianto, 2022).

*Loss function* memiliki faktor penting yang mempengaruhi presisi deteksi dalam tugas mendeteksi objek. *Total loss* mendeskripsikan kolaborasi nilai melalui *localization loss* dan *classification loss*, nilai sebenarnya dengan hasil prediksi model. *Localization loss* menjelaskan nilai *error* yang dihasilkan melalui prediksi lokasi objek dengan sebenarnya, nilai dari perhitungan IoU (*Intersection over Union*) (Arifianto, 2022). Sedangkan *classification loss* menginterpretasikan nilai *error* yang dihasilkan dari prediksi kelas-kelas objek.

Selanjutnya, pengujian sistem menggunakan metode *black box testing* teknik *equivalence partitioning* dan *white box testing* dengan teknik *cyclomatic*. *Black box testing* menguji perangkat lunak berdasarkan fungsionalitas masukan. Serta keluaran sistem tanpa menguji desain serta kode program. Jika fungsionalitas sistem diketahui dapat menerima masukan data, maka data disimpan valid dan sebaliknya. Sementara *white box testing*, menguji segi desain dan kode program masukan dan keluaran yang sesuai. Pengujian kode program dilakukan dengan menampilkan *source code* program dan dipetakan ke *node* grafik. Menentukan kompleksitas logika program menggunakan *cyclomatic* dari *node* grafik dengan rumus perhitungan persamaan 11.

$$V(G) = E - N + 2 \quad (11)$$

Dimana : E = jumlah busur atau link

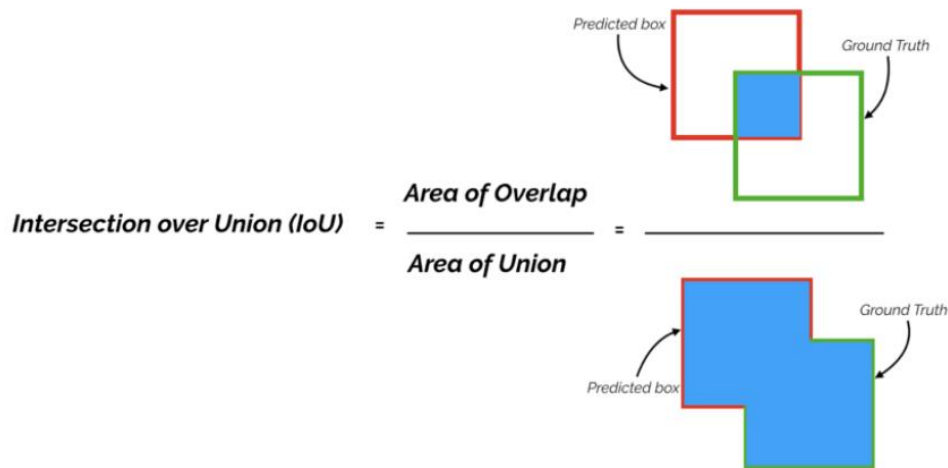
N = jumlah simpul

### 3.3 Evaluasi Hasil

Evaluasi hasil penelitian menggunakan parameter yang dikenal *performance metric* terdiri dari TP, FP, FN, dan TN. *Confusion matrix* digunakan untuk mengevaluasi model pendeteksian objek yang terdiri dari *precision*, *recall*, dan *mean average precision* (mAP) (Arifianto, 2022). Penelitian deteksi objek, membutuhkan proses klasifikasi dan lokalisasi (Arifianto, 2022). Hasil model tersebut dihitung juga menggunakan IoU (*intersection over union*) melalui persamaan 12. Gambar 3.6 menunjukkan *confusion matrix* dalam mengukur kinerja model.

Predicted	Actually		
	Positif (1)	Positif (1)	Negatif (0)
		<i>True Positive</i>	<i>False Positive</i>
	Negatif (0)	<i>False Negative</i>	<i>True Negative</i>

Gambar 3. 6 *Confusion Matrix*  
(Sumber : Nurhikmat (2020))



Gambar 3. 7 Ilustrasi Perhitungan IoU  
(Sumber : Yohanandan (2020))

$$IoU = \frac{|A1 (\text{predicted box}) \cap A2 (\text{ground truth})|}{|A1 (\text{predicted box}) \cup A2 (\text{ground truth})|} \quad (12)$$

Pada Gambar 3.6 terdapat ilustrasi perhitungan IoU untuk mengukur akurasi model deteksi objek. *Union* mendeskripsikan kolaborasi area prediksi (*A1*) dengan area *ground truth* (*A2*) (Arifianto, 2022). Sementara *intersection* mendeskripsikan potongan area prediksi dengan area *ground truth* (Arifianto, 2022). Jika skor IoU di atas ambang batas, dan nilai prediksi sesuai dengan nilai sebenarnya, maka *True Positive* (TP) (Arifianto, 2022). Jika tidak di atas ambang batas, maka itu *False Positif* (FP) (Arifianto, 2022). Apabila objek kebenaran tidak memiliki objek yang diprediksi, maka itu *False Negatif* (FN).

*Precision* mendeskripsikan nilai presisi yang didapatkan dari pembagian jumlah total contoh positif, saat benar diklasifikasikan melalui model (Harani *et al.*, 2019). Rumus *precision* yang digunakan untuk kalkulasi nilai *precision* ditulis melalui persamaan 13.

$$\text{Precision} = \frac{TP (\text{True Positive})}{TP (\text{True Positive}) + FP (\text{False Positive})} \quad (13)$$

*Recall* mendeskripsikan kemampuan model deteksi objek. *Recall* mendefinisikan nilai rasio dari jumlah total contoh positif yang di klasifikasikan dengna benar (Harani *et al.*, 2019). Nilai *recall* yang baik menampilkan klasifikasi



dikenali dengan benar (Harani *et al.*, 2019). Rumus *recall* rumus yang digunakan untuk kalkulasi nilai *recall* ditunjukkan pada persamaan 14.

$$Recall = \frac{TP (True Positive )}{TP (True Positive ) + FN (False Negative)} \quad (14)$$

*Mean average precision* (mAP) berfungsi untuk mengevaluasi modek deteksi objek (Arifianto, 2022). Jika membuat model ketika menghitung mAP hanya menghitung untuk satu kelas saja (Wilianto, 2021). Nilai mAP dan AR menunjukkan tingkat akurasi deteksi objek dan posisinya (Syaikhoni & Ariyadi, 2018). Nilai parameter mAP menjadi peran utama untuk mengukur akurasi suatu model yang akan diaplikasikan, dan seberapa baik nya model deteksi objek mengatasi objek tanpa kesalahan (Syaikhoni & Ariyadi, 2018).

Nilai mAP diperoleh dengan kalkulasi rata-rata AP dari masing-masing kelas (Arifianto, 2022). Ketika evaluasi model yang menghasilkan nilai *recall* tetapi *precision* rendah. Maka model mengklasifikasikan besar sampel positif dengan benar, tetapi memiliki banyak positif palsu (mengklasifikasikan banyak sampel negatif sebagai positif) (Syah, 2022). Saat evaluasi model menghasilkan nilai *precision* tinggi tetapi *recall* rendah. Maka model tersebut akurat ketika mengklasifikasikan sampel positif, tetapi mungkin mengklasifikasikan beberapa sampel positif (Syah, 2022) .Oleh karena itu, akurasi dipastikan kurang baik apabila terjadi *imbalanced* pada mAP dan *Average Recall*. Persamaan menghitung mAP dapat ditulis melalui persamaan 15.

$$mAP@ \propto = \frac{1}{n} \sum_{i=1}^n AP_i \quad (15)$$

Dimana :  $n$  = jumlah kelas

$AP_i$  = jumlah kelas

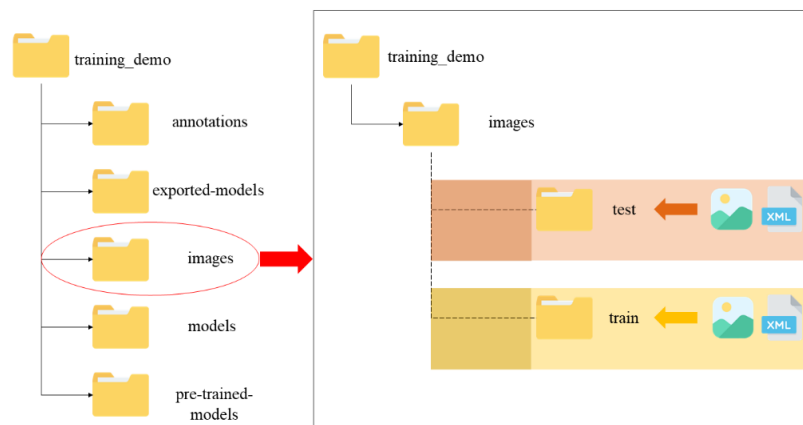
## BAB IV

### HASIL DAN PEMBAHASAN

#### 4.1 Perancangan Model

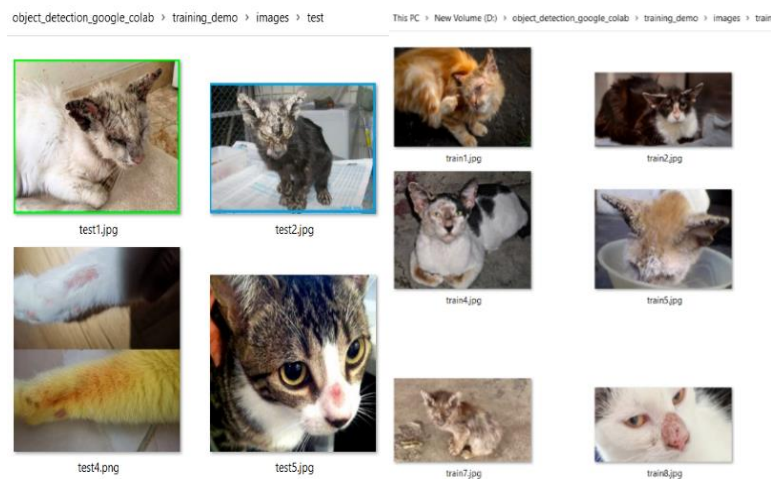
##### 4.1.1 Akuisisi Citra

Sebanyak 43 citra *dataset* yang berhasil didapatkan dan dikelompokkan ke dalam folder *images*. Citra tersebut disimpan ke dalam dua folder yang dinamakan *train* dan *test* di dalam folder *images*. Folder *test* dan *train* memiliki jumlah takaran citra yang berbeda. Struktur folder *images* untuk *dataset* dapat ditunjukkan pada Gambar 4.1.



Gambar 4. 1 Struktur Folder *Images*

Sebanyak 34 atau 80% dari jumlah *dataset* pada folder *train* penyakit kulit kucing yang terdiri dari kudis, *ringworm*, dan tungau. Sedangkan folder *test* sebanyak 9 atau 20% dari jumlah *dataset* penyakit kulit kucing terdiri dari kudis, *ringworm*, dan tungau. Gambar 4.2 menunjukkan isi folder *test* dan *train*.



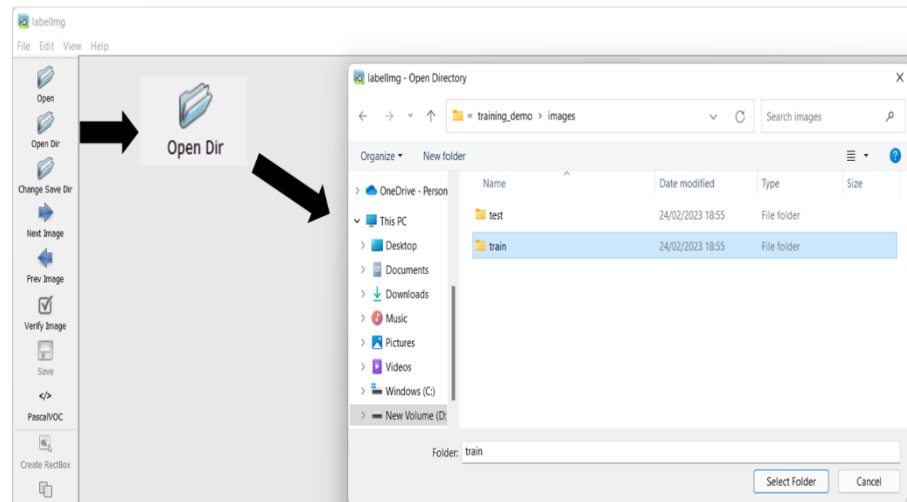
Gambar 4. 2 Isi Folder *Test* dan *Train*

Pada Gambar 4.2 isi folder *test* dan *train* memiliki ciri penamaan yang khusus. Berkas citra folder *test* dinamakan “test”, sementara berkas citra folder *train* dinamakan “train”. Penamaan tersebut disertai dengan angka untuk kemudahan pengelompokan saat membaca berkas.

#### 4.1.2 Preprocessing Citra

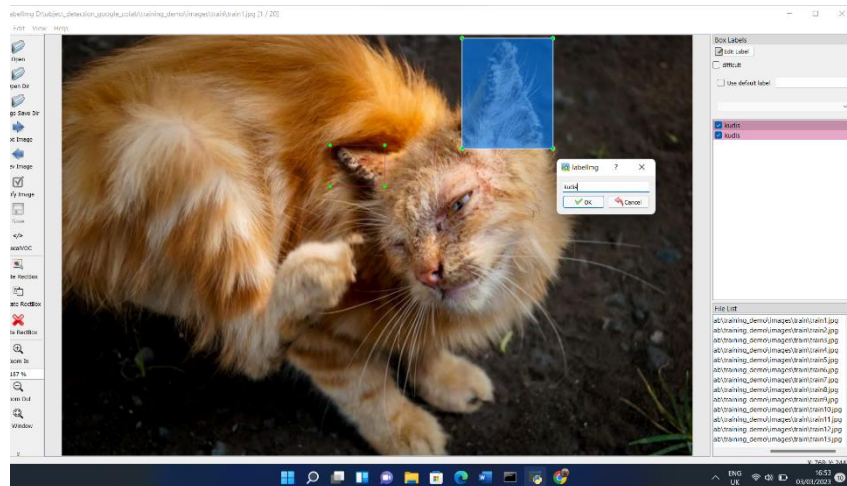
##### 1. Pelabelan Citra

Pelabelan citra menggunakan *labelimg* yang di mana harus diinstal terlebih dahulu. Menginstal *labelimg* dengan membuka CMD pada komputer dengan mengetikkan *pip install labelimg*. Tahap pelabelan citra untuk memberi *bounding box* terhadap citra yang sudah disusun. Pelabelan ini dilakukan bersama dengan pakar agar model deteksi dapat belajar sesuai dengan ciri penyakit kulit kucing yang terjangkau. Proses pelabelan citra menggunakan *labelimg* yang dipanggil menggunakan CMD. Citra yang tersusun di dalam berkas *test* dan *train* dibuka direktorinya melalui *labelimg* seperti pada Gambar 4.3.



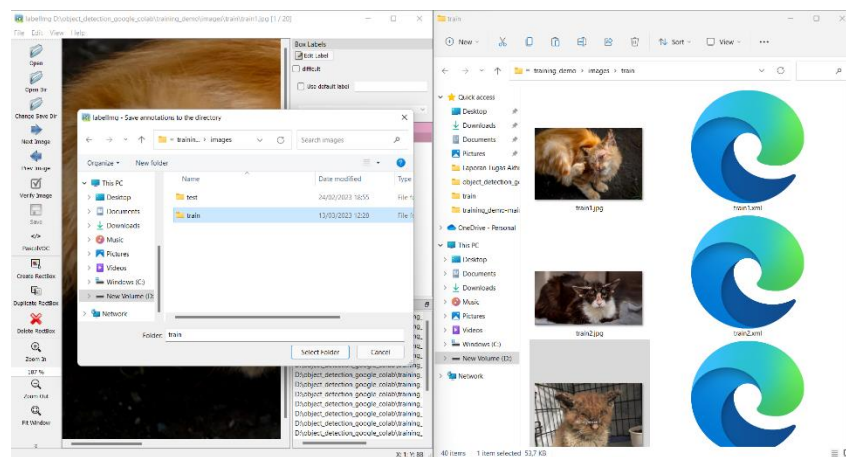
Gambar 4. 3 Membuka Direktori Folder *Test* dan *Train*

Proses pelabelan citra dapat dilakukan dengan memberikan *bounding box* terhadap objek yang dideteksi. Proses pelabelan dan penamaan citra dapat ditunjukkan pada Gambar 4.4.



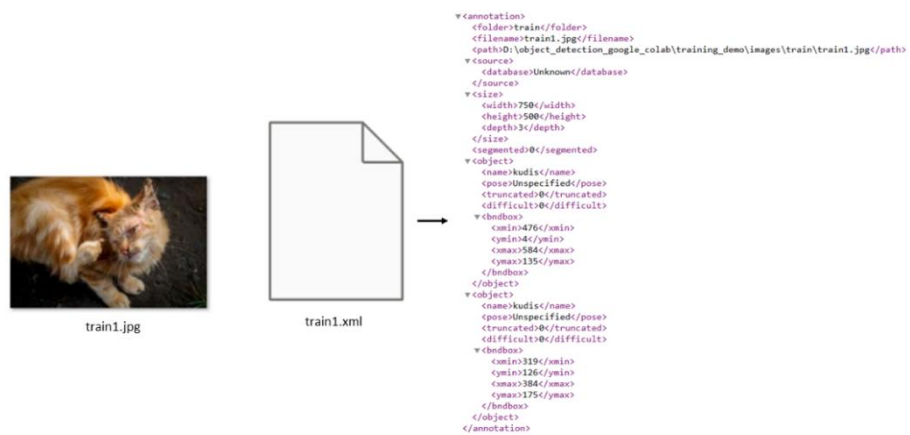
Gambar 4. 4 Pelabelan Citra

Pada Gambar 4.4 proses pelabelan citra dilakukan ketika memberi *bounding box* berdasarkan citra yang dideteksi. Saat pelabelan citra, perlu juga menentukan penamaan label pada citra. Penamaan label citra terdapat “kudis”, “ringworm”, dan “tungau”. Format pelabelan citra menggunakan PascalVOC, dan hasil pelabelan tersebut disimpan ke dalam folder *train* seperti pada Gambar 4.5.



Gambar 4. 5 Menyimpan Hasil Pelabelan

Pada Gambar 4.5 hasil format pelabelan citra berupa file xml dengan menggunakan PascalVOC. File xml berisi anotasi citra yang telah diberikan *bounding box* berdasarkan titik koordinat objek yang dipilih. Gambar 4.6 menunjukkan isi file xml.



Gambar 4. 6 Isi File XML

#### 4.1.3 Membuat *Labelmap*

Mengelola file *labelmap* di mana file ini didapat melalui pakar, file tersebut bisa diunduh dan ditempatkan di dalam folder *annotations*. *Labelmap* disimpan dalam format pbtX yang berfungsi untuk mengatur pelatihan model. Struktur format pbtX *id* diisi dengan nilai integer untuk mengklasifikasikan objek yang dideteksi. Sementara *name*, diisi berdasarkan nama objek yang dideteksi seperti ditunjukkan pada Gambar 4.7.

```

item {
  id: 1
  name: 'kudis'
}
item {
  id: 2
  name: 'ringworm'
}
item {
  id: 3
  name: 'tungau'
}

```

Gambar 4. 7 Isi File *Labelmap*

Pada Gambar 4.7 isi file *labelmap* berisi 3 item dengan *id* 1 sampai dengan 3 dan *name* yang berisi nama objek “kudis”, “ringworm”, dan “tungau”. File *labelmap* format pbtX digunakan saat pelatihan model deteksi objek untuk mengidentifikasi kelas label yang digunakan.

#### 4.1.4 Membuat TF Record

##### 1. TF Record Train

*Dataset* pada folder *train* dan juga file *labelmap* yang sudah disiapkan, kemudian diubah ke dalam format *record*. Perubahan format menggunakan *google colab* program *generate\_tfrecord.py* yang terdapat pada tensorflow API. Hasil dari program tersebut berupa berkas yang dinamakan *train.record* yang dapat ditunjukkan pada Gambar 4.8.



Gambar 4. 8 File *Train Record*

##### 2. TF Record Test

Kemudian, membuat *record test* melalui dataset folder *test* dan file *labelmap* yang sudah disiapkan. Perubahan format *record test* juga menggunakan *google colab* dengan menjalankan program *generate\_tfrecord.py*. Hasil dari program tersebut berupa berkas yang dinamakan *test.record* yang dapat ditunjukkan pada Gambar 4.9.



Gambar 4. 9 File *Test Record*

#### 4.1.5 Konfigurasi Pipeline

Penelitian ini menggunakan model arsitektur *MobileNetV2 FPN Lite* 320x320, berkas *pipeline* berada pada berkas model arsitektur yang digunakan. File *pipeline.config* memiliki struktur yang harus dikonfigurasi sebelum masuk ke dalam penelitian (lampiran).

#### 4.1.6 Training Model Deteksi Objek

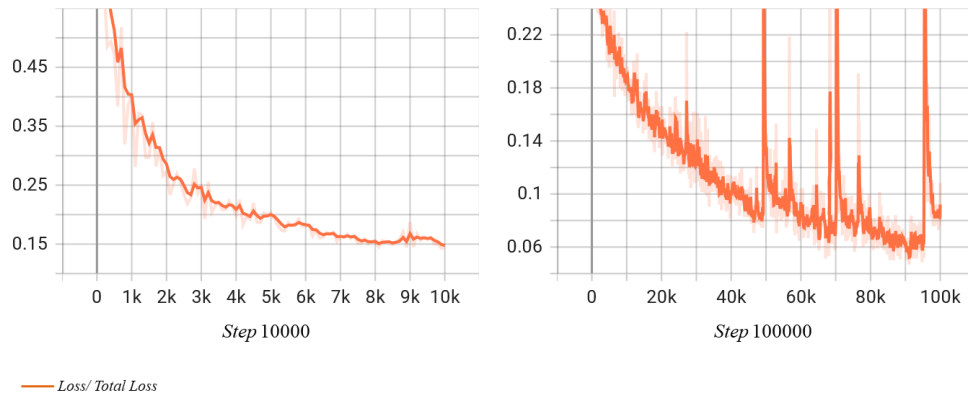
Pelatihan memiliki tahapan penting dalam melatih model deteksi objek untuk memahami bentuk gambar agar dapat mengetahui objek di dalamnya. Proses pelatihan model ini yang dilatih untuk mempelajari ciri-ciri dari objek kelas yang dideteksi dengan melakukan penyesuaian bobot dan bias pada tiap neuron. Parameter program yang perlu disiapkan terdapat direktori model, dan letak file *pipeline.config* yang sudah dikelola.

Pada penelitian ini, proses pelatihan berhenti ketika jumlah *steps* yang terdapat di dalam berkas *pipeline.config*. Selama proses pelatihan berjalan, dapat dipantau dari *log* yang ditampilkan *google colab*. Normalnya, nilai *total loss* semakin mengecil dengan bertambahnya *step* yang dilewati. Hasil dari percobaan pelatihan dengan jumlah *step* yang berbeda dapat dilihat pada Tabel 4.1.

Tabel 4. 1 Tabel Percobaan Pelatihan Model

Total Step	Waktu Pelatihan	Average Precision (mAP)	Average Recall (mAR)
10000	16 menit	0.420080	0.230000
100000	4 jam 52 menit	0.205778	0.216667

Berdasarkan Tabel 4.1 telah dilakukan pengujian model deteksi objek terhadap *total step* yang berbeda. Normalnya, bertambah *step* nilai mAP dan mAR semakin meningkat, namun berdasarkan tabel tersebut terjadi penurunan. Pengujian model ini menunjukkan model pendeteksian objek mengalami *overfitting*. Hal ini terjadi karena *dataset* yang diperoleh terbatas dan jumlah yang sedikit. Model dengan *step* 10000 memiliki nilai mAP dan nilai mAR yang tinggi. Sedangkan model *step* 100000 memiliki mAP dan nilai mAR yang rendah. Sehingga model *step* 10000 menunjukkan performa yang bagus untuk mendeteksi karena memiliki nilai mAP dan nilai mAR. Perolehan nilai *total loss* model deteksi objek berdasarkan jumlah *step* dapat ditunjukkan pada Gambar 4.10.



Gambar 4. 10 Grafik Total *Loss* Berdasarkan Jumlah *Step*

Berdasarkan Gambar 4.10 menunjukkan grafik total *loss* berdasarkan jumlah *step* 10000, dan 100000. Total *loss* terdiri dari kolaborasi *localization loss* dan *classification loss* jarak antara nilai sebenarnya dengan hasil prediksi model. *Localization loss* berupa nilai *error* yang dihasilkan dari prediksi lokasi objek yang sebenarnya, nilai dari perhitungan IoU. Sementara *classification loss* berupa nilai *error* yang dihasilkan dari prediksi kelas objek. Garis oren menunjukkan nilai total *loss*, jumlah total *loss* ketika *step* 10000 bernilai 1.31. Sedangkan ketika *step* 100000 bernilai 2.68. Pada saat proses pelatihan berjalan, tensorflow membuat berkas *checkpoint* dengan otomatis. Berkas *checkpoint* berisi informasi selama proses pelatihan berjalan. Hasil berkas *checkpoint* saat pelatihan selesai dapat ditunjukkan pada Gambar 4.11.

```
model_checkpoint_path: "ckpt-11"
all_model_checkpoint_paths: "ckpt-5"
all_model_checkpoint_paths: "ckpt-6"
all_model_checkpoint_paths: "ckpt-7"
all_model_checkpoint_paths: "ckpt-8"
all_model_checkpoint_paths: "ckpt-9"
all_model_checkpoint_paths: "ckpt-10"
all_model_checkpoint_paths: "ckpt-11"
all_model_checkpoint_timestamps: 1681464442.4761515
all_model_checkpoint_timestamps: 1681464528.9171548
all_model_checkpoint_timestamps: 1681464616.3928597
all_model_checkpoint_timestamps: 1681464703.3053737
all_model_checkpoint_timestamps: 1681464789.203698
all_model_checkpoint_timestamps: 1681464873.2706254
all_model_checkpoint_timestamps: 1681464957.4961581
last_preserved_timestamp: 1681464062.9583693
```

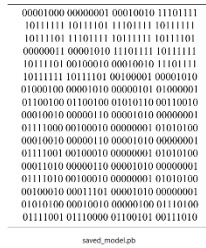
Gambar 4. 11 Hasil Berkas Pelatihan Model

#### 4.2.7 Export Model Deteksi Objek

Berkas *checkpoint* diekspor menjadi model deteksi objek yang digunakan untuk pendeteksian. Parameter program yang perlu disiapkan terdapat direktori



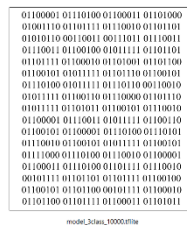
model ckeckpoint, letak file pipeline.config, dan direktori folder menyimpan hasil expor model. Hasil expor model deteksi objek dapat dilihat pada Gambar 4.12.



Gambar 4. 12 Hasil Berkas Expor Model

#### 4.2.8 Konversi Model Menjadi Tensorflow Lite

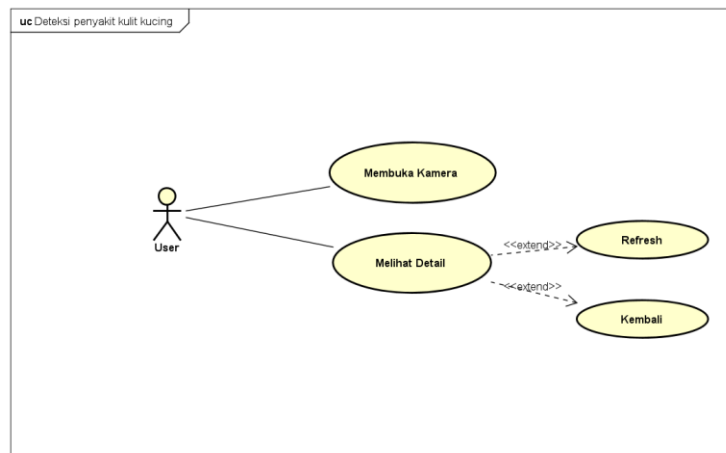
Setelah model deteksi objek selesai expor, selanjutnya melakukan konversi model menjadi tensorflow *lite*. Berkas yang dikonversi *saved\_model.pb*. Parameter program yang diperlukan letak file *saved\_model.pb*, dan direktori menyimpan file. Hasil konversi model deteksi objek menjadi tensorflow *lite* dapat dilihat pada Gambar 4.13.



Gambar 4. 13 Hasil Konversi Model Deteksi Tensorflow Lite

## 4.2 Desain Sistem Dan Aplikasi

### 4.2.1 Use Case Diagram



Gambar 4. 14 Use Case Diagram

Pada Gambar 4.14 *use case* diagram menunjukkan sebuah aktivitas dalam sistem yang dimana terdapat aktor yang berperan sebagai *user*. Aktivitas *user* yang dilakukan terdapat membuka kamera, dan melihat detail. Ketika *user* melakukan aktivitas detail, maka di dalam aktivitas detail terdapat fungsi *refresh*, dan kembali.

#### 4.2.2 Hasil Desain

##### 1. Hasil Desain Tampilan Menu Utama



Gambar 4. 15 Hasil Desain Tampilan Menu Utama

Berdasarkan Gambar 4.15 hasil desain tampilan menu utama terdapat tombol buka kamera yang berfungsi untuk menampilkan kamera *mobile*.

##### 2. Hasil Desain Tampilan Kamera

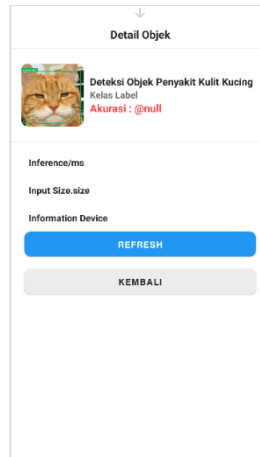


Gambar 4. 16 Hasil Desain Tampilan Kamera

Gambar 4.16 hasil desain tampilan kamera menampilkan kamera pada *mobile* yang digunakan sehingga *user* dapat mendeteksi penyakit kulit

kucing. Terdapat dua tombol detail dan cek yang dimana pada tombol detail untuk menampilkan halaman detail. Sementara tombol cek untuk melihat informasi nilai atau angka model deteksi saat melakukan proses ekstraksi yang dimunculkan pada cek sumber.

### 3. Hasil Desain Tampilan Detail

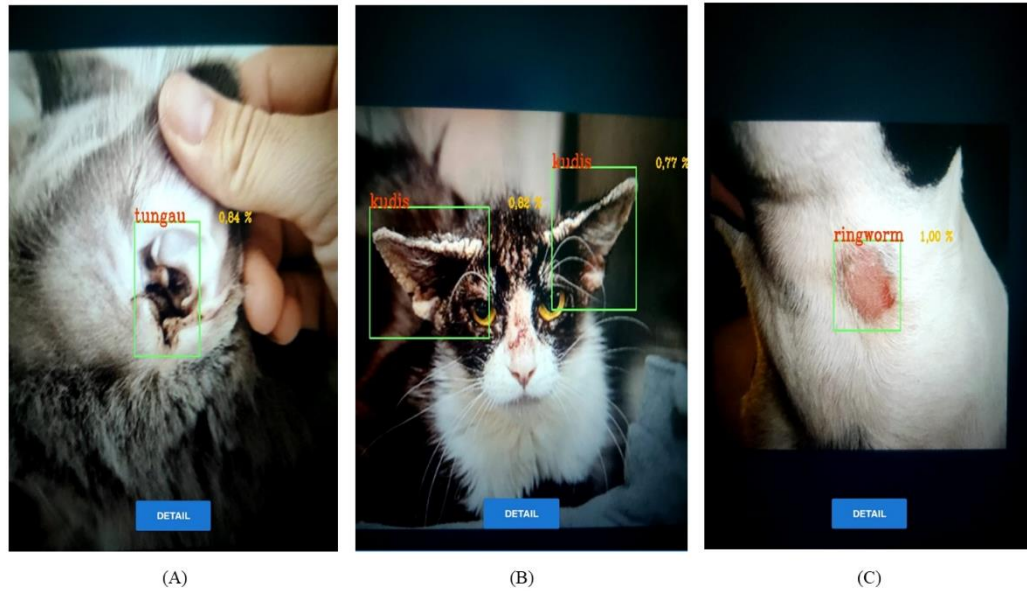


Gambar 4. 17 Hasil Desain Tampilan Detail

Berdasarkan Gambar 4.17 hasil desain tampilan detail menampilkan informasi informasi terkait deteksi objek. Informasi tersebut seperti nilai *inference*, *input size*, nama model yang digunakan, dan informasi perangkat.

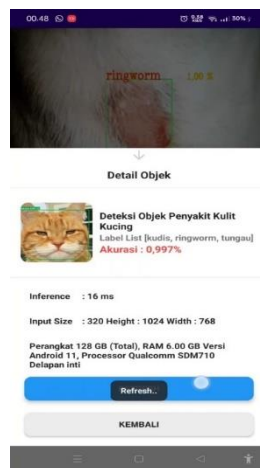
### 4.3 Implementasi Model

Model deteksi objek yang sudah dikonversi diterapkan ke dalam *mobile* dengan memberikan parameter nama folder, nama label, dan mengisi *input size*. File model yang digunakan dengan *step* 10000 karena memiliki nilai mAP tinggi dan mAR yang cukup tinggi. Nama label berupa file txt yang disamakan dengan kelas objek yang dideteksi. Objek penyakit kulit kucing yang terdeteksi ditunjukkan melalui pelabelan bentuk kotak warna hijau. Kotak berwarna hijau menandakan lokasi dari objek yang terdeteksi. Sementara pada atas kotak terdapat teks yang menunjukkan kelas label yang dideteksi. Model deteksi objek mengembalikan nilai probabilitasnya berupa nilai persentase. Nilai persentase digunakan untuk menentukan *threshold* objek penyakit kulit kucing ditampilkan atau tidak. Model hasil penelitian sudah dapat digunakan pada aplikasi deteksi penyakit kulit kucing kucing seperti pada Gambar 4.18.



Gambar 4. 18 (A) Deteksi Tungau (B) Deteksi Kudis (C) Deteksi *Ringworm*

Pada Gambar 4.18 menunjukkan hasil deteksi objek penyakit kulit kucing kudis, *ringworm*, dan tungau. Nilai tertinggi akurasi saat mendeteksi memperoleh 1.00% ketika mendeteksi *ringworm*, sedangkan kudis 0.83%, dan tungau 0.84%. Pada halaman tampilan detail menunjukkan hasil nilai yang ditunjukkan pada saat melakukan deteksi penyakit kulit kucing melalui tampilan kamera. Nilai akurasi persentase ditampilkan dengan cara menekan tombol *refresh* saat menangkap objek. *Inference* berupa nilai saat proses eksekusi model tensorflow *lite* untuk membuat prediksi berdasarkan data masukan. Nilai *input size* menunjukkan angka *height* dan *width* yang digunakan. Tampilah halaman hasil detail deteksi penyakit kulit kucing dapat ditunjukkan pada Gambar 4.19.



Gambar 4. 19 Tampilah Hasil Halaman Detail

#### 4.4 Whitebox Testing

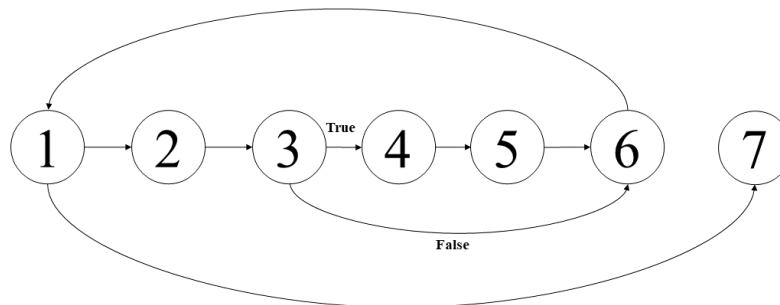
##### 4.4.1 Fungsi Mendeteksi Objek Penyakit Kulit Kucing

Fungsi mendeteksi objek penyakit kulit kucing digunakan untuk menerima masukan kamera yang berfungsi untuk mendeteksi objek. Ketika mendeteksi, tampilan pada kamera terdapat kotak-kotak berwarna hijau dan tulisan teks diatas sebagai nama kelas objek. Kode program dapat dilihat pada Tabel 4.2.

Tabel 4. 2 Kode Fungsi Deteksi Objek Penyakit Kulit Kucing

No	Kode Program
1	for (int i=0;i<10;i++){
2	class_value=(float) Array.get(Array.get(Object_class,0),i); score_value= (float) Array.get(Array.get(score,0),i);  xscore_value= String.format("%.2f",score_value);
3	if (score_value>0.5){
4	Object box1=Array.get(Array.get(value,0),i);  float top = (float) Array.get(box1,0)*height; float left= (float) Array.get(box1,1)*width; float bottom =(float) Array.get(box1,2)*height; float right =(float) Array.get(box1,3)*width;
5	Imgproc.rectangle(rotated_mat_image,new Point(left,top),new Point(right,bottom),new Scalar(100,255,100),2);  Imgproc.putText(rotated_mat_image,labelList.get((int)class_value) , new Point(left,top),3,1,new Scalar(250,60,0),2); Imgproc.putText(rotated_mat_image," "+xscore_value+" %", new Point(right,top),3,0.7,new Scalar(250,200,0),2,Imgproc.LINE_4);
6	} end kondisi if
7	} end kondisi for

Hasil penelitian ini menunjukkan fungsi aplikasi berjalan sesuai logika, ketika terjadi *error* sistem secara otomatis melakukan *force close* pada aplikasi. Berdasarkan kode program pada Tabel 4.2 yang berfungsi mendeteksi objek, maka dibuat *flow graph* seperti pada Gambar 4.20.



Gambar 4. 20 Flow Graph

Pada Gambar 4.20 *cyclomatic* pada *flowchart node* mendeteksi penyakit kulit kucing, proses tersebut mempunyai 8 jumlah busur (E), dan 7 jumlah simpul (N). Maka untuk *complexity* dan *path* dapat dihitung :

1. *Complexity*

Diketahui

Jumlah busur (E) : 8

Jumlah simpul (N) : 7

Jawab :

$$\begin{aligned}
 V(G) &= E - N + 2 \\
 &= 8 - 7 + 2 \\
 &= 3
 \end{aligned}$$

Sehingga jumlah *path* sebanyak 3 :

- Jalur 1 : 1-7
- Jalur 2 : 1-2-3-4-5-6-1-7
- Jalur 3 : 1-2-3-6-1-7

2. *Path*

Diketahui

Jumlah node simpul/bercabang (P) : 1, dan 1-2-3

Jawab :

$$\begin{aligned}
 V(G) &= P + 1 \\
 &= 2 + 1 \\
 &= 3
 \end{aligned}$$

#### 4.5 Blackbox Testing

Pada penelitian ini, pengujian dilakukan oleh dokter hewan Drh. Fyrr F. H TA yang memiliki pengetahuan tentang penyakit kulit pada kucing. Pengujian *blackbox testing* berdasarkan menu aktivitas yang terdapat pada aplikasi dilakukan dengan menguji beberapa tombol dan mendeteksi objek. Pada tampilan utama terdapat tombol buka kamera, tampilan kamera terdapat tombol detail, tampilan detail terdapat tombol *refresh*, dan tombol kembali.

Tabel 4. 3 Tabel Pengujian *Blackbox Testing*

No	Deskripsi Pengujian	Hasil yang diharapkan	Hasil Pengujian	Keterangan
1.	Menekan tombol kamera	Sistem menampilkan kamera dan mendeteksi objek penyakit kulit kucing	Sesuai	Valid
2.	Menekan tombol detail	Sistem menampilkan halaman detail	Sesuai	Valid
3.	Menekan tombol <i>refresh</i>	Sistem menampilkan informasi detail nilai akurasi, <i>inference</i> , <i>input size</i> , dan informasi perangkat	Sesuai	Valid
4.	Menekan tombol kembali	Sistem menampilkan halaman kamera	Sesuai	Valid

#### 4.6 Evaluasi Hasil

Model deteksi objek metode *deep learning* algoritma CNN membutuhkan *dataset* yang sudah terdapat pelabelan berupa *bouding box* dan kelas objeknya. Berdasarkan *dataset* yang dikumpulkan masih terbatas sehingga perlu diperbanyak untuk meningkat performa model yang lebih baik. Jumlah *dataset* yang dikumpulkan sebanyak 43 citra penyakit kulit kucing, namun data tersebut masih kurang. Hasil model deteksi objek metode *deep learning* algoritma CNN dengan arsitektur *MobileNetV2 FPN Lite 320x320* berhasil diterapkan. Hal tersebut ditunjukkan pada hasil mAP tinggi dengan *step* 10000 bernilai 42% dan berhenti meningkat ketika *step* 100000 bernilai 20%. Proses implementasi model deteksi objek pada aplikasi *mobile* dapat berjalan dengan baik. Melalui uji coba menggunakan aplikasi *mobile*, proses mendeteksi objek penyakit kulit kucing membutuhkan *inference* yang lebih sedikit. Evaluasi sistem dilakukan bersama dengan pakar dokter hewan Drh. Fyarr F. H TA yang memiliki pengetahuan tentang penyakit kulit pada kucing. Melalui penelitian ini, evaluasi hasil sistem menggunakan sebuah *input* citra kucing yang terjangkit penyakit kulit seperti kudis, *ringworm*, dan tungau. Saat sistem melakukan deteksi terhadap *input* citra, maka hasil sistem memberikan *output* hasil citra dan memberikan keterangan pada sistem (lampiran). Pada penelitian ini, proses validasi hasil dilakukan dengan mengetahui hasil penilaian melalui validasi pakar. Terdapat 2 data Setuju (S) dan Tidak Setuju (TS) melalui validasi pakar saat menyesuaikan hasil dari *output* dan keterangan pada hasil evaluasi sistem (lampiran). Tabel 4.4 menunjukkan validasi hasil pakar.

Tabel 4. 4 Tabel Validasi Pakar

No	Nama Penyakit	Hasil Pakar	Keterangan
1	Kudis	Mendeteksi penyakit kudis	Sesuai, mendeteksi penyakit kudis
2	Kudis	Mendeteksi penyakit kudis	Sesuai, mendeteksi penyakit kudis
3	Kudis	Mendeteksi penyakit <i>ringworm</i>	Tidak Sesuai, seharusnya penyakit kudis
4	Kudis	Mendeteksi penyakit kudis	Sesuai, mendeteksi penyakit kudis



5	Kudis	Mendeteksi penyakit kudis	Sesuai, mendeteksi penyakit kudis
6	<i>Ringworm</i>	Mendeteksi penyakit <i>ringworm</i>	Sesuai, mendeteksi penyakit <i>ringworm</i>
7	<i>Ringworm</i>	Mendeteksi penyakit <i>ringworm</i>	Sesuai, mendeteksi penyakit <i>ringworm</i>
8	<i>Ringworm</i>	Mendeteksi penyakit <i>ringworm</i>	Sesuai, mendeteksi penyakit <i>ringworm</i>
9	<i>Ringworm</i>	Mendeteksi penyakit <i>ringworm</i>	Sesuai, mendeteksi penyakit <i>ringworm</i>
10	<i>Ringworm</i>	Mendeteksi penyakit <i>ringworm</i>	Sesuai, mendeteksi penyakit <i>ringworm</i>
11	Tungau	Mendeteksi penyakit tungau	Sesuai, mendeteksi penyakit tungau
12	Tungau	Mendeteksi penyakit <i>ringworm</i>	Tidak Sesuai, seharusnya penyakit tungau
13	Tungau	Mendeteksi penyakit tungau	Sesuai, mendeteksi penyakit tungau
14	Tungau	Mendeteksi penyakit <i>ringworm</i>	Tidak Sesuai, seharusnya penyakit tungau
15	Tungau	Mendeteksi penyakit <i>ringworm</i>	Tidak Sesuai, seharusnya penyakit tungau

Diketahui :

Sesuai (S) : 11

Tidak Sesuai (TS) : 4

$$Validasi Pakar = \frac{S}{S + TS} \times 100\%$$

$$Validasi Pakar = \frac{11}{11 + 4} \times 100\%$$

$$Validasi Pakar = 73\%$$

Berdasarkan perhitungan yang dilakukan maka hasil validasi pakar pada penelitian ini sebesar 73%.

## **BAB V**

### **KESIMPULAN DAN SARAN**

#### **5.1 Kesimpulan**

Berdasarkan hasil penelitian yang sudah dilakukan, sehingga diperoleh beberapa kesimpulan.

1. Model deteksi objek menggunakan metode *deep learning* algoritma CNN dengan arsitektur *MobileNetV2 FPN Lite* dapat dilakukan. Model deteksi objek penyakit kucing memerlukan *dataset* yang sudah diberi pelabelan citra berupa *bounding box* beserta dengan nama kelas objeknya.
2. Hasil pembuatan aplikasi *mobile* berbasis android, dan penerapan model *tensorflow lite* pada aplikasi *mobile* berbasis android dalam mendeteksi penyakit kulit kucing berhasil diterapkan.
3. Hasil evaluasi model metode *deep learning* algoritma CNN dengan arsitektur *MobileNetV2 FPN Lite* berhasil dilakukan. Hal tersebut ditunjukkan melalui nilai mAP dari model sebesar 42% dan mAR sebesar 23% ketika *step* 10000. Hasil evaluasi sistem dan validasi pakar melalui perhitungan dengan 2 data Setuju (S) dan Tidak Setuju (TS) sebesar 73%.

#### **5.2 Saran**

Melalui penelitian yang telah dilaksanakan, terdapat beberapa saran yang diberikan.

1. Jumlah *dataset* yang digunakan pelatihan model deteksi objek penyakit kulit kucing perlu diperbanyak untuk meningkatkan performa model.
2. Model deteksi objek penyakit kulit kucing dapat dikembangkan lebih lanjut untuk aplikasi *mobile* lainnya seperti iOS dan berbasis *website*.

## DAFTAR PUSTAKA

- Arifianto, J. (2022). *APLIKASI WEB PENDETEKSI JERAWAT PADA WAJAH MENGGUNAKAN MODEL DEEP LEARNING DENGAN TENSORFLOW*.
- Dharmaputra, A., Cahyanti, M., Septian, M. R. D., & Swedia, E. R. (2021). Aplikasi Face Mask Detection Menggunakan Neural Network Mobilenetv2 Berbasis Android. *Sebatik*, 25(2), 382–389. <https://doi.org/10.46984/sebatik.v25i2.1503>
- Dwiramadhan, F., Wahyuddin, M. I., & Hidayatullah, D. (2022). Sistem Pakar Diagnosa Penyakit Kulit Kucing Menggunakan Metode Naive Bayes Berbasis Web. *Jurnal JTIK (Jurnal Teknologi Informasi Dan Komunikasi)*, 6(3), 429–437. <https://doi.org/10.35870/jtik.v6i3.466>
- Felix, F., Wijaya, J., Sutra, S. P., Kosasih, P. W., & Sirait, P. (2020). Implementasi Convolutional Neural Network Untuk Identifikasi Jenis Tanaman Melalui Daun. *Jurnal SIFO Mikroskil*, 21(1), 1–10. <https://doi.org/10.55601/jsm.v21i1.672>
- Fidyaningsih, S., Agus, F., & Maharani, S. (2017). *Sistem Pakar Diagnosa Penyakit Kucing Menggunakan Metode Forward Chaining (Fc) Berbasis Web*. 1–27.
- Harani, H., Prianto, N., & Cahyo Hasanah, M. (2019). Deteksi Objek Dan Pengenalan Karakter Plat Nomor Kendaraan Indonesia Menggunakan Metode Convolutional Neural Network (CNN) Berbasis Python. *Jurnal Teknik Informatika*, 11(3), 47–53.
- Harijanto, B., & Latif, R. A. (2016). Sistem Pakar Diagnosa Penyakit Pada Kucing Dengan Metode Teorema Bayes Berbasis Android. *Jurnal Informatika Polinema*, 2(4), 176. <https://doi.org/10.33795/jip.v2i4.79>
- How, T. (2021). *Metode SDLC*. Kumparan. <https://kumparan.com/how-to-teknologi/metode-sdlc-definisi-manfaat-dan-jenis-modelnya-1zqTZD1AzKH>
- Images, G. (2022). *Google Image Dataset*. Google Images. <https://images.google.com/>
- Ishak, A., & Pakaya, N. (2021). Sistem Informasi Wedding Organizer Berbasis Android. *Jambura Journal of Informatics*, 3(2), 97–108. <https://doi.org/10.37905/jji.v3i2.11746>


- Khoiruddin, M., Junaidi, A., & Saputra, W. A. (2022). Klasifikasi Penyakit Daun Padi Menggunakan Convolutional Neural Network. *Journal of Dinda : Data Science, Information Technology, and Data Analytics*, 2(1), 37–45. <https://doi.org/10.20895/dinda.v2i1.341>
- Munantri, N. Z., Sofyan, H., & Florestiyanto, M. Y. (2020). Aplikasi Pengolahan Citra Digital Untuk Identifikasi Umur Pohon. *Telematika*, 16(2), 97. <https://doi.org/10.31315/telematika.v16i2.3183>
- Nasyuha, A. H., Arif, S. N., Zunaidi, M., Yanti, N., & Gaol, L. (2022). Implementasi Algoritma K-Nearest Neighbor Dalam Mendiagnosis Kurap Pada Kucing. 4(1), 61–65. <https://doi.org/10.47065/josyc.v4i1.2479>
- Nurajizah, S., & Saputra, M. (2018). *SISTEM PAKAR BERBASIS ANDROID UNTUK DIAGNOSA PENYAKIT*. 14(1), 7–14.
- Nurhikmat, T. (2020). Implementasi Deep Learning untuk Image Classification Menggunakan Algoritma Convolutional Neural Network (CNN) Pada Citra Wayang Golek. *Universitas Islam Indonesia*, June. <https://doi.org/10.13140/RG.2.2.10880.53768>
- Patria, H., Anton, A., & Astuti, P. (2021). Sistem Pakar Menggunakan Metode Certainty Factor Untuk Mendiagnosa Penyakit Kulit Pada Hewan Kucing. *Simpatik: Jurnal Sistem Informasi Dan Informatika*, 1(1), 1–8. <http://103.75.24.116/index.php/simpatik/article/view/70>
- Permana, A. Y., & Romadlon, P. (2019). PERANCANGAN SISTEM INFORMASI PENJUALAN PERUMAHAN MENGGUNAKAN METODE SDLC PADA PT. MANDIRI LAND PROSPEROUS BERBASIS MOBILE. *Бухимия*, 84(10), 1511–1518. <https://doi.org/10.1134/s0320972519100129>
- Putri, P. (2022). *IDENTIFIKASI TUNGAU TELINGA (Otodectes cynotis) PADA KUCING DI KLINIK HEWAN VETOPET 5 PANDU RAYA BOGOR*.
- Rosalina, R., & Wijaya, A. (2020). Pendeteksian Penyakit pada Daun Cabai dengan Menggunakan Metode Deep Learning. *Jurnal Teknik Informatika Dan Sistem Informasi*, 6(3), 452–461. <https://doi.org/10.28932/jutisi.v6i3.2857>
- Setyawan, D. M. B., Haryoko, A., Nurlifa, A., & Suryanto, A. A. (2021). *Sistem Pakar Diagnosa Penyakit Kucing Dengan Naïve Bayes*. 2(1), 37–46.
- Sudirman, sudirman, Dasan, A., & Fortuna, A. (2022). *Deteksi penyakit kulit pada*

*kucing dengan metode forward chaining berbasis android.*

- Susanto, H., Kartikaningrum, M., Wahjuni, R., Warsito, S., & Yuliani, G. (2020). Kasus Scabies (*Sarcoptes Scabiei*) Pada Kucing Di Klinik Intimedipet Surabaya. *Jurnal Biosains Pascasarjana*, 22(1), 37. <https://doi.org/10.20473/jbp.v22i1.2020.37-45>
- Syah, D. (2022). *Mean Average Precision*. V7labs. <https://www.v7labs.com/blog/mean-average-precision>
- Syaikhoni, A., & Ariyadi, A. (2018). *Deteksi Objek Deteksi API Tensorflow*. MTI Binus. <https://mti.binus.ac.id/2018/12/26/deteksi-objek-dengan-tensorflow-object-detection-api/>
- Tensorflow. (2022). *Tensorflow Model Zoo*. Tensorflow. <https://github.com/tensorflow/models.git>
- Widyaningsih, M., & Gunadi, R. (2017). Dempster Shafer Untuk Sistem Diagnosa Gejala Penyakit Kulit Pada Kucing. *Jurnal SAINTEKOM*, 7(1), 81. <https://doi.org/10.33020/saintekom.v7i1.24>
- Wilianto, K. (2021). *Evaluasi Metrik Pada Computer Vision*. Data Folks Indonesia. <https://medium.com/data-folks-indonesia/evaluation-metrics-pada-computer-vision-dari-klasifikasi-hingga-deteksi-objek-5049d3fd90d2>

## LAMPIRAN

### Lampiran 1 Form Bimbingan Proposal Tugas Akhir













**UNIVERSITAS BUANA PERJUANGAN KARAWANG**  
**FAKULTAS ILMU KOMPUTER**  
**Terakreditasi BAN-PT**  
 Jl. H.S. Ronggowaluyo Telukjambe Timur Karawang 41161 Telp./Fax. (0267) 8403140  
 Site: <http://fik.ubpkarawang.ac.id> email: [fik@ubpkarawang.ac.id](mailto:fik@ubpkarawang.ac.id)

---

F02/TA/2022



**LEMBAR BIMBINGAN PROPOSAL TUGAS AKHIR**

Nama Mahasiswa : Rizandi Aditya Fikrah  
 NIM : 12416255201185 Program Studi : Teknik Informatika  
 Pembimbing I : Anis Fitri Nur Masrumah, M.kom  
 Pembimbing II : Ayu Ratna Juwita, M.kom  
 Judul Proposal : DETEKSI PENYAKIT KULIT KUCING MENGGUNAKAN ALGORITMA CNN DENGAN TENSORFLOWLITE BERBASIS ANDROID


No.	Tanggal Bimbingan	Target Bimbingan	Hasil Bimbingan dan Rencana Selanjutnya	Paraf Pembimbing
1	09/11/2022	TOPIC	Sudah menentukan topik magang dan lanjut Bab I	
2	10/11/2022	Mengembangkan bab I, dan bab II	Revisi Bab I dan Bab II	
3	18/11/2022	Mengembangkan revisi Bab I dan II	Revisi Bab I dan II	
4	18/11/2022	Mengembangkan revisi Bab I dan II	Revisi Bab I	
5	21/11/2022	Mengembangkan Bab I revisi	Melanjutkan Bab III	
6	25/11/2022	Mengembangkan Bab III	Revisi Bab III	
7	23/11/2022	Mengembangkan Bab III	Revisi Bab III	
8	07/12/2022	Mengembangkan Bab III	Revisi Bab III	
9	23/12/2022	Mengembangkan Bab III	Revisi Bab III	
10	23/12/2022	Mengembangkan Bab III		

**Catatan :**  
 Formulir ini diisi setiap bimbingan dan dilampirkan saat pendaftaran seminar proposal TA dengan minimal 8 kali bimbingan.

**Rekomendasi Mengikuti Seminar Proposal Tugas Akhir**

<b>Pembimbing I,</b>  Nama : Tanggal :	<b>Pembimbing II,</b>  Nama : Tanggal :
--	---

## Lampiran 2 Lembar Perbaikan Penguji



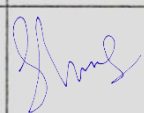

**UNIVERSITAS BUANA PERJUANGAN KARAWANG**  
**FAKULTAS ILMU KOMPUTER**  
**Terakreditasi BAN-PT**  
 Jl. H.S. Ronggowaluyo Telukjambe Timur Karawang 41361 Telp./Fax. (0267) 8403140  
 Site: <http://fik.ubpkarawang.ac.id> email: [fik@ubpkarawang.ac.id](mailto:fik@ubpkarawang.ac.id)

---

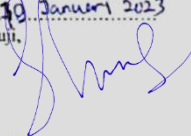
SPT-03/MP-01/SOP-220-217-02

**LEMBAR PERBAIKAN PENGUJI**  
**SEMINAR PROPOSAL TUGAS AKHIR**

<ol style="list-style-type: none"> <li>1. Nama Mahasiswa</li> <li>2. NIM</li> <li>3. Program Studi</li> <li>4. Judul Proposal Tugas Akhir</li> </ol>	<p>: <u>Rizandi Aditya Fitrah</u></p> <p>: <u>19116255201105</u></p> <p>: <u>Teknik Informatika</u></p> <p>: <u>Deteksi Penyakit Kulit Leung Menggunakan</u>  <u>Algoritma CNN Dengan Tersertifikasi Utl</u>  <u>Berbasis Android</u></p>
--	---

No.	Perbaikan	Paraf
1	Rumusan masalah ditambahkan kata presisi	
2.	Dibuatkan narasi untuk gambar flowchart 3.5 Alur sistem	

Demikian untuk dipergunakan sebagaimana mestinya.

Karawang, 10 Januari 2023  
Ketua Penguji,   
( Sutan Faisal )



**UNIVERSITAS BUANA PERJUANGAN KARAWANG**  
**FAKULTAS ILMU KOMPUTER**  
**Terakreditasi BAN-PT**

Jl. H.S. Ronggowaluyo Telukjambe Timur Karawang 41361 Telp./Fax. (0267) 8403140  
 Site: <http://fik.ubpkarawang.ac.id> email: [fik@ubpkarawang.ac.id](mailto:fik@ubpkarawang.ac.id)

SPT-03/MP-01/SOP-220-217-02

**LEMBAR PERBAIKAN PENGUJI**  
**SEMINAR PROPOSAL TUGAS AKHIR**

- |                               |   |
|-------------------------------|---|
| 1. Nama Mahasiswa             | : Puwandi Aditya Filrah   |
| 2. NIM                        | : 10411255201105  |
| 3. Program Studi              | : Teknik Informatika  |
| 4. Judul Proposal Tugas Akhir | : Deteksi Penyakit Kulit Kucing Menggunakan Algoritma CNN Dengan TensorFlow Lite Berbasis Android |

No.	Perbaikan	Paraf
1.	Perbaikan sesuai catatan pada lap. proposal yang diterima.	
2.	Perbaikan sesuai arahan penguji	

Demikian untuk dipergunakan sebagaimana mestinya.

Karawang, 19 Januari 2023  
 Anggota Penguji,

Anis Fit Murningsih



## Lampiran 3 Dokumentasi Wawancara Pertama



Lampiran 4 Konfigurasi *Pipeline*





Konfigurasi	Isi	Keterangan
Num_classes	3	Jumlah kelas objek yang dideteksi
Batch_size	4-8	Jumlah sampel data yang dieksekusi
Fine_tune_checkpoint	model/checkpoint/ckpt-0	Lokasi berkas ckpt dari model
Fine_tune_checkpoint_type	Detection	Tipe model yang digunakan
Label_map_path	Annotations/labelmap.pbtxt	Lokasi berkas <i>labelmap</i>
Input_path( <i>train</i> )	Annotations/train.record	Lokasi berkas TF <i>Record</i> data <i>train</i>
Input_path( <i>test</i> )	Annotations/test.record	Lokasi berkas TF <i>Record</i> data <i>test</i>


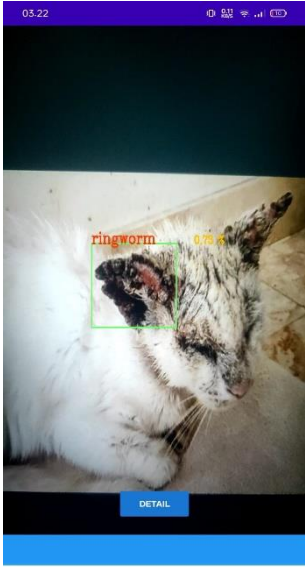

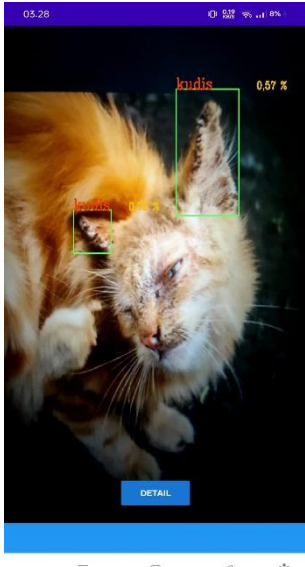
## Lampiran 5 Dokumentasi Wawancara Kedua




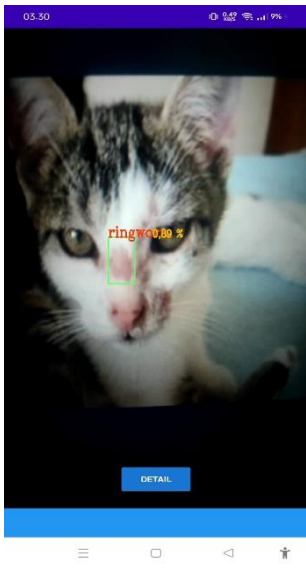



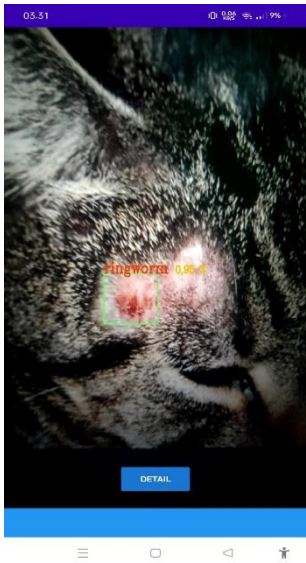





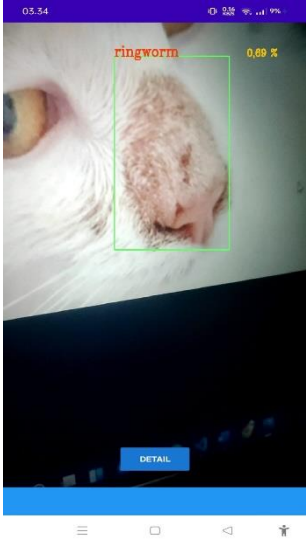


## Lampiran 6 Evaluasi Sistem

No	Input Citra (Kudis, Ringworm, dan Tungau)	Hasil Sistem		Validasi Pakar	Tanda Tangan Pakar
		Output	Keterangan		
1			Sistem mendeteksi penyakit kulit kucing objek kudis <i>bounding box</i> hijau	Sesuai	
2			Sistem mendeteksi penyakit kulit kucing objek kudis <i>bounding box</i> hijau	Sesuai	


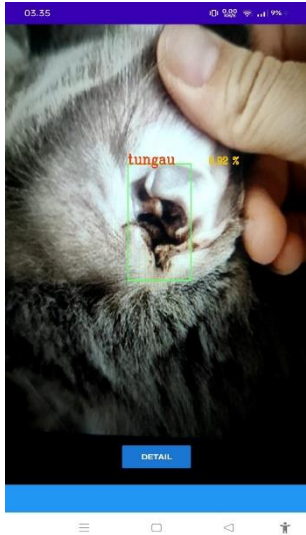

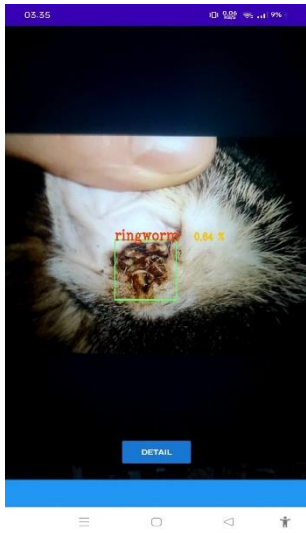
No	Input Citra (Kudis, Ringworm, dan Tungau)	Hasil Sistem		Validasi Pakar	Tanda Tangan Pakar
		Output	Keterangan		
3			Sistem mendeteksi penyakit kulit kucing objek <i>ringworm</i> bounding box hijau	Tidak Sesuai, seharusnya mendeteksi penyakit kulit kucing objek kudis	
4			Sistem mendeteksi penyakit kulit kucing objek kudis bounding box hijau	Sesuai	



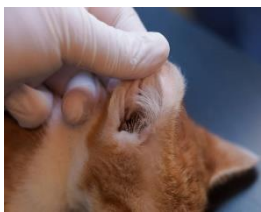

No	Input Citra (Kudis, Ringworm, dan Tungau)	Hasil Sistem		Validasi Pakar	Tanda Tangan Pakar
		Output	Keterangan		
5			Sistem mendeteksi penyakit kulit kucing objek kudis <i>bounding box</i> hijau	Sesuai	
6			Sistem mendeteksi penyakit kulit kucing objek ringworm <i>bounding box</i> hijau	Sesuai	


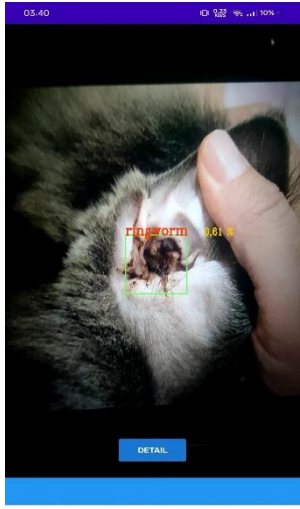
No	Input Citra (Kudis, Ringworm, dan Tungau)	Hasil Sistem		Validasi Pakar	Tanda Tangan Pakar
		Output	Keterangan		
7			Sistem mendeteksi penyakit kulit kucing objek <i>ringworm</i> bounding box hijau	Sesuai	
8			Sistem mendeteksi penyakit kulit kucing objek <i>ringworm</i> bounding box hijau	Sesuai	

No	Input Citra (Kudis, Ringworm, dan Tungau)	Hasil Sistem		Validasi Pakar	Tanda Tangan Pakar
		Output	Keterangan		
9			Sistem mendeteksi penyakit kulit kucing objek <i>ringworm</i> bounding box hijau	Sesuai	
10			Sistem mendeteksi penyakit kulit kucing objek <i>ringworm</i> bounding box hijau	Sesuai	

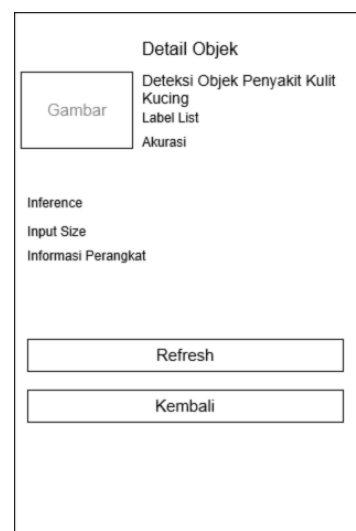
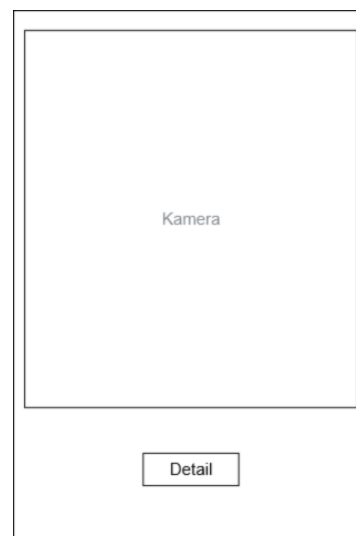


No	Input Citra (Kudis, Ringworm, dan Tungau)	Hasil Sistem		Validasi Pakar	Tanda Tangan Pakar
		Output	Keterangan		
11			Sistem mendeteksi penyakit kulit kucing objek tungau <i>bounding box</i> hijau	Sesuai	
12			Sistem mendeteksi penyakit kulit kucing objek <i>ringworm bounding box</i> hijau	Tidak Sesuai, seharusnya mendeteksi penyakit kulit kucing objek tungau	

No	Input Citra (Kudis, Ringworm, dan Tungau)	Hasil Sistem		Validasi Pakar	Tanda Tangan Pakar
		Output	Keterangan		
13			Sistem mendeteksi penyakit kulit kucing objek tungau <i>bounding box</i> hijau	Sesuai	
14			Sistem mendeteksi penyakit kulit kucing objek <i>ringworm</i> <i>bounding box</i> hijau	Tidak Sesuai, seharusnya mendeteksi penyakit kulit kucing objek tungau	

No	Input Citra (Kudis, Ringworm, dan Tungau)	Hasil Sistem		Validasi Pakar	Tanda Tangan Pakar
		Output	Keterangan		
15			Sistem mendeteksi penyakit kulit kucing objek ringworm bounding box hijau	Tidak Sesuai, seharusnya mendeteksi penyakit kulit kucing objek tungau	
$Validasi Pakar = \frac{S}{S + TS} \times 100\%$ $Validasi Pakar = \frac{11}{11 + 4} \times 100\%$ $Validasi Pakar = \frac{11}{15} \times 100\%$ $Validasi Pakar = 73\%$ <p>Dimana :</p> <p>S = Sesuai</p> <p>TS = Tidak Sesuai</p>					

## Lampiran 7 Desain UX Aplikasi Deteksi



## Lampiran 8 Perintah dan Program Model Tensorflow 2.5.0

```
#Library
!python -version
!python -m pip install --upgrade pip==20.3
!pip install tensorflow==2.5.0
!pip install tensorflow-gpu==2.8.0
import tensorflow as tf
if tf.test.gpu_device_name():
    print('Default GPU Device:
{}'.format(tf.test.gpu_device_name()))
else:
    print("Please install GPU version of TF")
import tensorflow as tf
print(tf.__version__)

pwd
cd /content/models/research

!protoc object_detection/protos/*.proto --
python_out=.

#Menyalin Data Coco
!git clone https://github.com/cocodataset/cocoapi.git
cd cocoapi/PythonAPI

!make

cp -r pycocotools /content/models/research
pwd
cd ..
cd ..

cp object_detection/packages/tf2/setup.py .

!python -m pip install --use-feature=2020-resolver .

!pip uninstall keras
!pip install keras --upgrade
```

#Tes Instal <i>Library</i>
!python object_detection/builders/model_builder_tf2_test.py
#Menyalin Data GitHub
cd /content/  !git clone <a href="https://github.com/riyandifitrah/training_demo.git">https://github.com/riyandifitrah/training_demo.git</a>
cd /content/training_demo/pre-trained-models/
#Mengunduh Data
!wget <a href="http://download.tensorflow.org/models/object_detection/tf2/20200711/ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8.tar.gz">http://download.tensorflow.org/models/object_detection/tf2/20200711/ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8.tar.gz</a>
#Transfer Data Arsitektur
!tar -xvf ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8.tar.gz
cd /content/training_demo
#Membuat Tensorflow <i>Record</i>
!python generate_tfrecord.py -x /content/training_demo/images/train -l /content/training_demo/annotations/labelmap.pbtxt -o /content/training_demo/annotations/train.record
cd /content/training_demo  ls
#Melakukan Pelatihan Model
!python model_main_tf2.py -- model_dir=/content/training_demo/models/pengujian_12_100000 -- pipeline_config_path=/content/training_demo/models/pengujian_12_100000/pipeline.config
#Evaluasi Model Deteksi Objek
!python /content/training_demo/model_main_tf2.py -- model_dir=/content/5000 -- pipeline_config_path=/content/5000pipeline.config -- checkpoint_dir=/content/5000

```
#Expor Model

!python
/content/training_demo/export_tflite_graph_tf2.py --
pipeline_config_path /content/100000/pipeline.config
--trained_checkpoint_dir /content/100000/models/lite2
--output_directory /content/training_demo/ex_lite3
```

```
#Membuat File Zip dan Mengunduhnya

!zip -r /content/training_demo/models/lite2.zip
/content/training_demo/models/lite2

from google.colab import files

path_exported_models_zip="/content/training_demo/mod
els/lite2.zip"

files.download(path_exported_models_zip)
```

### Lampiran 9 Perintah dan Program Model Tensorflow 2.8.0

```
# Menyalin repositori tensorflow model dari GitHub
!git clone --depth 1
https://github.com/tensorflow/models

# Salin file setup ke folder model/penelitian
%%bash
cd models/research/
protoc object_detection/protos/*.proto --python_out=.

# Ubah file setup.py untuk menginstal repositori tf-
models-official yang ditargetkan pada TF v2.8.0
import re

with
open('/content/models/research/object_detection/packa
ges/tf2/setup.py') as f:

    s = f.read()

with open('/content/models/research/setup.py', 'w')
as f:

    # Set fine_tune_checkpoint path
    s = re.sub('tf-models-official>=2.5.1',
                'tf-models-official==2.8.0', s)

    f.write(s)

# Instal Object Detection API Tensorflow
!pip install /content/models/research/

# Perlu menurunkan versi ke TF v2.8.0 karena bug
kompatibilitas Colab dengan TF v2.10 (mulai 10/03/22)
!pip install tensorflow==2.8.0

#Tes Instal Library
!python
/content/models/research/object_detection/builders/mo
del_builder_tf2_test.py
```



#Menyalin Data Dari GitHub
!git clone https://github.com/riyandifitrah/training_demo.git
cd /content/training_demo/pre-trained-models/
#Mengunduh Data
!wget http://download.tensorflow.org/models/object_detection/tf2/20200711/ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8.tar.gz
#Transfer Data Arsitektur
!tar -xvf ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8.tar.gz
cd /content/training_demo
#Membuat Tensorflow Record
!python generate_tfrecord.py -x /content/training_demo/images/train -l /content/training_demo/annotations/labelmap.pbtxt -o /content/training_demo/annotations/train.record
cd /content/training_demo
ls
#Koneksi Google Drive
from google.colab import drive drive.mount('/content/drive')
#Melakukan Pelatihan Model
!python model_main_tf2.py -- model_dir=/content/drive/MyDrive/Tensorflow_TA_Pengujian/100000 -- pipeline_config_path=/content/drive/MyDrive/Tensorflow_TA_Pengujian/100000/pipeline.config
#Export Model
!python /content/training_demo/export_tflite_graph_tf2.py -- pipeline_config_path /content/100000/content/training_demo/models/100000/pipeline.config --trained_checkpoint_dir

```
/content/100000/content/training_demo/models/100000 -  
-output_directory  
/content/100000/content/training_demo/models/100000/1  
ite_100000
```

```
#Evaluasi Model
```

```
!python /content/training_demo/model_main_tf2.py --  
model_dir=/content/training_demo/models/pengujian10_1  
000 --  
pipeline_config_path=/content/training_demo/models/pe  
ngujian10_1000/pipeline.config --  
checkpoint_dir=/content/training_demo/models/pengujia  
n10_1000
```

```
#Tensorboard
```

```
%load_ext tensorboard
```

```
%tensorboard --logdir
```

```
'/content/drive/MyDrive/Tensorflow_TA_Pengujian/10000  
/content/training_demo/models/10000'
```

## Lampiran 10 Kode Program Implementasi Model

```
package com.example.object_detection;

import androidx.appcompat.app.AppCompatActivity;
import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import com.airbnb.lottie.LottieAnimationView;
import org.opencv.android.OpenCVLoader;

public class MainActivity extends AppCompatActivity {

    static {
        if(OpenCVLoader.initDebug()){
            Log.d("MainActivity: ", "Opencv Load");
        }
        else {
            Log.d("MainActivity: ", "Opencv Failed Load");
        }
    }

    private Button btn_camera;
    private LottieAnimationView lt1;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

```

//select device and run

lt1=findViewById(R.id.loti_camera);

lt1.loop(true);

btn_camera=findViewById(R.id.camera_button);

btn_camera.setOnClickListener(new View.OnClickListener() {

    @Override

    public void onClick(View view) {

        startActivity(new
Intent(MainActivity.this,CameraActivity.class).addFlags(Intent.FLAG_ACTIV
ITY_CLEAR_TASK | Intent.FLAG_ACTIVITY_CLEAR_TOP ));

    }

});

}

}

```

```

package com.example.object_detection;

import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;
import android.Manifest;
import android.app.Activity;
import android.content.pm.PackageManager;
import android.content.res.Resources;
import android.graphics.Camera;
import android.net.Uri;
import android.os.Bundle;
import android.os.Environment;
import android.util.Log;

```

```
import android.view.LayoutInflater;
import android.view.SurfaceView;
import android.view.View;
import android.view.Window;
import android.view.WindowManager;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.TextView;
import android.widget.Toast;
import com.google.android.material.bottomsheet.BottomSheetBehavior;
import com.google.android.material.bottomsheet.BottomSheetDialog;
import org.opencv.android.BaseLoaderCallback;
import org.opencv.android.CameraBridgeViewBase;
import org.opencv.android.LoaderCallbackInterface;
import org.opencv.android.OpenCVLoader;
import org.opencv.core.CvType;
import org.opencv.core.Mat;
import org.opencv.imgcodecs.Imgcodecs;
import org.opencv.imgproc.Imgproc;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.io.InputStream;
import java.lang.reflect.Array;
import java.text.SimpleDateFormat;
import java.util.Date;
```

```

public class CameraActivity extends Activity implements
CameraBridgeViewBase.CvCameraViewListener2 {

    private static final String TAG="MainActivity";

    private Mat mRgba;
    private Mat mGray;
    private int capture = 0;
    private CameraBridgeViewBase mOpenCvCameraView;
    Bundle intentExtras;
    BottomSheetBehavior bottomSheetBehavior;
    private EditText tv3;
    private objectDetectorClass objectDetectorClass;

    private BaseLoaderCallback mLoaderCallback=new
BaseLoaderCallback(this) {

        @Override
        public void onManagerConnected(int status) {
            switch (status){
                case LoaderCallbackInterface
                    .SUCCESS: {
                        Log.i(TAG,"Opencv is loaded");
                        mOpenCvCameraView.enableView();

                    }
                default: {
                    super.onManagerConnected(status);
                }
                break;
            }
        }
    }
}

```

```

};

public CameraActivity(){
    Log.i(TAG,"Instantiated"+this.getClass());
}

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    requestWindowFeature(Window.FEATURE_NO_TITLE);

getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);

    int MY_PERMISSIONS_REQUEST_CAMERA = 0;

    if (ContextCompat.checkSelfPermission(CameraActivity.this,
Manifest.permission.CAMERA)

        == PackageManager.PERMISSION_DENIED) {

        ActivityCompat.requestPermissions(CameraActivity.this, new
String[]{Manifest.permission.CAMERA},
MY_PERMISSIONS_REQUEST_CAMERA);

    }

    setContentView(R.layout.activity_camera);
//    Button btn_sheetnested=(Button) findViewById(R.id.btn_expand);
    View view_nested = findViewById(R.id.bottom_nested);
    Uri path = Uri.parse("file:///android_asset/model_2class.tflite");

    String newPath = path.toString();

    bottomSheetBehavior =BottomSheetBehavior.from(view_nested);
//    btn_sheetnested.setOnClickListener(new View.OnClickListener() {

```

```

//      @Override
//      public void onClick(View view) {
//
bottomSheetBehavior.setState(BottomSheetBehavior.STATE_EXPANDED);
//
////      TextView tv_nested = (TextView)findViewById(R.id.txt_nested);
//
////
tv_nested.setText(""+objectDetectorClass.inferencetime+objectDetectorClass.w
idth+objectDetectorClass.height+objectDetectorClass.INPUT_SIZE+objectDet
ectorClass.score_value);
//
//      }
//  });
//      String filePath =
Uri.parse("file:///android_asset/model_2class.tflite").toString();

        Button btn_detail_objek = (Button) findViewById(R.id.button_detail);
        btn_detail_objek.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                BottomSheetDialog bottomSheetDialog = new
                BottomSheetDialog(CameraActivity.this, R.style.BottomSheetDialogTheme);

                View bottomSheetView =
                LayoutInflater.from(getApplicationContext())
                    .inflate(
                        R.layout.bottom_sheet,
                        (LinearLayout)findViewById(R.id.bottomsheetContainer)
                    );

                TextView tv31 = (TextView)
                bottomSheetView.findViewById(R.id.class_label);

```



```

        tv31.setText("Label List "+objectDetectorClass.labelList);

        TextView tv12 = (TextView)
bottomSheetView.findViewById(R.id.acc);

        TextView tv13 = (TextView)
bottomSheetView.findViewById(R.id.text_inference);

        TextView tv14 = (TextView)
bottomSheetView.findViewById(R.id.text_input_size);

//        TextView tv15 = (TextView)
bottomSheetView.findViewById(R.id.text_name_model);

        TextView tv16 = (TextView)
bottomSheetView.findViewById(R.id.text_informasi_device);

bottomSheetView.findViewById(R.id.buttonsheet_Back).setOnClickListener(n
ew View.OnClickListener() {

        @Override

        public void onClick(View view) {

            Toast.makeText(CameraActivity.this, "Refresh.",
Toast.LENGTH_SHORT).show();

            tv12.setText(String.format("Akurasi :
%.3f",objectDetectorClass.score_value)+"%");

            tv13.setText("Inference    : " +
objectDetectorClass.inferencetime+" ms");

            tv14.setText("Input Size    : " +
objectDetectorClass.INPUT_SIZE+" Height : " + objectDetectorClass.height+ "
Width : "+ objectDetectorClass.width);

            tv16.setText("Perangkat 128 GB (Total), RAM "+"6.00 GB
Versi Android 11, Processor Qualcomm SDM710 Delapan\u0002 inti");

        }

    });

bottomSheetView.findViewById(R.id.btn_sheet_kembali).setOnClickListener(
new View.OnClickListener() {

        @Override

```

```

        public void onClick(View view) {
            bottomSheetDialog.dismiss();

        }
    });

    bottomSheetDialog.setContentView(bottomSheetView);
    bottomSheetDialog.show();

}

});

mOpenCvCameraView = (CameraBridgeViewBase)
findViewById(R.id.frame_Surface);

mOpenCvCameraView.setVisibility(SurfaceView.VISIBLE);
mOpenCvCameraView.setCvCameraViewListener(this);

try {
    //input size is 300 for this model
    objectDetectorClass = new objectDetectorClass
        (getAssets(), "model_3class_10000.tflite", "label.txt", 320);
    Log.d("MainActivity", "Model Success loaded");
} catch (IOException e) {
    Log.d("MainActivity", "Getting Some Error");
    e.printStackTrace();
}

}

@Override
protected void onResume() {
    super.onResume();
    if (OpenCVLoader.initDebug()){
        Log.d(TAG, "Opencv initialization done");
    }
}

```

```

mLoaderCallback.onManagerConnected(LoaderCallbackInterface.SUCCESS);

    }else {
        Log.d(TAG,"Opencv is not loaded. try again");

OpenCVLoader.initAsync(OpenCVLoader.OPENCV_VERSION_3_4_0,this,m
LoaderCallback);

    }
}

@Override
protected void onPause() {
    super.onPause();
    if (mOpenCvCameraView != null){
        mOpenCvCameraView.disableView();
    }
}

public void onDestroy(){
    super.onDestroy();
    if (mOpenCvCameraView != null){
        mOpenCvCameraView.disableView();
    }
}

public void onCameraViewStarted(int width ,int height){
    mRgba=new Mat(height,width,CvType.CV_8UC4);
    mGray=new Mat(height,width,CvType.CV_8UC1);
}

```

```

    }

    public void onCameraViewStopped(){
        mRgba.release();
    }

    public Mat onCameraFrame(CameraBridgeViewBase.CvCameraViewFrame
inputFrame){
        mRgba=inputFrame.rgba();
        mGray=inputFrame.gray();
        capture=funcitonCapture(capture,mRgba);

        //now call the function
        Mat out=new Mat();
        out=objectDetectorClass.recognizeImage(mRgba);
        return out;
    }
}

```

```

package com.example.object_detection;

import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.content.res.AssetFileDescriptor;
import android.content.res.AssetManager;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.content.Context;
import android.graphics.Bitmap;
import android.os.Bundle;
import android.os.SystemClock;

```

```
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import java.io.IOException;
import java.util.List;

import org.tensorflow.lite.support.image.ImageProcessor;
import org.tensorflow.lite.support.image.TensorImage;
import org.tensorflow.lite.support.image.ops.Rot90Op;

import org.tensorflow.lite.task.vision.classifier.Classifications;
import org.tensorflow.lite.task.vision.classifier.ImageClassifier;
//import com.google.android.gms.tflite.gpu.GpuDelegate;
import org.checkerframework.checker.units.qual.A;
import org.opencv.android.Utils;
import org.opencv.core.Core;
import org.opencv.core.Mat;
import org.opencv.core.Point;
import org.opencv.core.Scalar;
import org.opencv.imgproc.Imgproc;
import org.tensorflow.lite.Interpreter;
import org.tensorflow.lite.gpu.GpuDelegate;
import org.tensorflow.lite.task.vision.classifier.Classifications;
import org.tensorflow.lite.task.vision.classifier.ImageClassifier;
import java.io.BufferedReader;
import java.io.File;
```

```
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.lang.reflect.Array;
import java.nio.ByteBuffer;
import java.nio.ByteOrder;
import java.nio.channels.FileChannel;
import java.util.ArrayList;
import java.util.List;
import java.util.Map;
import java.util.TreeMap;

public class objectDetectorClass {

    String getScore;

    private Interpreter interpreter;
    List<String> labelList;
    int INPUT_SIZE;
    private int PIXEL_SIZE=3;//rgb
    private int IMAGE_MEAN=0;
    private float IMAGE_STD=255.0f;
    //use to inisialisasi gpu in app
    GpuDelegate gpuDelegate;
    int height;
    int width;
    float score_value;
    float class_value;
    String xscore_value;
```

```

    objectDetectorClass(AssetManager assetManager, String modelPath, String
    labelPath, int inputSize) throws IOException {

        INPUT_SIZE = inputSize;

        //USE TO DEFINE CPU or GPU

        Interpreter.Options options = new Interpreter.Options();
        gpuDelegate = new GpuDelegate();
        options.addDelegate(gpuDelegate);
        options.setNumThreads(4);//set it according to your phone

        //loading model

//    this.threshold = threshold;
//    this.numThreads = numThreads;
//    this.maxResults = maxResults;

        interpreter = new Interpreter(loadModelFile(assetManager, modelPath),
        options);

        //load labelmap

        labelList=loadLabelList(assetManager,labelPath);

    }

    private List<String> loadLabelList(AssetManager assetManager, String
    labelPath ) throws IOException {

        //to store label

        List<String> labelList=new ArrayList<>();

        BufferedReader reader=new BufferedReader(new
        InputStreamReader(assetManager.open(labelPath)));

        String line;

        //loop through each line and store it ro labellist

        while ((line=reader.readLine())!=null){

            labelList.add(line);

        }
    }

```

```

        reader.close();

        return labelList;
    }

    public Mat recognizeImage(Mat mat_image){
        Mat rotated_mat_image=new Mat();
        Mat a=mat_image.t();
        Core.flip(a,rotated_mat_image,1);
        a.release();

        Bitmap bitmap=null;

        bitmap=Bitmap.createBitmap(rotated_mat_image.cols(),rotated_mat_image.ro
ws(),Bitmap.Config.ARGB_8888);

        Utils.matToBitmap(rotated_mat_image,bitmap);

        height=bitmap.getHeight();
        width=bitmap.getWidth();

        Bitmap
scaledBitmap=Bitmap.createScaledBitmap(bitmap,INPUT_SIZE,INPUT_SIZE
,false);

        ByteBuffer byteBuffer=convertBitmapToByteBuffer(scaledBitmap);

//        float[][][]result=new float[1][10][4];
        Object[] input=new Object[1];
        input[0]=byteBuffer;

        Map<Integer,Object> output_map=new TreeMap<>();
    
```



```

float[][][]boxes = new float[1][10][4];

float[][] scores=new float[1][10];
float[][] classes=new float[1][10];

output_map.put(1,boxes);
output_map.put(3,classes);
output_map.put(0,scores);

interpreter.runForMultipleInputsOutputs(input,output_map);

//untuk deteksi hanya 1 kelas saja
//  output_map.put(0,boxes);
//  output_map.put(1,classes);
//  output_map.put(2,scores);
//  Object value = output_map.get(0);
//  Object Object_class=output_map.get(1);
//  Object score=output_map.get(2);

//untuk deteksi lebih dari 1 kelas
Object value = output_map.get(1);
Object Object_class=output_map.get(3);
Object score=output_map.get(0);

for (int i=0;i<10;i++){
    class_value=(float) Array.get(Array.get(Object_class,0),i);
    score_value= (float) Array.get(Array.get(score,0),i);
    xscore_value=String.format("%.2f",score_value);
    //define threshold for score

```

```

        if (score_value>0.5){
            Object box1=Array.get(Array.get(value,0),i);
            //we are multiplying it with original height and width of frame

            float top = (float) Array.get(box1,0)*height;
            float left= (float) Array.get(box1,1)*width;
            float bottom =(float) Array.get(box1,2)*height;
            float right =(float) Array.get(box1,3)*width;

            //draw rectangle in Original frame //starting point /ending point of
            box /color of box  thickness

            Imgproc.rectangle(rotated_mat_image,new Point(left,top),new
            Point(right,bottom),new Scalar(100,255,100),2);

            Imgproc.putText(rotated_mat_image,labelList.get((int)class_value) ,
            new Point(left,top),3,1,new Scalar(250,60,0),2);

            Imgproc.putText(rotated_mat_image,"  "+xscore_value+" %", new
            Point(right,top),3,0.7,new Scalar(250,200,0),2,Imgproc.LINE_4);

//
            Imgproc.putText(rotated_mat_image,labelList.get(labelList.indexOf("kudis")),
            new Point(left,top),3,1,new Scalar(100,100,200),2);
//
            Imgproc.putText(rotated_mat_image,labelList.get((labelList.indexOf("ringwor
            m"))), new Point(left,top),3,1,new Scalar(200,100,100),2);

            System.out.println("scores_value"+score_value);
            System.out.println("scores"+scores);
            System.out.println("score"+score);

```

```

        System.out.println("input_size"+INPUT_SIZE);
        System.out.println("labellist"+labelList);
        System.out.println("class value: "+class_value);
        System.out.println("height"+height);
        System.out.println("width"+width);

    }
}

//before returning rotate
Mat b=rotated_mat_image.t();
Core.flip(b,mat_image,0);
b.release();

return mat_image;
}

private ByteBuffer loadModelFile(AssetManager assetManager, String
modelPath) throws IOException {
    AssetFileDescriptor fileDescriptor=assetManager.openFd(modelPath);

    FileInputStream inputStream = new
FileInputStream(fileDescriptor.getFileDescriptor());

    FileChannel fileChannel=inputStream.getChannel();

    long startOffset = fileDescriptor.getStartOffset();
    long declaredLength=fileDescriptor.getDeclaredLength();

    return
fileChannel.map(FileChannel.MapMode.READ_ONLY,startOffset,declaredLe
ngth);
}

long inferencetime;

```

```


ByteBuffer convertBitmapToByteBuffer(Bitmap bitmap) {
    ByteBuffer byteBuffer;
    int quant = 1;
    int size_images = INPUT_SIZE;
    if (quant == 0) {
        byteBuffer = ByteBuffer.allocateDirect(1*size_images*size_images*3);

    } else {
        byteBuffer = ByteBuffer.allocateDirect(4*1* size_images * size_images
* 3);
    }
    byteBuffer.order(ByteOrder.nativeOrder());
    int[] intValues = new int[size_images * size_images];
    bitmap.getPixels(intValues, 0, bitmap.getWidth(), 0, 0, bitmap.getWidth(),
bitmap.getHeight());
    final long startTime = SystemClock.uptimeMillis();
    int pixel = 0;
    for (int i = 0; i < size_images; ++i) {
        for (int j = 0; j < size_images; ++j) {
            final int val = intValues[pixel++];
            if (quant == 0) {
                byteBuffer.put((byte)((val >> 16) & 0xFF));
                byteBuffer.put((byte)((val >> 8) & 0xFF));
                byteBuffer.put((byte)(val & 0xFF));
            } else {
                byteBuffer.putFloat((((val >> 16) & 0xFF)) / 255.0f);
                byteBuffer.putFloat((((val >> 8) & 0xFF)) / 255.0f);
                byteBuffer.putFloat((((val) & 0xFF)) / 255.0f);
            }
        }
    }
}

```

```
inferencetime = (long) SystemClock uptimeMillis() - startTime;
System.out.println("inference : " + inferencetime + "ms");
String xx = String.valueOf(inferencetime + "second");
}
return byteBuffer;
}
String xx = String.valueOf(inferencetime);
}
```

## Lampiran 11 Lembar Bimbingan Tugas Akhir



**UNIVERSITAS BUANA PERJUANGAN KARAWANG**  
**FAKULTAS ILMU KOMPUTER**  
**Terakreditasi BAN-PT**  
 Jl. H.S. Ronggowaluyo Telukjambe Timur Karawang 41361 Telp./Fax. (0267) 8403140  
 Site: <http://fik.ubpkarawang.ac.id> email: [fik@ubpkarawang.ac.id](mailto:fik@ubpkarawang.ac.id)

---

F04/TA/2022

**LEMBAR BIMBINGAN LAPORAN TUGAS AKHIR**









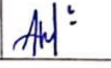
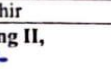
Nama Mahasiswa : Rizandi Aditya Fitrah

NIM : 19411255201185 Program Studi : Teknik Informatika

Pembimbing I : Amis Fitri Nur Masruyah, Mkom



Pembimbing II : Ayu Ratna Juwita, M.kom

Judul TA : Deteksi Penyakit Kulit Kulit Menggunakan Algoritma CNN Dengan TensorFlow Lite Berbasis Android

No.	Tanggal Bimbingan	Target Bimbingan	Hasil Bimbingan dan Rencana Selanjutnya	Paraf Pembimbing
1	03/03/2023	Menyelesaikan Sistem	Menentukan kelanjutan Sistem	
2	07/03/2023	Menyelesaikan bab IV	Revisi Bab IV	
3	23/05/2023	Menyelesaikan bab IV	Revisi Bab IV	
4	24/05/2023	Menyelesaikan bab IV	Revisi Bab IV	
5	25/05/2023	Menyelesaikan bab IV	Revisi Bab IV	
6	30/05/2023	Menyelesaikan bab IV	Revisi Bab IV	
7	31/05/2023	Tanda tangan laporan	Mendapat tanda tangan laporan	
8	29/05/2023	Menyelesaikan bab IV	Revisi Bab IV	
9	25/05/2023	Menyelesaikan bab IV	Revisi Bab IV	
10	26/05/2023	Menyelesaikan bab IV	Revisi Bab IV	

Catatan :

- Formulir ini lanjutan formulir bimbingan proposal TA
- Formulir ini diisi setiap bimbingan dan dilampirkan saat pendaftaran sidang Tugas Akhir dengan minimal 8 kali bimbingan.

Rekomendasi Mengikuti Sidang Tugas Akhir	
Pembimbing I,	Pembimbing II,
 Nama : Tanggal : 31 Mei 2023	 Nama : Tanggal : 05 Juni 2023

### **RIWAYAT PENULIS**



Nama lengkap Riyandi Aditya Fitrah. Kelahiran di Karawang, tanggal 12 bulan agustus tahun 2001. Putra kedua dari pasangan seorang Bapak Sarip Hidayatullah dan Ibu Dina Andini. Menyelesaikan pendidikan saat SDN Purwasari III, SMP Negeri 1 Klari, SMA Negeri 2 Karawang, serta melanjutkan pendidikan perguruan tinggi Universitas Buana Perjuangan Karawang.