

PERANCANGAN SISTEM DETEKSI SAPI TERSERANG CACING PITA BERBASIS CITRA DAN *RASPBERRY PI*

SKRIPSI

untuk memenuhi salah satu persyaratan
mencapai derajat Sarjana S1



Disusun oleh:

Fikri Prayoga

15524033

**Jurusan Teknik Elektro
Fakultas Teknologi Industri
Universitas Islam Indonesia
Yogyakarta
2020**

LEMBAR PENGESAHAN

PERANCANGAN SISTEM DETEKSI SAPI TERSERANG CACING PITA BERBASIS CITRA DAN
RASPBERRY PI

TUGAS AKHIR

Diajukan sebagai Salah Satu Syarat untuk Memperoleh
Gelar Sarjana Teknik
pada Program Studi Teknik Elektro
Fakultas Teknologi Industri
Universitas Islam Indonesia

Disusun oleh:

Fikri Prayoga
15524033

Yogyakarta, 24 Agustus 2020

Menyetujui,

Pembimbing 1



Dwi Ana Ratna Wati
035240102

LEMBAR PENGESAHAN

SKRIPSI

PERANCANGAN SISTEM DETEKSI SAPI TERSERANG CACING PITA BERBASIS CITRA DAN *RASPBERRY PI*

Dipersiapkan dan disusun oleh:

Fikri Prayoga

15524033

Telah dipertahankan di depan dewan penguji

Pada tanggal: 4 September 2020

Susunan dewan penguji

Ketua Penguji : Dwi Ana Ratna Wati, S. T, M. Eng,

Anggota Penguji 1: Alvin Sahroni, S. T, M. Eng, Ph. D,

Anggota Penguji 2: Dzata Farahiyah, S. T, M. Sc,

Skripsi ini telah diterima sebagai salah satu persyaratan
untuk memperoleh gelar Sarjana

Tanggal: 4 September 2020

Ketua Program Studi Teknik Elektro



Yusuf Aziz Amrullah, S. T, M. Sc, Ph. D

045240101

PERNYATAAN

Dengan ini Saya menyatakan bahwa:

1. Skripsi ini tidak mengandung karya yang diajukan untuk memperoleh gelar kesarjanaan di suatu Perguruan Tinggi, dan sepanjang pengetahuan Saya juga tidak mengandung karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis diacu dalam naskah ini dan disebutkan dalam daftar pustaka.
2. Informasi dan materi Skripsi yang terkait hak milik, hak intelektual, dan paten merupakan milik bersama antara tiga pihak yaitu penulis, dosen pembimbing, dan Universitas Islam Indonesia. Dalam hal penggunaan informasi dan materi Skripsi terkait paten maka akan diskusikan lebih lanjut untuk mendapatkan persetujuan dari ketiga pihak tersebut diatas.



Yogyakarta, 4 September 2020



Fikri Prayoga

KATA PENGANTAR

Alhamdulillah rabbil ‘aalamin, atas limpahan rahmat Allah Subhanawata’ala yang telah memberikan kemudahan kepada penulis untuk menyelesaikan skripsi dengan judul Perancangan Sistem Deteksi Sapi Terserang Cacing Pita Berbasis Citra dan *Raspberry Pi*. Ini untuk memenuhi salah satu syarat dalam menyelesaikan studi serta dalam rangka memperoleh gelar Sarjana Pendidikan Strata Satu pada Program Studi Teknik Elektro Fakultas Teknologi Industri Universitas Islam Indonesia.

Penghargaan dan terima kasih yang setulus-tulusnya kepada Ayahanda tercinta Syaiful Ampu dan Ibunda yang saya sayangi karena telah mencurahkan segenap cinta dan kasih sayang serta perhatian moril maupun materil. Semoga Allah Subhanahu wata’ala selalu melimpahkan rahmat, kesehatan, karunia dan keberkahan di dunia dan di akhirat atas budi baik yang telah diberikan kepada penulis. Penghargaan dan terima kasih penulis berikan kepada Ibu Dwi Ana Ratna Wati, S.T, M.Eng selaku Pembimbing yang telah membantu penulisan skripsi ini. Serta ucapan terima kasih kepada :

1. Bapak Prof. Fathul Wahid, ST., M.Sc., Ph.D selaku Rektor Universitas Islam Indonesia.
2. Bapak Prof. Dr. Ir. Hari Purnomo, M. T selaku Dekan Fakultas Teknologi Industri Universitas Islam Indonesia.
3. Bapak Yusuf Aziz Amrulloh, S. T, M. Eng, Ph.D selaku Ketua Jurusan Program Studi Teknik Elektro.
4. Kedua orang tua saya yang telah memberikan dukungan dan do’a sehingga penulis dapat menyelesaikan skripsi ini.
5. Kerabat dan teman-teman saya yang tidak bisa saya sebutkan satu-persatu yang telah membantu saya dalam proses penyelesaian skripsi ini.

Akhir kata saya menyadari bahwa dalam penulisan skripsi ini masih jauh dari kesempurnaan. Karena itu, penulis memohon saran dan kritik yang sifatnya membangun demi kesempurnaannya dan semoga bermanfaat bagi kita semua.

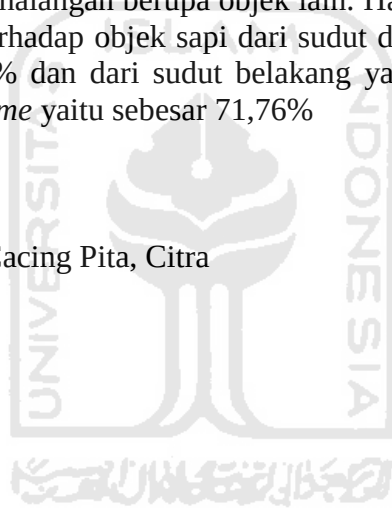
Yogyakarta, 4 September 2020

Fikri Prayoga

ABSTRAK

Latar belakang dari penelitian ini yaitu ketersediaan daging sapi yang berkualitas dan aman dari bahan berbahaya merupakan salah satu kunci penting untuk mencapai tingkat gizi yang baik bagi masyarakat. Penyakit yang disebabkan parasit terutama cacing pada hewan di peternakan merupakan salah satu permasalahan yang sering dihadapi peternak sehingga diperlukan teknologi untuk membantu menyelesaikan masalah pada kesehatan sapi. Tujuan dari penelitian ini yaitu dapat meningkatkan efisiensi dalam usaha untuk mendeteksi sapi yang mempunyai potensi terkena parasit cacing pita menggunakan teknologi dengan cara merekam gambar atau video, lalu mendapatkan pemberitahuan sapi yang mempunyai potensi terkena serangan parasit cacing pita. Hasil dari penelitian ini yaitu sistem mampu untuk mendeteksi sapi yang mempunyai potensi terkena serangan parasit cacing pita, tetapi terdapat kendala pada beberapa kondisi yang menyebabkan sistem tidak dapat menganalisis kondisi sapi. Berdasarkan analisis, kesalahan ini terjadi dikarenakan kurangnya variasi dan jumlah gambar untuk dilakukan *training* oleh sistem. Kesimpulan dari penelitian ini yaitu sistem yang dirancang dapat membedakan kondisi sapi yang memiliki potensi terkena serangan parasit cacing pita tetapi terdapat beberapa kesalahan analisa pada keadaan tertentu seperti pada saat kamera tidak dapat mendeteksi kepala sapi karena ada halangan berupa objek lain. Hasil dari pengujian yaitu tingkat keakurasian alat yang dirancang terhadap objek sapi dari sudut depan yaitu sebesar 62,5%, dari sudut samping yaitu sebesar 26,6% dan dari sudut belakang yaitu sebesar 28,5%. Sedangkan tingkat keakuratan pada saat *real time* yaitu sebesar 71,76%

Keyword : Parasit, Penyakit sapi, Cacing Pita, Citra



DAFTAR ISI

LEMBAR PENGESAHAN.....	i
LEMBAR PENGESAHAN.....	ii
KATA PENGANTAR.....	iv
ABSTRAK	v
DAFTAR ISI.....	vi
DAFTAR GAMBAR	viii
DAFTAR TABEL	x
ARTI LAMBANG DAN SINGKATAN	xi
BAB 1 PENDAHULUAN	1
1.1 Latar Belakang Masalah	1
1.2 Rumusan Masalah.....	2
1.3 Batasan Masalah	2
1.4 Tujuan Penelitian	2
1.5 Manfaat Penelitian	2
BAB 2 TINJAUAN PUSTAKA	3
2.1 Studi Literatur	3
2.2 Tinjauan Teori.....	5
2.2.1 <i>Raspberry Pi</i>	5
2.2.2 <i>Raspberry Pi Camera Module</i>	5
2.2.3 <i>Internet of Things(IoT)</i>	6
2.2.4 Cacing Pita.....	7
2.2.5 Ciri-ciri fisik sapi yang terkena serangan parasit cacing pita	8
2.2.6 <i>Convolutional Neural Network</i>	10
2.2.7 <i>Tensorflow</i>	10
BAB 3 METODOLOGI.....	11

3.1 Perancangan Sistem	11
3.2 <i>Training System</i>	12
3.3 Implementasi Sistem.....	18
3.4 Metode Pengujian	19
BAB 4 HASIL DAN PEMBAHASAN.....	21
4.1 Hasil perancangan dan cara kerja sistem	21
4.2 Pengujian gambar sapi pada berbagai sudut dan akurasiya	21
4.3 Pengujian pada sistem <i>real time</i> dan akurasiya	23
4.4 Pembahasan	26
BAB 5 KESIMPULAN DAN SARAN.....	28
5.1 Kesimpulan	28
5.2 Saran	28
DAFTAR PUSTAKA	29
LAMPIRAN.....	32
1. Hasil Deteksi Sapi	32
2. <i>Training Script</i>	44
3. <i>XML to CSV Script</i>	47
4. <i>Generate TFRecord Script</i>	49
5. <i>Testing Script</i>	52
6. <i>Resizer Script</i>	57

DAFTAR GAMBAR

Gambar 2.1 Diagram prevalensi cacing saluran pencernaan pada sapi di Provinsi Lampung[2]...	4
Gambar 2.2 Diagram Prevalensi cacing saluran pencernaan pada sapi di Kecamatan Tikung, Lamongan, Jawa Timur[9]	4
Gambar 2.3 Raspberry Pi 3 B+[11].....	5
Gambar 2.4 Modul kamera <i>Raspberry Pi</i> [13].....	6
Gambar 2.5 Gambaran konsep dari <i>Internet of Things(IoT)</i> [14]	6
Gambar 2.6 Cacing Pita[16].....	7
Gambar 2.7 Sapi kelihatan tulang rusuk akibat cacing pita[19]	8
Gambar 2.8 Sapi yang terkena serangan cacing pita[20]	9
Gambar 2.9 Sapi sehat terlihat gemuk[22].....	9
Gambar 2.10 Sapi gemuk yang sehat[23]	9
Gambar 2.11 Ilustrasi <i>Convolutional Neural Network</i> [25].....	10
Gambar 2.12 Posisi <i>tensorflow</i> pada <i>hardware</i> [27].....	10
Gambar 3.1 Tahapan perancangan sistem.....	11
Gambar 3.2 Alur <i>training model system</i>	12
Gambar 3.3 <i>CSV File</i>	13
Gambar 3.4 <i>Label Map Data</i>	13
Gambar 3.5 Ilustrasi proses konvolusi[29]	14
Gambar 3.6 Ilustrasi proses <i>pooling</i> [31]	15
Gambar 3.7 Ilustrasi proses <i>ReLU</i> [33].....	15
Gambar 3.8 Ilustrasi perubahan nilai negatif menjadi 0[34].....	16
Gambar 3.9 Ilustrasi model yang dihasilkan[35]	16
Gambar 3.10 Ilustrasi pengujian dan penentuan nilai keakuratan[36].....	17
Gambar 3.11 Alur logika proses klasifikasi sapi yang memiliki ciri-ciri terkena serangan parasit cacing pita	18
Gambar 3.12 Sapi yang berada di Kecamatan Prambanan, Kabupaten Sleman	19
Gambar 3.13 Sapi yang berada di Kecamatan Kalibawang, Kabupaten Kulon Progo	19
Gambar 3.14 Alur persiapan pengujian.....	20
Gambar 4.1 Diagram sistem yang dirancang	21
Gambar 4.2 Sapi terinfeksi cacing pita yang diuji dengan data training[19]	22
Gambar 4.3 Sapi sehat yang diuji dengan data <i>training</i> [37].....	22
Gambar 4.4 Sapi terdeteksi dengan sempurna	24

Gambar 4.5 Sapi yang tidak memiliki ciri-ciri fisik terkena serangan parasit cacing pita	24
Gambar 5.1 Monitoring proses training	32
Gambar 5.2 Proses training pada CMD	32
Gambar 5.3 Penyedia server lokal.....	33
Gambar 5.4 Sapi yang terkena serangan parasit cacing namun salah deteksi.....	33
Gambar 5.5 Sapi yang terkena serangan parasit cacing namun salah deteksi.....	34
Gambar 5.6 Cacing yang terkena serangan parasit cacing dan terdeteksi benar.....	34
Gambar 5.7 Sapi yang terkena serangan parasit cacing namun salah deteksi.....	35
Gambar 5.8 Sapi yang terkena serangan parasit cacing namun salah deteksi.....	35
Gambar 5.9 Sapi yang terkena serangan parasit cacing namun salah deteksi.....	36
Gambar 5.10 Sapi yang tidak terkena serangan parasit cacing dan terdeteksi benar	36
Gambar 5.11 Sapi yang terkena serangan parasit cacing namun salah deteksi.....	37
Gambar 5.12 Sapi yang tidak terkena serangan parasit cacing dan terdeteksi benar	37
Gambar 5.13 Sapi yang tidak terkena serangan parasit cacing namun salah deteksi.....	38
Gambar 5.14 Sapi yang terkena serangan parasit cacing namun salah deteksi.....	39
Gambar 5.15 Sapi yang terkena serangan parasit cacing namun salah deteksi.....	40
Gambar 5.16 Sapi yang terkena serangan parasit cacing dan terdeteksi benar.....	40
Gambar 5.17 Sapi yang terkena serangan parasit cacing namun salah deteksi.....	41
Gambar 5.18 Sapi yang terkena serangan parasit cacing namun tidak terdeteksi.....	41
Gambar 5.19 Sapi yang terkena serangan parasit cacing namun tidak terdeteksi.....	42
Gambar 5.20 Sapi yang tidak terkena serangan parasit cacing dan terdeteksi.....	42
Gambar 5.21 Sapi yang tidak terkena serangan parasit cacing dan terdeteksi benar	43
Gambar 5.22 Sapi yang terkena serangan parasit cacing namun salah deteksi.....	43

DAFTAR TABEL

Tabel 4.1 Tabel keakuratan analisis berdasarkan sudut	22
Tabel 4.2 Tabel hasil pengujian dari klasifikasi dengan data <i>training</i>	23
Tabel 4.3 Tabel kemampuan deteksi.....	25
Tabel 4.4 Tabel hasil pengujian dari klasifikasi secara <i>real time</i>	25



ARTI LAMBANG DAN SINGKATAN

Cestoda	: Cacing Pita
IoT	: <i>Internet of Things</i>
CNN	: <i>Convolutional Neural Network</i>
CSV	: <i>Comma Separated Values</i>
ReLU	: <i>Rectified Linear Unit</i>
Wifi	: <i>Wireless Fidelity</i>



BAB 1

PENDAHULUAN

1.1 Latar Belakang Masalah

Ketersediaan daging sapi yang berkualitas dan aman dari bahan berbahaya merupakan salah satu kunci penting untuk mencapai tingkat gizi yang baik bagi masyarakat[1]. Penyakit yang disebabkan parasit terutama cacing pada hewan di peternakan merupakan salah satu permasalahan yang sering dihadapi peternak[2]. Penyakit pada ternak dapat menimbulkan kerugian ekonomi yang cukup besar bagi petani dikarenakan selain merusakkan kehidupan ternak juga dapat menular kepada manusia[3]. *Taeniasis* merupakan penyakit yang disebabkan oleh cacing pita (*cestoda*) dari genus *Taenia*. *Sistiserkosis* merupakan penyakit yang disebabkan oleh infeksi larva dari *Taenia sp.* *Sistiserkosis* pada manusia umumnya disebabkan infeksi oleh larva *Taenia solium* sedangkan *Taenia saginata* dapat menyebabkan infeksi saluran pencernaan oleh cacing dewasa[4]. Akibat dari parasit cacing pita menyebabkan sapi menjadi sakit dengan ciri-ciri fisik badan yang kurus dan perut besar, serta ciri-ciri lainnya yaitu seperti tidak nafsu makan, pertumbuhan lambat, kotoran encer dan demam[5].

Pada penelitian sebelumnya yang terkait dengan *smart farming* beberapa diantaranya yaitu seperti penggunaan teknologi untuk mengetahui posisi sapi agar sapi tetap bisa dilacak posisinya untuk peternakan sapi yang dilepas. Contoh lainnya yaitu penyiraman pada tanaman secara otomatis maupun dikontrol secara manual. Perbedaannya pada penelitian ini yaitu penggunaan teknologi untuk menurunkan resiko penyakit cacing pita pada sapi. Ciri-ciri fisik sapi yang terkena serangan parasit cacing pita dapat ditangkap melalui kamera sehingga berdasarkan latar belakang tersebut, maka dirancang sebuah sistem yang dapat meningkatkan efisiensi dalam penanganan serangan parasit cacing pita pada sapi dengan menggunakan *Raspberry Pi* agar proses klasifikasi sapi yang memiliki ciri-ciri fisik terkena serangan parasit cacing pita menjadi lebih cepat.

1.2 Rumusan Masalah

Rumusan masalah pada penelitian ini yaitu :

- a. Bagaimana perancangan sistem untuk mendeteksi sapi yang berpotensi terkena penyakit cacing pita menggunakan *Raspberry Pi*?
- b. Berapa tingkat akurasi sistem dalam mendeteksi potensi sapi berpenyakit cacing pita?

1.3 Batasan Masalah

Batasan masalah pada penelitian ini yaitu sebagai berikut :

- a. Klasifikasi pada penelitian ini dilakukan pada sapi yang berukuran besar dengan bobot siap potong ideal yaitu 250-350 kg[6]
- b. Pendeteksian dilakukan pada sapi dalam kondisi berdiri
- c. Pendeteksian dilakukan pada saat siang hari dengan kondisi langit cerah yaitu dengan rata-rata intensitas cahaya sebesar 29950 lux[7], pagi maupun sore hari

1.4 Tujuan Penelitian

Tujuan dari penelitian ini yaitu:

- a. Agar sistem mampu mendeteksi sapi yang berpotensi terkena penyakit cacing pita menggunakan *Raspberry Pi* yaitu dengan cara melakukan perekaman terhadap sapi.
- b. Agar diketahui tingkat akurasi sistem dalam mendeteksi potensi sapi berpenyakit cacing pita.

1.5 Manfaat Penelitian

Adapun manfaat penelitian ini yaitu sebagai berikut :

- a. Menjadi salah satu fitur dalam pengembangan pemanfaatan teknologi pada peternakan
- b. Dapat mempercepat proses klasifikasi sapi yang mempunyai ciri-ciri fisik terkena serangan parasit cacing pita
- c. Dapat meningkatkan efisiensi dalam penanggulangan serangan parasit cacing pita pada sapi

BAB 2

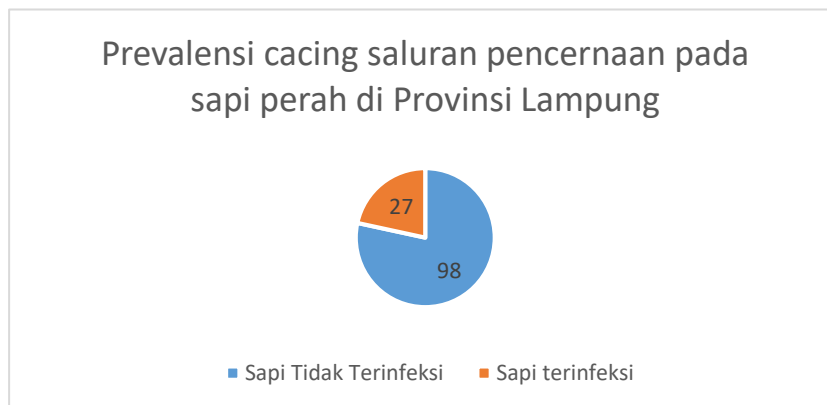
TINJAUAN PUSTAKA

2.1 Studi Literatur

Berdasarkan hasil riset, penelitian tentang efisiensi pada peternakan sapi ini sudah banyak dilakukan, namun masih banyak yang menggunakan teknologi peternakan pintar hanya untuk memantau kondisi di sebuah peternakan dengan sedikit kontrol tetapi masih kurang efisien[8]. Pada salah satu jurnal yang berjudul “*Smart Cow Health Monitoring, Farm Environmental Monitoring And Control System Using Wireless Sensor Networks*” [5] membuat sebuah sistem yang berfungsi untuk memonitoring kondisi ternak secara *real time* agar bisa memantau kondisi sapi secara langsung dengan komponen seperti kamera, sensor kelembapan untuk mengukur kelembapan lingkungan pada area peternakan, lalu sensor suhu untuk memantau suhu pada area peternakan. Sedangkan untuk pengontrolan ada semprot air yang yang bisa dikendalikan oleh peternak secara langsung untuk meningkatkan kelembapan pada area peternakan dengan cara menyemprotkan butiran air pada area peternakan.

Riset literatur yang didapatkan, yaitu berjudul “*Smart Dairy Farming through the Internet of Things*” [6] membuat sebuah sistem yang cukup rumit dibandingkan dengan literatur yang hanya memantau beberapa komponen pada lingkungan seperti suhu dan kelembapan pada area peternakan [5]. Pada literatur ini [6], menggunakan beberapa komponen yang lebih banyak untuk memantau kondisi pada area peternakan seperti “*digital tracking for each cow, Genomic Testing: Herd Inteligent, A Digitally signaled birth, Sensor driven crop management*”, sehingga bisa membuat nilai efisiensi pada peternakan menjadi meningkat dikarenakan banyaknya komponen pemantau pada area peternakan. Perbedaan antara penelitian ini dengan yang sudah ada berdasarkan hasil pencarian beberapa referensi yang didapat, yaitu terletak pada fitur klasifikasi sapi yang memiliki ciri-ciri fisik terkena serangan parasit cacing pita. Pada referensi hanya mempunyai beberapa kemampuan *monitoring* seperti suhu, kelembapan dan pendeteksi lokasi pada binatang ternak, sedangkan pada penelitian ini, membuat sebuah sistem untuk mengklasifikasi sapi yang mempunyai ciri-ciri fisik terkena serangan parasit cacing pita.

Berikut ini adalah prevalensi cacing saluran pencernaan pada sapi di beberapa daerah Indonesia. Salah satu hasil penelitian dari Universitas Lampung sebagai berikut:



Gambar 2.1 Diagram prevalensi cacing saluran pencernaan pada sapi di Provinsi Lampung[2]

Berdasarkan gambar 2.1 dapat diketahui bahwa prevalensi cacing saluran pencernaan pada sapi di daerah Lampung yaitu sebesar 21,60% yang didapat dari keseluruhan contoh sapi yang berjumlah 125 ekor sapi, 98 ekor sapi yang negatif terinfeksi cacing pada saluran pencernaan dan 27 ekor sapi yang positif terinfeksi cacing pada saluran pencernaan[2]. Kedua yaitu salah satu hasil penelitian dari Universitas Airlangga sebagai berikut:



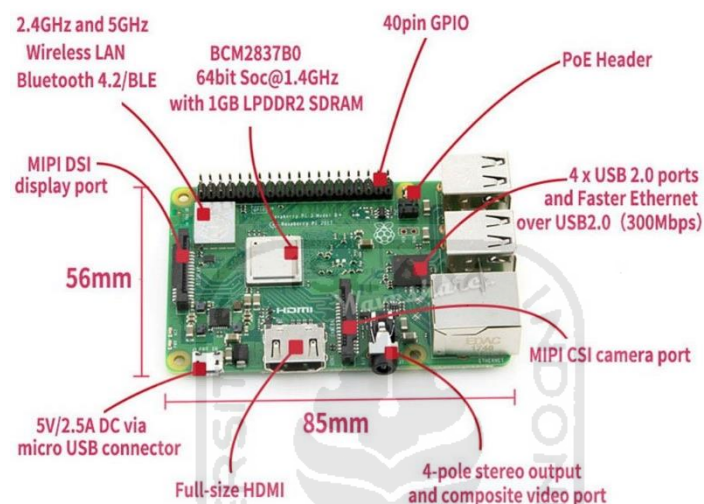
Gambar 2.2 Diagram Prevalensi cacing saluran pencernaan pada sapi di Kecamatan Tikung, Lamongan, Jawa Timur[9]

Berdasarkan gambar 2.2 dapat diketahui bahwa prevalensi cacing saluran pencernaan pada sapi di Kecamatan Tikung, Lamongan, Jawa Timur yaitu sebesar 54% yang didapat dari keseluruhan sapi yang berjumlah 50 ekor sapi, 27 ekor sapi yang positif terinfeksi cacing saluran pencernaan dan 23 ekor sapi yang negatif terinfeksi terkena cacing saluran pencernaan[9]. Kontribusi literatur sebelumnya pada penelitian ini yaitu menggunakan jaringan koneksi, *machine learning platform* dan *Raspberry Pi system* yang sama dengan literatur sebelumnya untuk mendeteksi sapi yang memiliki potensi terkena serangan parasit cacing pita.

2.2 Tinjauan Teori

2.2.1 Raspberry Pi

Raspberry Pi adalah sebuah komputer yang memiliki ukuran kecil. Komputer ini bisa dihubungkan dengan monitor atau televisi, *keyboard*, *mouse*, *pen-drive* dan sebagainya. *Raspberry Pi* mempunyai sistem operasi bawaan yang dinamakan dengan *Raspbian*. Pengembang perangkat lunak bisa menjalankan *script* dengan menggunakan Bahasa *Python*[10].



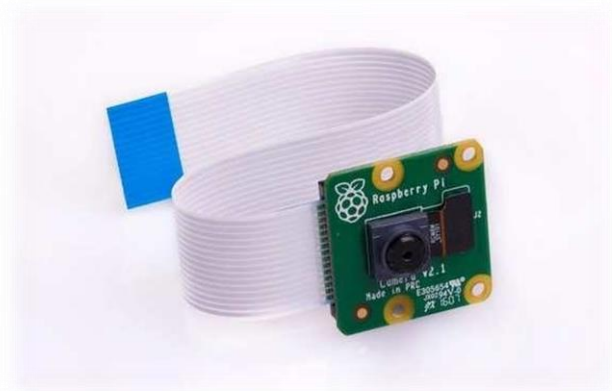
Gambar 2.3 Raspberry Pi 3 B+[11]

Gambar 2.3 menunjukkan gambar *Raspberry Pi 3 B+* yang mempunyai beberapa *pin* dan *port* seperti komputer biasa yang digabungkan dengan sebuah *Arduino*. Berdasarkan hasil riset penulis, komputer mini seperti *Raspberry Pi* ini biasanya memang digunakan untuk pengolah data pada robot yang mempunyai tingkat kompleksitas pemrosesan yang relatif tinggi, karena *processor* dan komponen-komponen yang dimiliki sama seperti komponen pada komputer pada umumnya, hanya saja memiliki kapasitas yang relatif lebih rendah.

2.2.2 Raspberry Pi Camera Module

Modul kamera *Raspberry Pi v2* memiliki sensor Sony IMX219 8 Megapiksel. Modul kamera ini dapat merekam ataupun mengambil foto dengan resolusi tinggi. Modul kamera ini juga mempunyai *library* agar pengembang program bisa dengan mudah membuat efek berdasarkan *library* yang telah disediakan oleh *Raspberry Pi*. Kamera ini bisa digunakan untuk semua model *Raspberry Pi* 1, 2, 3 dan 4. Kamera ini dapat diakses melalui API MMAL dan V4L dan ada banyak *library* pihak ketiga yang bisa digunakan untuk mengontrol kamera. Kamera ini mendukung video

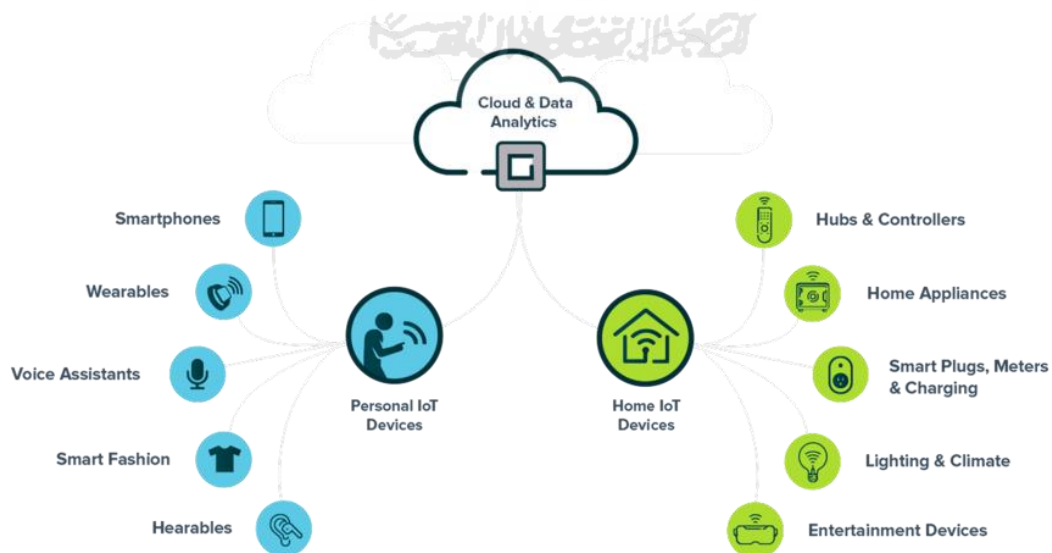
dengan resolusi 1080p, 720p60 dan VGA90. Kamera ini menggunakan *flexible cable* sepanjang 15 cm yang bisa dihubungkan pada jenis port CSI pada *Raspberry Pi*[12].



Gambar 2.4 Modul kamera *Raspberry Pi*[13]

2.2.3 Internet of Things(IoT)

Internet of thing (IoT) merupakan suatu konsep yang bertujuan untuk memperluas manfaat dari konektivitas internet yang tersambung secara terus menerus. IoT bisa dimanfaatkan pada gedung untuk mengendalikan peralatan elektronik seperti lampu ruangan yang dapat dioperasikan dari jarak jauh melalui jaringan komputer, tidak dapat dipungkiri kemajuan teknologi yang sedemikian cepat harus bisa dimanfaatkan, dipelajari serta diterapkan dalam kehidupan sehari-hari.



Gambar 2.5 Gambaran konsep dari *Internet of Things*(IoT)[14]

Contohnya adalah perkembangan teknologi yang bisa dimanfaatkan dari adanya koneksi internet ini bisa mengakses peralatan elektronik seperti lampu ruangan yang dapat dioperasikan dengan cara daring melalui telepon pintar. Sehingga, dapat memudahkan pengguna memantau ataupun mengendalikan lampu kapanpun dan dimanapun dengan catatan di lokasi yang akan diterapkan teknologi kendali jarak jauh mempunyai jaringan internet yang memadai. Sistem kendali jarak jauh, memudahkan pengguna dalam mengontrol lampu gedung yang jaraknya cukup jauh lokasinya[15].

2.2.4 Cacing Pita

Jenis cacing pita yang dapat menyerang sapi ialah spesies *Taenia sp*, *Moniezia sp*, dan *Echinococcus granulosus*. Dari ketiga cacing tersebut, hanya spesies *Moniezia sp* yang hidup sampai dewasa dalam tubuh sapi. Namun, serangan cacing pita yang paling umum ditemukan pada sapi terutama oleh genus *Taenia*, yaitu *Taenia saginata*. Serangan cacing pita ini tidak berbahaya bagi ternak sapi itu sendiri karena dalam tubuh sapi telur cacing yang termakan bersama rumput hanya berkembang sampai fase larva. Larva cacing *T. saginata* yang berada dalam usus sapi selanjutnya akan menembus pembuluh darah dan ikut bersama aliran darah hingga sampai di otot. Selanjutnya, manusia perlu waspada terhadap serangan cacing pita ini, karena larva yang termakan dari daging sapi mentah atau yang dimasak kurang matang dapat berkembang menjadi cacing dewasa dalam usus halus manusia. Cacing pita dewasa akan menyerap sari-sari makanan dalam usus dan dapat menyebabkan penyumbatan usus.



Gambar 2.6 Cacing Pita[16]

Panjang cacing *T. saginata* dewasa berkisar antara 4 – 8 meter dan terdiri atas segmen-segmen yang disebut *proglotida*. *Proglotida* yang telah matang, atau disebut juga *proglotida gravid*, pada cacing dewasa berisi alat reproduksi jantan dan betina serta puluhan ribu telur. Bisa dibayangkan betapa banyaknya telur yang dihasilkan oleh 1 ekor cacing pita dewasa yang

selanjutnya siap masuk kembali kedalam tubuh sapi untuk berkembang menjadi bentuk yang siap masuk ke dalam tubuh manusia[17]

2.2.5 Ciri-ciri fisik sapi yang terkena serangan parasit cacing pita

Sistem pencernaan adalah sebuah sistem yang terdiri dari saluran pencernaan yang dilengkapi dengan beberapa organ yang bertanggung jawab atas pengambilan, penerimaan, dan pencernaan bahan makanan dalam perjalanannya melalui tubuh (saluran pencernaan) mulai dari rongga mulut sampai ke kloaka. Salah satu organ sistem pencernaan ternak *ruminisia* yaitu usus halus dan usus besar(*Caecum* dan *Colon*)[18]. Soulsby (1986) menyebutkan jenis cacing saluran pencernaan yang sering menyerang tenak sapi berasal dari kelas *Nematoda*, *Cestoda*, dan *Trematoda*[9]. *Taeniasis* merupakan penyakit yang disebabkan oleh cacing pita (*cestoda*) dari *genus Taenia*[4]. Gejala sapi terinfeksi cacing parasit usus yaitu kondisi menurun, pertumbuhan lambat, berat badan turun, kotoran encer/diare, demam[5].

Berikut ini adalah beberapa contoh gambar sapi yang kurus akibat cacing pita :



Gambar 2.7 Sapi kelihatan tulang rusuk akibat cacing pita[19]



Gambar 2.8 Sapi yang terkena serangan cacing pita[20]

Berikut ini adalah beberapa contoh sapi yang sehat:



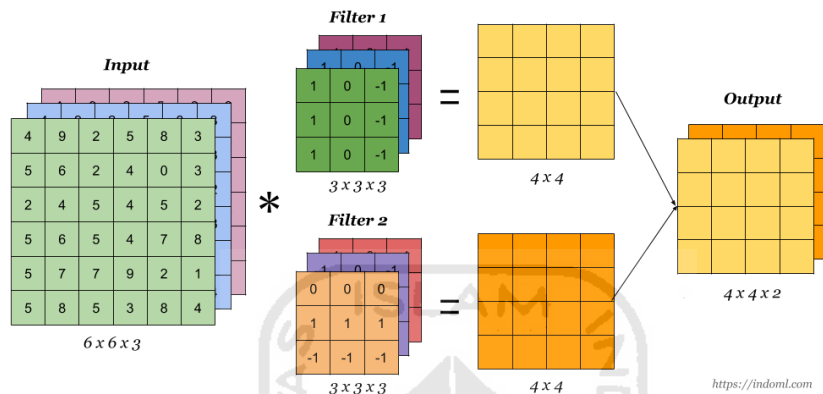
Gambar 2.9 Sapi sehat terlihat gemuk[22]



Gambar 2.10 Sapi gemuk yang sehat[23]

2.2.6 Convolutional Neural Network

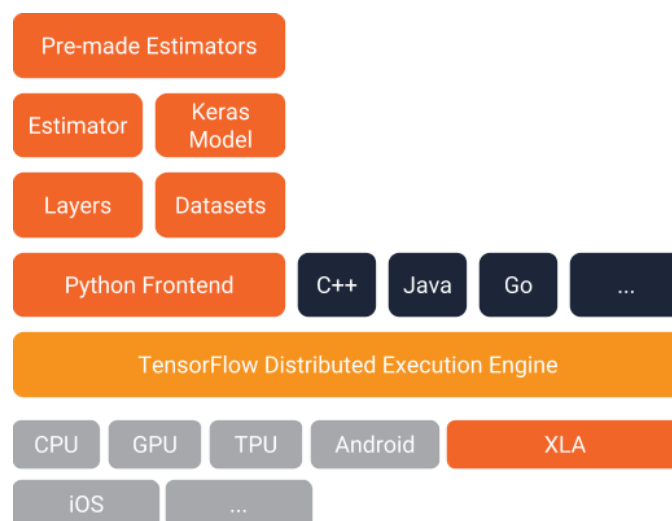
Convolutional Neural Network (CNN) adalah salah satu jenis *neural network* yang biasa digunakan pada data gambar. *CNN* dapat digunakan untuk mendeteksi dan mengenali objek pada sebuah gambar. *CNN* terdiri dari neuron yang memiliki berat, bias dan fungsi aktivasi[24]. Salah satu *library* yang menggunakan metode *CNN* yaitu *MobileNet-SSD-V1*.



Gambar 2.11 Ilustrasi *Convolutional Neural Network*[25]

2.2.7 Tensorflow

Tensorflow adalah sebuah sistem *machine learning* yang beroperasi pada skala yang besar dan berada pada area heterogen. *Tensorflow* menggunakan grafik aliran data untuk merepresentasikan sebuah komputasi[26].



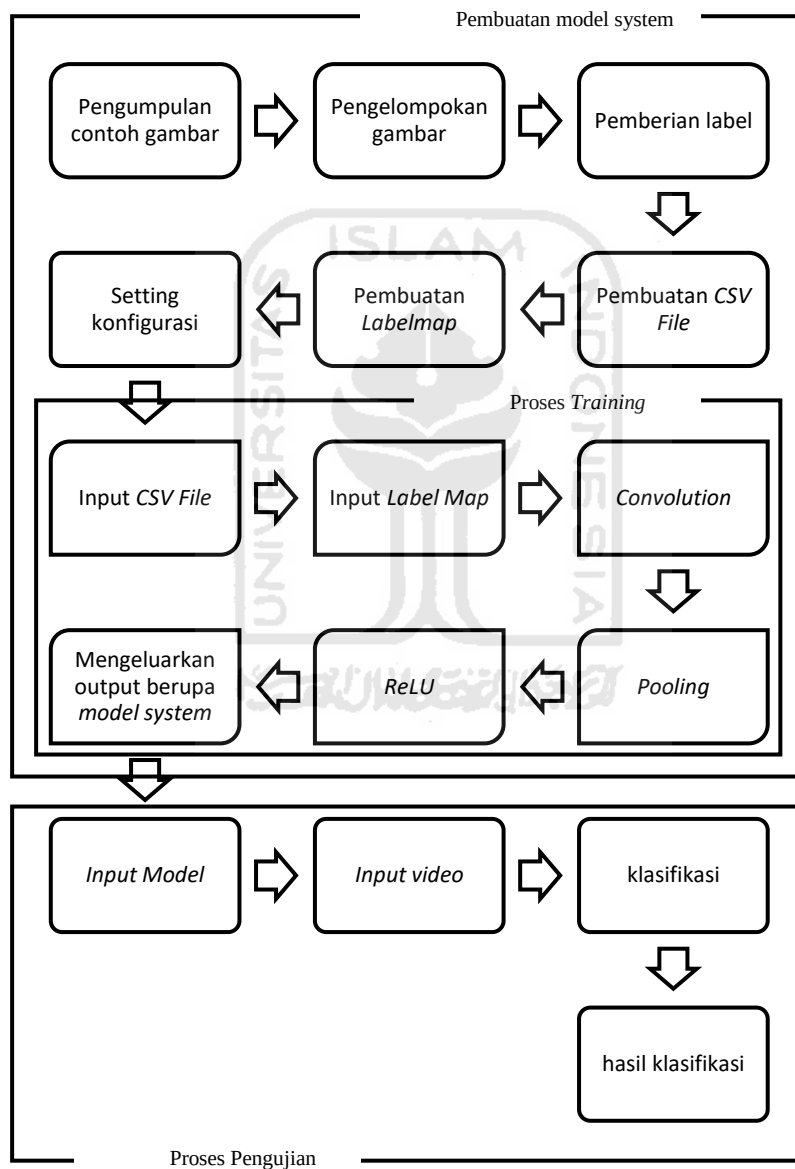
Gambar 2.12 Posisi *tensorflow* pada *hardware*[27]

BAB 3

METODOLOGI

3.1 Perancangan Sistem

Bagian perancangan sistem digunakan untuk melakukan klasifikasi terhadap sapi yang mempunyai ciri-ciri fisik terkena serangan parasit cacing pita. Tahapan umum yang dilakukan yaitu sebagai berikut :

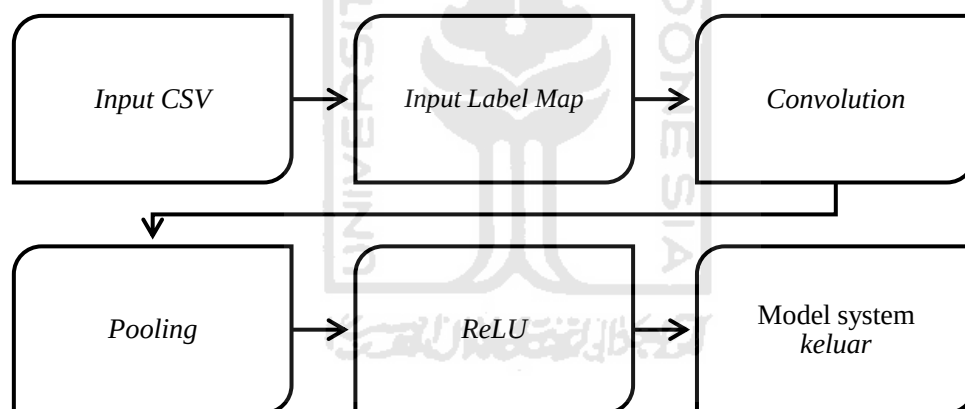


Gambar 3.1 Tahapan perancangan sistem

Pada Gambar 3.1 dapat dilihat secara umum tahapan pada perancangan sistem terbagi menjadi dua tahapan yaitu pembuatan model sistem lalu pengujian. Pada tahapan pembuatan model sistem, didalamnya terdapat beberapa tahapan yang disebut dengan proses *training*. Tahapan pembuatan model data adalah suatu tahapan yang berfungsi untuk membuat model sistem yang akan digunakan untuk implementasi sistem. Sedangkan model sistem sendiri adalah suatu bentuk data yang dihasilkan dari proses penyimpanan beberapa contoh gambar yang akan dijadikan pembanding dalam melakukan klasifikasi suatu gambar ketika implementasi maupun pengujian. Sedangkan tahapan pengujian adalah suatu tahapan yang dilakukan untuk menilai sistem yang dirancang terhadap tujuan penelitian termasuk dari segi tingkat keakuratan.

3.2 Training System

Training system merupakan bagian dari proses pembuatan model sistem yang memiliki sub proses tersendiri. Berikut ini adalah sub proses dari *training* sistem:



Gambar 3.2 Alur *training model system*

Pada gambar 3.2 dapat dilihat alur dari proses *training* model seperti berikut:

1. Input CSV

Input *Comma Separated Values(CSV) file* adalah sebuah proses untuk memasukkan *CSV file* ke dalam sistem:

	A	B	C	D	E	F	G	H
1	filename	width	height	class	xmin	ymin	xmax	ymax
2	10.jpg	800	600	sapi_sehat	31	97	595	600
3	11.jpg	800	600	sapi_sehat	153	138	745	512
4	12.jpg	800	600	sapi_sehat	181	200	693	589
5	12.jpg	800	600	sapi_sehat	12	15	531	504
6	13.jpg	800	600	sapi_sehat	21	3	721	575
7	16.jpg	800	600	sapi_sehat	115	178	684	502
8	17.jpg	800	600	sapi_sehat	34	67	755	542
9	18.jpg	800	600	sapi_sehat	187	38	790	600
10	18.jpg	800	600	sapi_sehat	11	13	363	394
11	19.jpg	800	600	sapi_cacingan	81	38	508	557

Gambar 3.3 CSV File

Gambar 3.3 terlihat data yang ada di dalam *CSV File*. Pada proses memasukkan *CSV File* terjadi pembacaan data oleh program pada *CSV File* untuk disimpan ke dalam variabel lokal pemrograman agar proses pengolahan menjadi lebih cepat.

2. Input Label Map

Pada proses memasukkan *label map* juga sama seperti proses memasukkan *CSV File*:

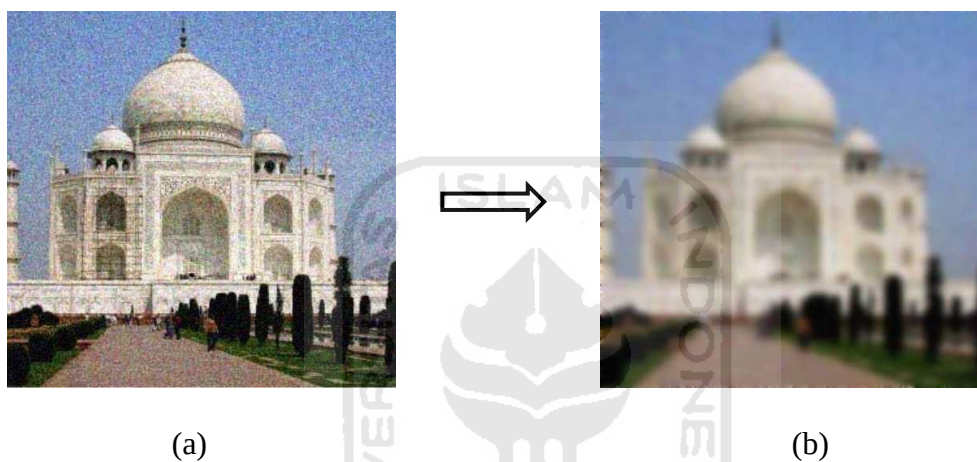
```
≡ labelmap.pbtxt X
C: > Users > 007 > Documents > Tensorflow > work
1  item {
2      id: 1
3      name: 'sapi_sehat'
4  }
5
6  item {
7      id: 2
8      name: 'sapi_cacingan'
9  }
```

Gambar 3.4 Label Map Data

Pada gambar 3.4 dapat dilihat bahwa *label map data* yang berisi daftar dari label yang akan dibuat pada sapi.

3. Convolution

Convolution adalah sebuah proses dimana citra dimanipulasi dengan menggunakan *external mask/subwindows* untuk menghasilkan citra yang baru[28]. Salah satu fungsi dari konvolusi yaitu memperhalus gambar seperti ilustrasi berikut:

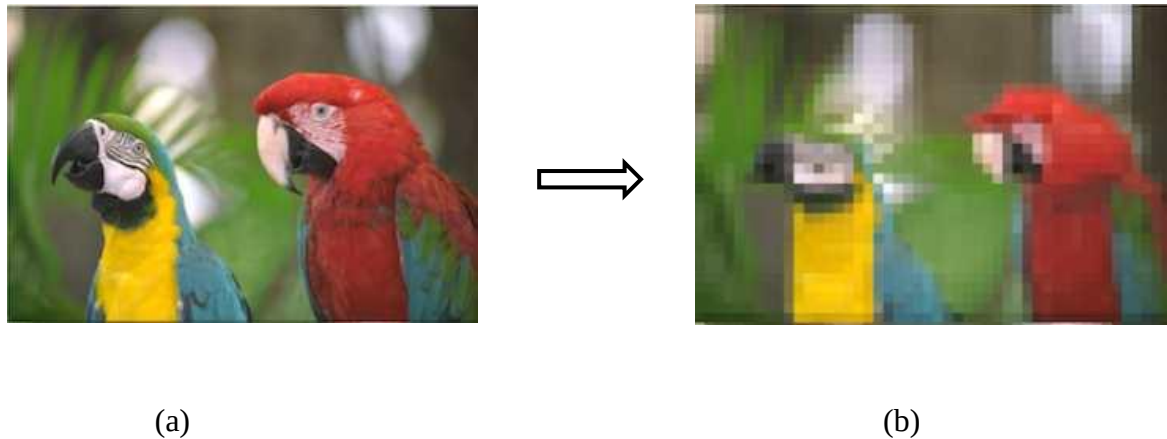


Gambar 3.5 Ilustrasi proses konvolusi[29]

Pada gambar 3.5 merupakan ilustrasi dari proses konvolusi. Dapat dilihat bahwa pada gambar dengan keterangan (a) adalah gambar sebelum dilakukan proses konvolusi dan memiliki *noise*, sedangkan gambar dengan keterangan (b) adalah gambar yang sudah dilakukan proses konvolusi untuk memperhalus gambar. Namun efek samping dari proses *smooth convolution* ini menyebabkan gambar menjadi *blur*[29].

4. *Pooling*

Pooling merupakan sebuah proses yang berfungsi untuk mereduksi ukuran citra [30]:

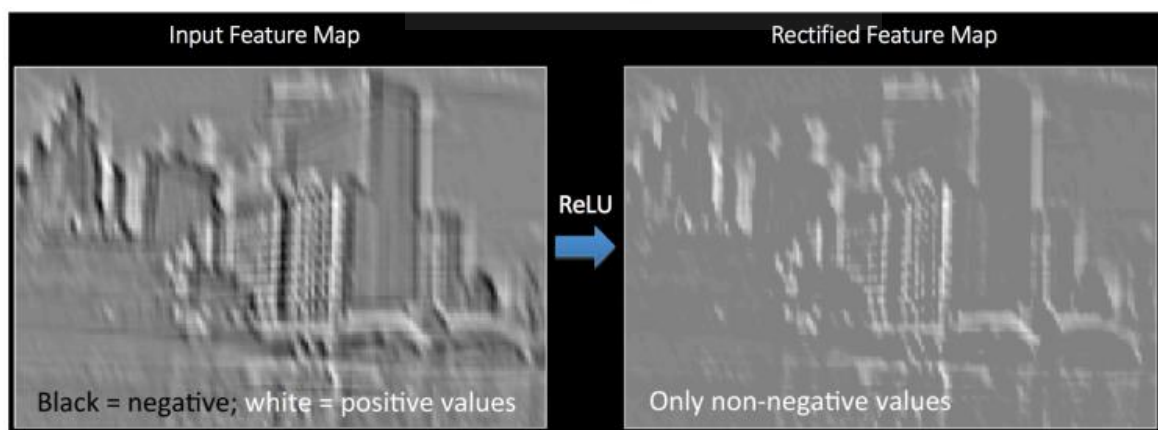


Gambar 3.6 Ilustrasi proses *pooling*[31]

Pada gambar 3.6 dapat dilihat bahwa proses *pooling* menyebabkan gambar menjadi pecah karena ukuran yang masih sama dengan gambar sebelum dilakukan proses *pooling* akibat dari penurunan resolusi gambar. *Pooling* berfungsi untuk mengurangi kebutuhan komputasi karena terjadi proses *resize* pada gambar menjadi ukuran yang lebih kecil[32].

5. *Rectified Linear Unit (ReLU)*

Proses kelima yaitu aktivasi *Rectified Linear Unit (ReLU)*. *ReLU* merupakan lapisan aktivasi pada model. Fungsi ini melakukan *thresholding* dengan nilai nol terhadap nilai piksel pada *input* citra. Fungsinya yaitu untuk memperjelas gambar agar mudah di deteksi.



Gambar 3.7 Ilustrasi proses *ReLU*[33]

Pada gambar 3.7 dapat dilihat gambar *input* sangat buram dan setelah dilakukan proses *ReLU*, gambar terlihat lebih baik. Perbedaan pada gambar tersebut ada pada garis yang berwarna

hitam atau bernilai negatif. Pada proses ReLU, semua nilai negatif akan diubah menjadi 0. Sehingga nilai *thresholding* adalah 0 seperti pada ilustrasi berikut :

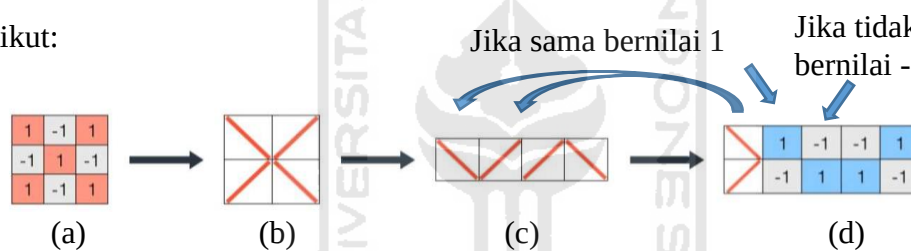
Rectified Linear Units (ReLUs)



Gambar 3.8 Ilustrasi perubahan nilai negatif menjadi 0[34]

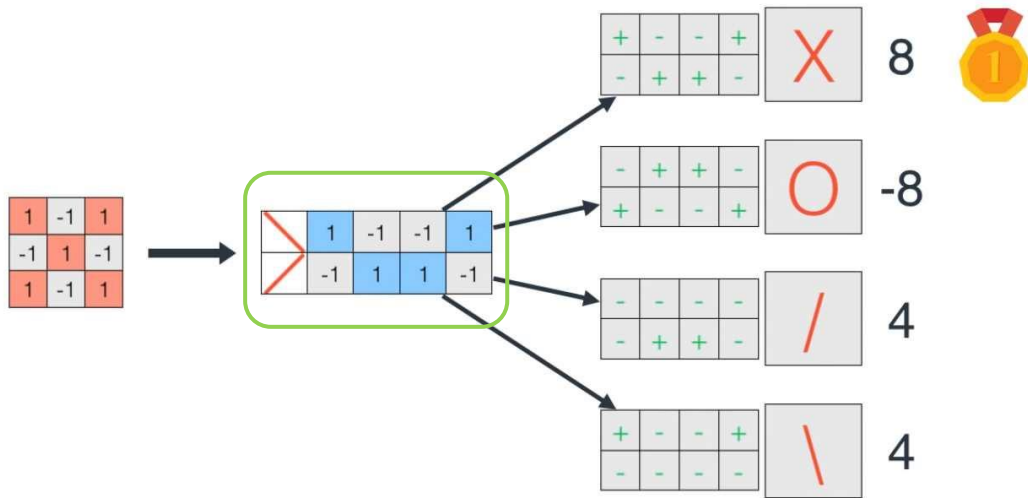
6. Model

Proses terakhir yaitu mengeluarkan output model. Model yang dibentuk dapat diilustrasikan sebagai berikut:



Gambar 3.9 Ilustrasi model yang dihasilkan[35]

Pada gambar 3.9 Dapat bisa dilihat proses ilustrasi pembentukan model yaitu bermula dari gambar yang dikonversikan menjadi nilai piksel pada setiap warna yaitu pada gambar dengan keterangan (a). Setelah itu gambar diilustrasikan dengan gambar huruf X dengan matrik 2×2 yaitu pada gambar dengan keterangan (b). Gambar dengan keterangan (c) adalah gambar sebelumnya yang di proses dari gambar dengan matrik 2×2 menjadi 1×4 , karena dijadikan satu baris semua, dan matrik pada baris selanjutnya dipindahkan ke sebelah kanan matrik pada baris satu. Proses yang terakhir yaitu menentukan nilai atau jenis matrik yang berbeda, pada contoh ini terdapat nilai atau jenis matrik yang berbeda, yaitu jenis matrik \ dan /. Setelah diketahui nilai atau jenis matrik yang berbeda. Maka dilakukan pengujian untuk menentukan sama atau tidaknya matrik yang berbeda tadi terhadap gambar dengan keterangan gambar (c). Jika sama diberi nilai 1 jika tidak sama diberi nilai -1.



Gambar 3.10 Ilustrasi pengujian dan penentuan nilai keakuratan[36]

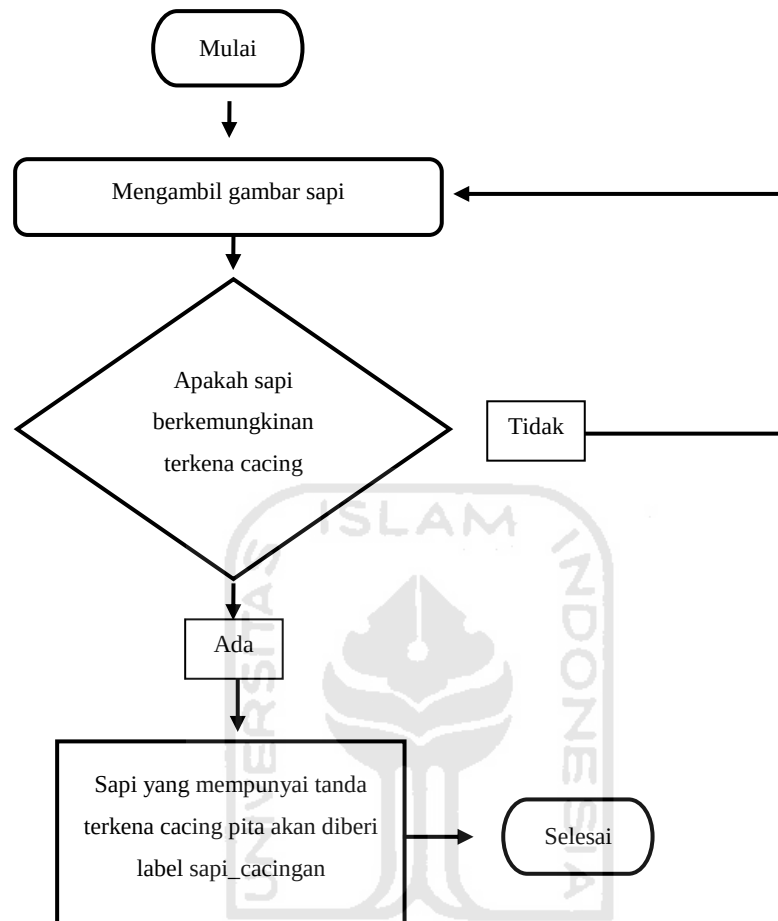
Pada gambar 3.10 dapat dilihat pada ilustrasi gambar masukan adalah gambar X, setelah itu dilakukan pembuatan modelnya dan setelah dilakukan pengujian dengan berbagai model yang sudah ada sebelumnya pada saat *training*. Model yang mempunyai akurasi tertinggi tingkat kecocokannya dengan model gambar yang diuji adalah model untuk gambar X. Sehingga pada pengujian ini nilai tingkat kesamaan model tertinggi dimiliki oleh model untuk gambar X.

Rumus atau persamaan untuk mengukur *system performance*:

$$\frac{\text{Jumlah gambar yang benar terdeteksi}}{\text{Jumlah seluruh gambar}} \times 100\% \quad (3.2)$$

3.3 Implementasi Sistem

Berikut ini adalah diagram alur logika pada saat implementasi sistem yang dirancang:



Gambar 3.11 Alur logika proses klasifikasi sapi yang memiliki ciri-ciri terkena serangan parasit cacing pita

Langkah terakhir yaitu melakukan pengujian seperti pada Gambar 3.11 dengan cara menjalankan program analisis yang telah dibuat yaitu mengambil gambar dan mendeteksi objek sapi pada gambar, mendeteksi ciri-ciri fisik penyakit cacingan pada sapi dan langsung membuat label pada sapi yang terkena cacingan ataupun yang sehat, namun akan ada sapi yang tidak terlabel karena tingkat pendeteksian objek di bawah 50%.

Berikut ini adalah contoh gambar sapi yang berada di Kecamatan Prambanan, Kabupaten Sleman:



Gambar 3.12 Sapi yang berada di Kecamatan Prambanan, Kabupaten Sleman

Berikut ini adalah contoh gambar sapi yang berada di Kecamatan Kalibawang, Kabupaten Kulon Progo:

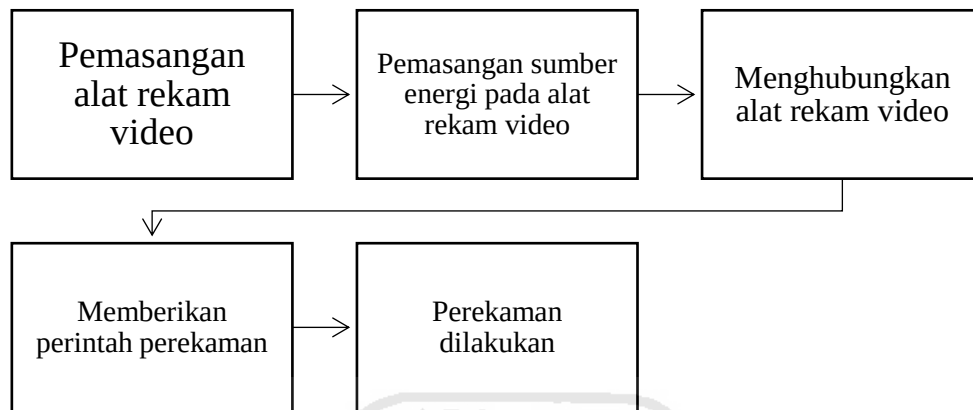


Gambar 3.13 Sapi yang berada di Kecamatan Kalibawang, Kabupaten Kulon Progo

3.4 Metode Pengujian

Untuk tahap pengujian dilakukan di dua tempat yaitu di Kecamatan Prambanan, Kabupaten Sleman dan Kecamatan Kalibawang Kabupaten Kulon Progo. Jumlah sapi yang dilakukan pengujian alat berjumlah 22 ekor. Spesifikasi sapi yang digunakan adalah sapi bali dengan bobot ideal yaitu 250 – 350 Kg berwarna putih dan juga berwarna coklat. Pengujian dilakukan pada waktu yang beragam, mulai dari pagi hari, siang hari, hingga sore hari. Sapi yang diuji juga memiliki kondisi yang beragam, baik dari segi umur, keturunan dan lain-lain.

Persiapan yang dilakukan sebelum melakukan pengujian yaitu menyiapkan alat rekam video, *powerbank* sebagai sumber energi untuk alat rekam telepon pintar yang berfungsi sebagai memberikan perintah pada alat rekam untuk memulai perekaman dan setelah itu melakukan analisis.



Gambar 3.14 Alur persiapan pengujian

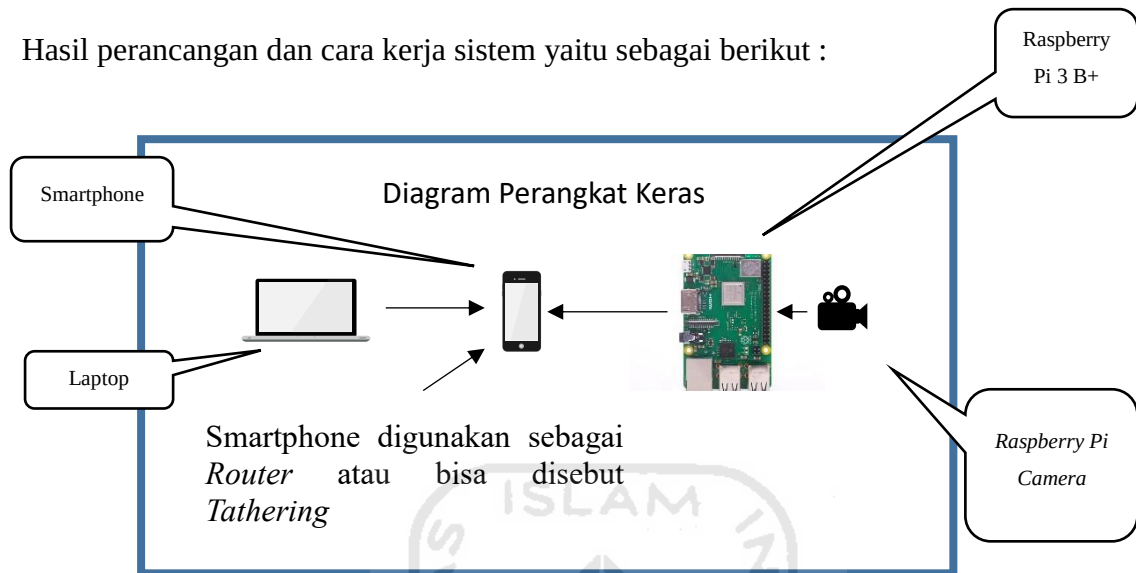
Pada Gambar 3.14 dapat dilihat alur persiapan pengujian yaitu dimulai dari pemasangan alat rekam video pada lokasi perekaman. Tahap kedua yaitu pemasangan sumber energi pada alat rekam video. Tahap ketiga yaitu menghubungkan alat rekam video ke telepon pintar melalui jaringan tanpa kabel atau yang biasa disebut dengan *wireless fidelity*. Tahap selanjutnya yaitu memberikan perintah kepada alat rekam video untuk memulai perekaman dan perekaman dilakukan.

BAB 4

HASIL DAN PEMBAHASAN

4.1 Hasil perancangan dan cara kerja sistem

Hasil perancangan dan cara kerja sistem yaitu sebagai berikut :



Gambar 4.1 Diagram sistem yang dirancang

Pada Gambar 4.1 dapat dilihat diagram sistem yang dirancang terdiri dari empat komponen yaitu laptop, ponsel pintar, *Raspberry Pi* dan juga kamera. Laptop berfungsi untuk pembuatan model data dan juga kontrol terhadap *Raspberry Pi*. Ponsel pintar berfungsi sebagai *hotspot* untuk menghubungkan antara *Raspberry Pi* dengan laptop. *Raspberry Pi* berfungsi sebagai pemroses data untuk klasifikasi pada gambar yang ditangkap oleh kamera. Dan kamera sendiri adalah suatu alat yang digunakan untuk menangkap gambar yang akan diproses oleh *Raspberry Pi*.

4.2 Pengujian gambar sapi pada berbagai sudut dan akurasi

Fungsi dari pengujian sistem dengan data *training* ini adalah untuk melihat keakuratan model yang dibentuk untuk mendeteksi keadaan sapi, jika nilai keakuratannya rendah, maka proses *training* akan dilanjutkan untuk menghasilkan model yang lebih baik. Pengujian yang pertama dilakukan dengan menggunakan *training data* pada *test folder*.

Tabel 4.1 Tabel keakuratan analisis berdasarkan sudut

Sudut gambar yang dianalisis	Jumlah Benar	Jumlah Salah	Persentase keakuratan
Depan	5	3	62,5%
Samping	4	11	26,6%
Belakang	2	5	28,5%

Berdasarkan pada tabel 4.1 dapat dilihat bahwa tingkat keakuratan tertinggi dalam analisis yaitu pada saat pengambilan gambar dari depan yaitu sebesar 62,5%. Berikut ini adalah beberapa contoh gambar pengujian yang dilakukan dari berbagai sudut yang sebagian lagi ada pada lampiran



Gambar 4.2 Sapi terinfeksi cacing pita yang diuji dengan data training[19]



Gambar 4.3 Sapi sehat yang diuji dengan data training[37]

Pada gambar 4.3 dapat dilihat bahwa gambar sapi sehat yang berada pada *training folder* saat dilakukan pengujian mendapatkan hasil yaitu sapi terdeteksi sehat, sedangkan pada gambar 4.2 dapat dilihat bahwa sapi yang terserang cacing pita terlihat terdeteksi dengan benar. Tingkat keakurasian pada pengujian menggunakan data *training* yaitu sebagai berikut :

Tabel 4.2 Tabel hasil pengujian dari klasifikasi dengan data *training*

		<i>Predict Label</i>		
<i>Actual Label</i>		Sapi tanpa gejala	Sapi dengan gejala	Objek Lain
	Sapi tanpa gejala	8	2	2
	Sapi dengan gejala	2	3	0
	Objek Lain	7	8	11

Pada Tabel 4.2 dapat dilihat bahwa sistem yang dirancang secara dominan benar mendeteksi suatu objek yang dapat dilihat dari tingkat kecocokan antar *Predict Label* dan *Actual Label* yang mendapatkan nilai tertinggi. Tabel ini merupakan tabel hasil analisa gambar sapi dari berbagai macam sudut. Seperti pada persamaan 3.2 maka dari tabel 4.2 bisa didapatkan nilai *system performance* atau tingkat keakuratannya yaitu :

$$\frac{(8 + 3 + 11)}{8 + 2 + 2 + 2 + 3 + 0 + 7 + 8 + 11} \times 100\% = 51,16\%$$

4.3 Pengujian pada sistem *real time* dan akurasi

Pengujian ini berfungsi untuk memastikan tujuan penelitian sesuai dengan sistem yang dibuat dengan cara *scanning object* secara langsung.



Gambar 4.4 Sapi terdeteksi dengan sempurna

Pada gambar 4.4 Terlihat bahwa komputer dapat mendeteksi sapi yang memiliki ciri-ciri fisik terkena serangan parasit cacing pita dengan label “**sapi_cacingan**” dan juga sapi yang tidak memiliki ciri-ciri fisik terkena serangan parasit cacing pita dengan label “**sapi_sehat**”. Hal ini adalah kondisi ideal pada penelitian ini dimana komputer dapat mendeteksi setiap sapi dan dapat menentukan bahwa sapi tersebut memiliki ciri-ciri fisik terkena serangan parasit cacing pita atau tidak.



Gambar 4.5 Sapi yang tidak memiliki ciri-ciri fisik terkena serangan parasit cacing pita

Pada gambar 4.5 dapat dilihat bahwa komputer dapat menganalisis sapi yang tidak memiliki ciri-ciri fisik terkena serangan parasit cacing pita. Sedangkan untuk sapi yang tidak terdeteksi di sebelah kiri dikarenakan terhalang oleh kayu dan juga rendahnya tingkat variasi gambar rusak pada contoh gambar yang dilakukan proses *train model* sehingga model yang dibuat tidak mampu untuk mendeteksi objek yang terhalang oleh objek lain. Untuk hasil analisis sistem lainnya dapat dilihat pada lampiran.

Tabel 4.3 Tabel kemampuan deteksi

Kondisi	Status Pendeteksian
Sapi berwarna coklat	Dapat dideteksi
Sapi berwarna putih	Dapat dideteksi
Sapi sehat	Dapat dideteksi
Sapi yang memiliki ciri-ciri fisik terkena serangan parasit cacing pita	Dapat dideteksi
Sapi lebih dari satu ekor	Dapat dideteksi
Sapi membelakangi kamera	Dapat dideteksi

Setelah melakukan pengujian, didapatkan hasil dari seperti pada tabel 4.3, dari hasil tersebut, didapatkan kesimpulan bahwa sistem dapat mendeteksi berbagai warna pada sapi dan juga dapat mendeteksi sapi yang memiliki ciri-ciri fisik terkena serangan parasit cacing pita seperti pada tujuan penelitian. Yang kedua, yaitu sistem dapat mendeteksi lebih dari satu sapi dan menunjukkan sapi yang memiliki ciri-ciri fisik terkena serangan parasit cacing pita seperti pada tujuan penelitian. Objek sapi juga dapat dideteksi bahkan jika dalam posisi membelakangi kamera. Sedangkan untuk tingkat keakurasian pendeteksian sapi yang memiliki gejala fisik terkena serangan parasit cacing sapi yaitu sebagai berikut:

Tabel 4.4 Tabel hasil pengujian dari klasifikasi secara *real time*

		Predict Label		
Actual Label		Sapi tanpa gejala	Sapi dengan gejala	Objek Lain
	Sapi tanpa gejala	15	12	3
	Sapi dengan gejala	3	6	0
	Objek Lain	3	3	40

Pada Tabel 4.4 dapat dilihat bahwa sistem yang dirancang untuk klasifikasi secara *real time* secara dominan benar mendeteksi suatu objek yang dapat dilihat dari tingkat kecocokan antar *Predict Label* dan *Actual Label* yang mendapatkan nilai tertinggi. Seperti pada persamaan 3.2 maka dari tabel 4.4 bisa didapatkan nilai *system performance* atau tingkat keakuratannya yaitu :

$$\frac{(15 + 6 + 40)}{15 + 12 + 3 + 3 + 6 + 0 + 3 + 3 + 40} \times 100\% = 71,76\%$$

Ketika melakukan pengujian terhadap dua ekor objek sapi, terkadang sistem tidak mampu mendeteksi dengan tingkat estimasi lebih dari 50%, hal ini terbukti dengan objek sapi yang terkadang tidak dilabel oleh sistem, sehingga menyebabkan pengujian harus menunggu waktu yang lebih lama agar sapi berada posisi yang lebih mudah dikenali oleh komputer sebagai objek sapi pada video. Proses analisis pada sapi juga terkendala atau yang biasa disebut dengan istilah *lag* dikarenakan proses analisis membutuhkan proses kinerja komputasi yang tinggi. Proses analisis ini juga tidak bisa berlangsung lama jika komputer yang digunakan tidak memiliki kipas pendingin untuk pengolah data atau yang disebut dengan *processor*. Proses analisis pada pengujian ini tidak membutuhkan koneksi internet maupun koneksi dengan telepon pintar karena dilakukan pada komputer bukan pada *cloud system*.

4.4 Pembahasan

Pada tahapan pengujian analisis potensi sapi terkena penyakit cacingan, didapat hasil yaitu tingkat keakuratan tertinggi ketika analisis yaitu pada saat gambar sapi diambil dari arah depan. Berdasarkan analisis penulis, hal ini dapat disebabkan karena gambar yang digunakan untuk *training* banyak diambil dari arah depan dan juga karena gambar dari arah depan baik untuk *training* maupun *testing* tidak murni dari arah depan seluruhnya, tetapi tubuhnya juga sedikit ke arah samping sehingga juga terlihat bagian tubuh sapi. Sedangkan untuk tingkat cahaya, hampir rata-rata gambar untuk *training* maupun *testing* dalam memiliki keadaan cahaya yang normal pada siang hari yaitu 29950 lux[7]. Sedangkan berdasarkan literatur terkait, yaitu seperti contoh pada sebuah literatur yang berjudul “*Deep Image: Scaling up Image Recognition*” menyatakan bahwa resolusi gambar pada saat *training* juga mempengaruhi tingkat akurasi suatu deteksi, semakin tinggi resolusi gambar maka, tingkat akurasi analisis juga semakin tinggi[38].

Tetapi pada literatur lain yang berjudul “*Improving Iris Recognition Accuracy Via Cascaded Classifier*” menyatakan bahwa algoritma untuk melakukan *training process* juga mempengaruhi tingkat akurasi deteksi[39]. Dari beberapa literatur termasuk literatur yang telah disebutkan sebelumnya, penulis menyimpulkan bahwa banyak faktor yang mempengaruhi tingkat keakuratan suatu sistem deteksi yang mempunyai kelebihan dan kekurangannya masing-masing seperti contohnya dengan melakukan proses *training* pada gambar dengan resolusi tinggi, maka dibutuhkan *processing resource* yang tinggi pula ataupun waktu yang lebih lama untuk melakukan proses *training*.



BAB 5

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Pada penelitian ini didapatkan kesimpulan sebagai berikut:

1. Dari hasil penelitian menunjukkan bahwa teknologi dapat membantu peternak dalam mengurangi resiko sapi terhadap serangan parasit cacing pita
2. Sistem yang dirancang dapat mendeteksi sapi yang memiliki potensi terkena serangan parasit cacing pita, tetapi masih terdapat kesalahan ketika mendeteksi pada keadaan tertentu dimana pada sudut pandang dari depan, samping, dan belakang masing-masing sebesar 62,5%, 26,6%, dan 28,5%
3. Persentase keakuratan sistem *real time* dalam mendeteksi sapi sebesar 71,76%

5.2 Saran

Untuk pengembangan penelitian ini dapat melakukan seperti saran-saran berikut:

1. Menambahkan contoh gambar sapi pada posisi yang lebih banyak
2. Menambahkan contoh gambar sapi pada kondisi terhalang oleh berbagai objek
3. Menambah program masukkan pada komputer untuk menaikkan tingkat keakuratan pendeteksian sapi yang memiliki ciri-ciri fisik terkena serangan parasit cacing pita
4. Menambah contoh gambar sapi dari berbagai sudut

DAFTAR PUSTAKA

- [1] R. Ratnawati, N. Nurliana, and R. Razali, “Tingkat Kepuasan Konsumen Terhadap Kualitas dan Harga Daging Sapi Yang Dijual di Kota Banda Aceh,” *J. Agripet*, vol. 14, no. 2, pp. 125–131, 2014, doi: 10.17969/agripet.v14i2.1889.
- [2] i gede wisnu paramadita, “Prevalensi Cacing Saluran Pencernaan Sapi Perah,” *Ekp*, vol. 13, no. 3, pp. 1576–1580, 2017.
- [3] Tolistiawaty et.al, “Tingkat Infeksi Cacing Hati Kaitannya dengan Kerugian Ekonomi Sapi Potong yang Disembelih di Rumah Potong Hewan Wilayah Eks-Kresidenan Banyumas,” *Balaba*, vol. 12, no. 2, pp. 71–78, 2016.
- [4] N. Suriawanto *et al.*, “Deteksi Cacing Pita (*Taenia solium* L.) Melalui Uji Feses Pada Masyarakat Desa Purwosari Kecamatan Torue Kabupaten Parigi Moutong Sulawesi Tengah,” *J. Biocelbes*, vol. 8, no. 1, pp. 1978–6417, 2014.
- [5] S. N. Ahmad, “Kajian Efektivitas Pemberian Obat Cacing Herbal Terhadap Performa Sapi Potong,” *Pros. Semin. Nas. Inov. Teknol. Pertan. Spesifik Lokasi*, vol. 1, no. 1, pp. 503–509, 2014, [Online]. Available: http://kalsel.litbang.pertanian.go.id/ind/images/pdf/semnas2014/57_salfina.pdf.
- [6] Y. M, B. Adnyana, and M. Sukada, “Profil Umur, Jenis Kelamin, Berat Badan dan Jejas Eksternal pada Kulit Sapi Bali yang Disembelih di Rumah Potong Hewan Kota Denpasar Periode Mei-Juni,” *Indones. Med. Veterinus*, vol. 5, no. 4, pp. 376–387, 2016.
- [7] D. Arsitektur, F. Teknik, and U. Hasanuddin, “Analisis Intensitas Pencahayaan Alami Pada Ruang Pertemuan DI Gedung COT Fakultas Teknik Gowa Universitas Hasanuddin,” pp. 1–173.
- [8] K. P. Biro Humas dan Informasi Publik, “Dengan Program Ini Kementan Dongkrak Populasi Sapi Dalam Negeri Dan Turunkan Impor,” *Kementrian Pertanian*, 2018. .
- [9] I. K. Sari, P. Dan, D. Infeksi, C. Saluran, and P. Pada, “Prevalensi dan Derajat Infeksi Cacing Saluran Pencernaan Pada Sapi Peranakan Ongole (PO) dan Limosin di Kecamatan Tikung Kabupaten Lamongan,” 2014.
- [10] C. Wai Zhao, J. Jegatheesan, and S. Chee Loon, “Exploring IOT Application Using Raspberry Pi,” *Int. J. Comput. Networks Appl.*, vol. 2, no. 1, pp. 27–34, 2015, [Online]. Available: <http://www.digi.com>.
- [11] “Raspberry Pi 3 B+.” www.waveshare.com.
- [12] R. P. Foundation, “Camera Module V2.” <https://www.raspberrypi.org/products/camera-module-v2/>.

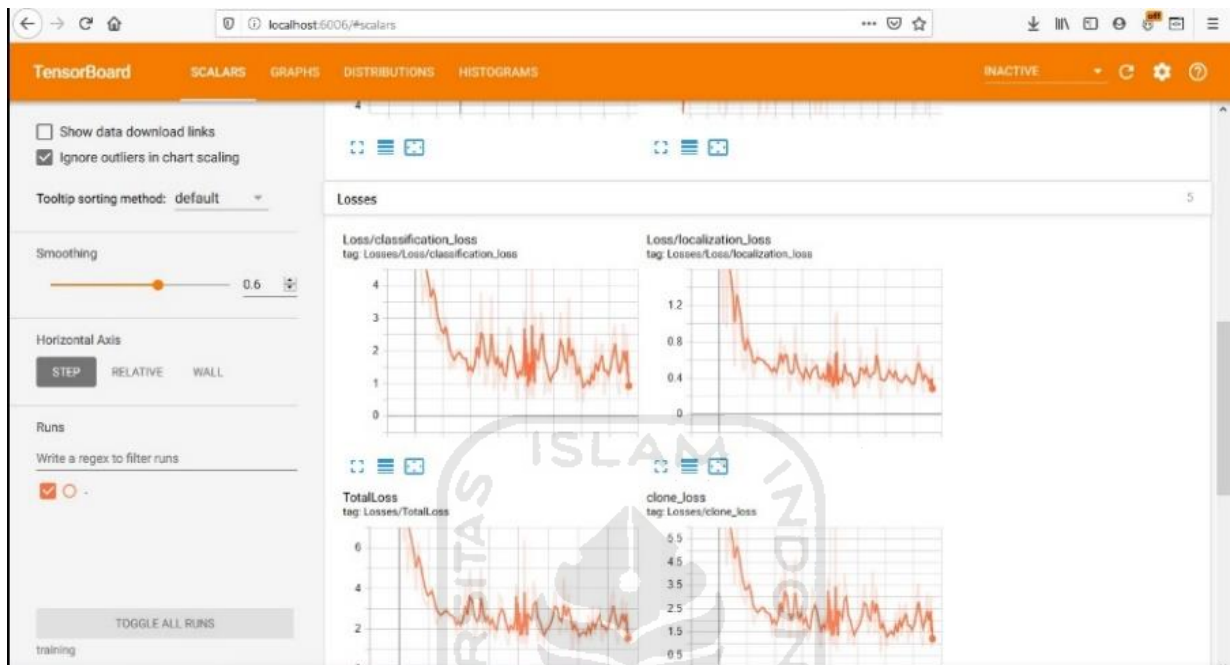
- [13] "A. Shop." www.alzashop.com.
- [14] "Linovhr." linovhr.com.
- [15] Y. Efendi, "Internet Of Things (Iot) Sistem Pengendalian Lampu Menggunakan Raspberry Pi Berbasis Mobile," *J. Ilm. Ilmu Komput.*, vol. 4, no. 2, pp. 21–27, 2018, doi: 10.35329/jiik.v4i2.41.
- [16] "Health Id." www.health.grid.id.
- [17] "Cacingan pada Sapi Jangan Dianggap Enteng," *Medion*, 2013. <https://www.medion.co.id/id/cacingan-pada-sapi-jangan-dianggap-enteng/>.
- [18] J. Oliver, *Nutrisi dan Pakan Ternak Ruminisia*, vol. 1, no. 2019.
- [19] "Medion." www.medion.co.id.
- [20] "Teman Petani." www.temanpetani.com.
- [21] "Unsur Tani." www.unsurtani.com.
- [22] "Pernakan Kita," [Online]. Available: www.peternakankita.com.
- [23] "Rumah Mesin." www.rumahmesin.com.
- [24] D. Snyder *et al.*, "Investigating the Use of Semi-Supervised Convolutional Neural Network Models for Speech/Music Classification and Segmentation," *ICASSP, IEEE Int. Conf. Acoust. Speech Signal Process. - Proc.*, vol. 20, no. 1, pp. 1–9, 2015, doi: 10.1017/CBO9781107415324.004.
- [25] "Indo ML." www.indoml.com.
- [26] J. D. Dignam, P. L. Martin, B. S. Shastry, and R. G. Roeder, "TensorFlow: A System for Large-Scale Machine Learning," *Methods Enzymol.*, vol. 101, no. C, pp. 582–598, 1983, doi: 10.1016/0076-6879(83)01039-3.
- [27] "Developers." www.developers.googleblog.com.
- [28] H. Sunandar, "Perbaikan kualitas Citra Menggunakan Metode Gaussian Filter," *MEANS (Media Inf. Anal. dan Sist.*, vol. 2, no. 1, pp. 19–22, 2017.
- [29] W. Gazali, H. Soeparno, and J. Ohliati, "Penerapan Metode Konvolusi Dalam Pengolahan Citra Digital," *J. Mat Stat*, vol. 12, pp. 103–113, 2012.
- [30] W. S. Eka Putra, "Klasifikasi Citra Menggunakan Convolutional Neural Network (CNN) pada Caltech 101," *J. Tek. ITS*, vol. 5, no. 1, 2016, doi: 10.12962/j23373539.v5i1.15696.
- [31] "Max Pooling." www.shortscience.org.
- [32] B. Graham, "Fractional Max Pooling," *HumanBrain*, 2014. https://humanbrain.gitbook.io/notes/notes/vision/fractional_max-pooling.
- [33] "Medium." www.medium.com.
- [34] "Brain Creator." www.slideshare.net.
- [35] "Machine Learning." www.machinelearning.mipa.ugm.ac.id.

- [36] “Intelligency.” www.intelligency.org.
- [37] “Referensi Bebas.” www.referensibebas.com.
- [38] R. Wu, S. Yan, Y. Shan, Q. Dang, and G. Sun, “Deep Image: Scaling up Image Recognition,” 2015, [Online]. Available: <http://arxiv.org/abs/1501.02876>.
- [39] Z. Sun, Y. Wang, T. Tan, and J. Cui, “Improving Iris Recognition Accuracy Via Cascaded Classifiers,” *IEEE Trans. Syst. MAN Cybern.*, vol. 35, no. 3, pp. 435–441, 2005.



LAMPIRAN

1. Hasil Deteksi Sapi



Gambar 5.1 Monitoring proses training

```
9022 09-04-03.731154 2956 Internal.py:113] :+ - [22/May/2020 09:04:03] "GET /
sta/plugin/scalars/scalars?tag=losses%3Floss%3Flocalization_loss&experiment= HTTP/1.1" 200 -
9022 09-04-03.739976 9466 Internal.py:114] :+ - [22/May/2020 09:04:03] "GET /
sta/plugin/scalars/scalars?tag=losses%3Fclone_loss&experiment= HTTP/1.1" 200 -
9022 09-04-03.749778 13286 Internal.py:113] :+ - [22/May/2020 09:04:03] "GET /
sta/plugin/scalars/scalars?tag=losses%3Fregularisation_loss&experiment= HTTP/1.1" 200 -
9022 09-04-03.758605 8840 Internal.py:113] :+ - [22/May/2020 09:04:03] "GET /
sta/plugin/scalars/scalars?tag=losses%3Fhard_example_miner%3Fnum_positives&experiment= HTTP/1.1" 200 -
9022 09-04-03.767432 10160 Internal.py:113] :+ - [22/May/2020 09:04:03] "GET /
sta/plugin/scalars/scalars?tag=losses%3Floss%3Fclassification_loss&experiment= HTTP/1.1" 200 -
9022 09-04-03.776259 9466 Internal.py:113] :+ - [22/May/2020 09:04:03] "GET /
sta/plugin/scalars/scalars?tag=losses%3Floss%3Flocalization_loss&experiment= HTTP/1.1" 200 -
9022 09-04-03.785087 13286 Internal.py:113] :+ - [22/May/2020 09:04:03] "GET /
sta/plugin/scalars/scalars?tag=losses%3Fhard_example_miner%3Fnum_positives&experiment= HTTP/1.1" 200 -
9022 09-04-03.793915 10160 Internal.py:113] :+ - [22/May/2020 09:04:03] "GET /
sta/plugin/scalars/scalars?tag=losses%3Floss%3Fclassification_loss&experiment= HTTP/1.1" 200 -
9022 09-04-03.802742 9466 Internal.py:113] :+ - [22/May/2020 09:04:03] "GET /
sta/plugin/scalars/scalars?tag=learning_rate%3Flearning_rate&experiment= HTTP/1.1" 200 -
9022 09-04-03.811569 8840 Internal.py:113] :+ - [22/May/2020 09:04:03] "GET /
sta/plugin/scalars/scalars?tag=losses%3Fhard_example_miner%3Fnum_positives&experiment= HTTP/1.1" 200 -
9022 09-04-03.820396 2956 Internal.py:113] :+ - [22/May/2020 09:04:03] "GET /
sta/plugin/scalars/scalars?tag=losses%3Floss%3Flocalization_loss&experiment= HTTP/1.1" 200 -
9022 09-04-03.829223 9466 Internal.py:113] :+ - [22/May/2020 09:04:03] "GET /
sta/plugin/scalars/scalars?tag=losses%3Fclone_loss&experiment= HTTP/1.1" 200 -
9022 09-04-03.838051 13286 Internal.py:113] :+ - [22/May/2020 09:04:03] "GET /
sta/plugin/scalars/scalars?tag=losses%3Fregularisation_loss&experiment= HTTP/1.1" 200 -
9022 09-04-03.846878 10160 Internal.py:113] :+ - [22/May/2020 09:04:03] "GET /
sta/plugin/scalars/scalars?tag=losses%3Fhard_example_miner%3Fnum_positives&experiment= HTTP/1.1" 200 -
9022 09-04-03.855705 9466 Internal.py:113] :+ - [22/May/2020 09:04:03] "GET /
sta/plugin/scalars/scalars?tag=losses%3Floss%3Flocalization_loss&experiment= HTTP/1.1" 200 -
9022 09-04-03.864532 8840 Internal.py:113] :+ - [22/May/2020 09:04:03] "GET /
sta/plugin/scalars/scalars?tag=losses%3Fclone_loss&experiment= HTTP/1.1" 200 -
```

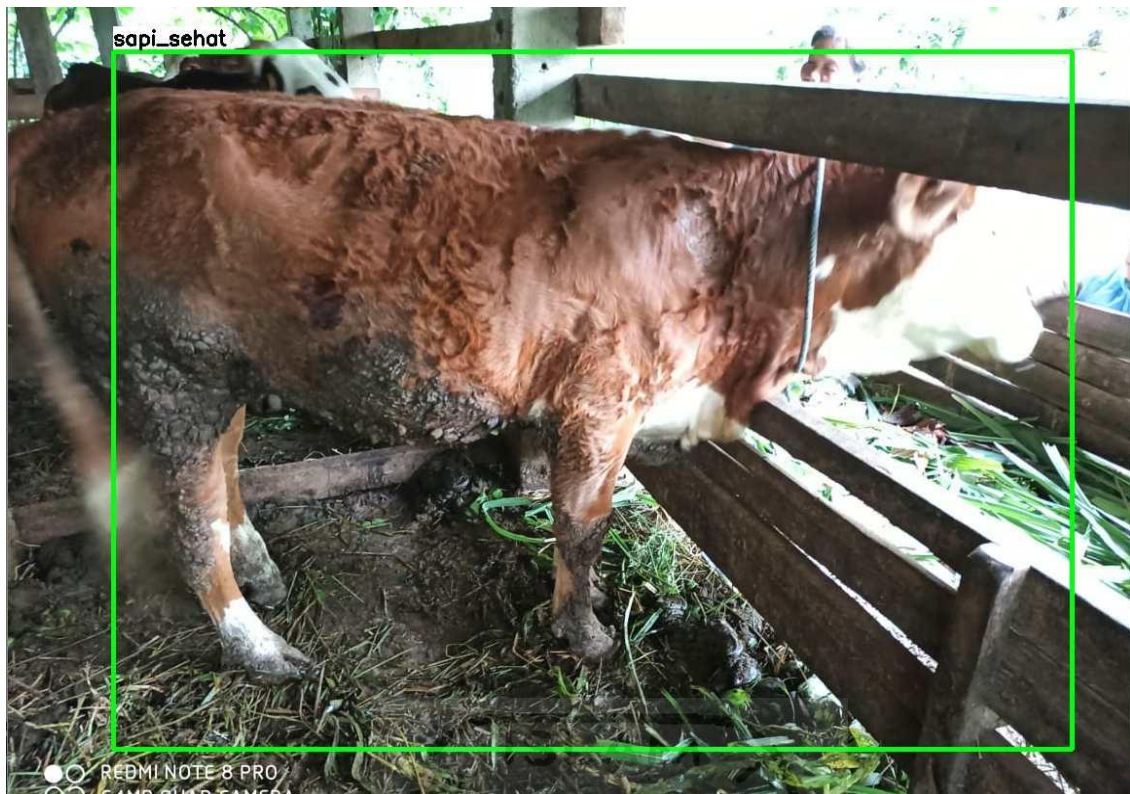
Gambar 5.2 Proses training pada CMD

```
0522 09:04:03.757154 2956 _internal.py:113] :1: - [22/May/2020 09:04:03] "GET /
sta/plugin/scalars/scalars/tag-losses%3Floss%3Flocalization_loss&run=&experiment=
HTTP/1.1" 200 -
0522 09:04:03.789576 9400 _internal.py:113] :1: - [22/May/2020 09:04:03] "GET /
sta/plugin/scalars/scalars/tag-losses%3Fclone_loss&run=&experiment= HTTP/1.1" 200
0522 09:04:03.789576 13200 _internal.py:113] :1: - [22/May/2020 09:04:03] "GET /
sta/plugin/scalars/scalars/tag-losses%3Fregularization_loss&run=&experiment= HTTP
1.1" 200 -
0522 09:04:32.386665 8840 _internal.py:113] :1: - [22/May/2020 09:04:32] "GET /
sta/environment HTTP/1.1" 200 -
0522 09:04:32.386665 10160 _internal.py:113] :1: - [22/May/2020 09:04:32] "GET /
sta/experiments HTTP/1.1" 200 -
0522 09:04:32.424404 13200 _internal.py:113] :1: - [22/May/2020 09:04:32] "GET /
sta/runs HTTP/1.1" 200 -
0522 09:04:33.543525 9400 _internal.py:113] :1: - [22/May/2020 09:04:33] "GET /
sta/plugins_listing HTTP/1.1" 200 -
0522 09:04:33.628051 9400 _internal.py:113] :1: - [22/May/2020 09:04:33] "GET /
sta/plugin/scalars/tags HTTP/1.1" 200 -
0522 09:04:33.928142 13200 _internal.py:113] :1: - [22/May/2020 09:04:33] "GET /
sta/plugin/scalars/scalars/tag-loss%3FhardExampleInner%3FnumPositives&run=&experim
ent= HTTP/1.1" 200 -
0522 09:04:33.922250 10160 _internal.py:113] :1: - [22/May/2020 09:04:33] "GET /
sta/plugin/scalars/scalars/tag-losses%3Floss%3Fclassification_loss&run=&experimen
t= HTTP/1.1" 200 -
0522 09:04:33.922200 9400 _internal.py:113] :1: - [22/May/2020 09:04:33] "GET /
sta/plugin/scalars/scalars/tag-learningRate%3FlearningRate%3Flearning_rate&run=&ex
periment= HTTP/1.1" 200 -
0522 09:04:33.922200 8840 _internal.py:113] :1: - [22/May/2020 09:04:33] "GET /
sta/plugin/scalars/scalars/tag-loss%3FhardExampleInner%3FnumNegatives&run=&experim
ent= HTTP/1.1" 200 -
0522 09:04:33.942542 2956 _internal.py:113] :1: - [22/May/2020 09:04:33] "GET /
sta/plugin/scalars/scalars/tag-losses%3Floss%3Flocalization_loss&run=&experiment=
HTTP/1.1" 200 -
0522 09:04:34.014061 10120 _internal.py:113] :1: - [22/May/2020 09:04:34] "GET /
sta/plugin/scalars/scalars/tag-losses%3FtotalLoss&run=&experiment= HTTP/1.1" 200
0522 09:04:34.022104 13200 _internal.py:113] :1: - [22/May/2020 09:04:34] "GET /
sta/plugin/scalars/scalars/tag-losses%3Fclone_loss&run=&experiment= HTTP/1.1" 200
0522 09:04:34.054503 10160 _internal.py:113] :1: - [22/May/2020 09:04:34] "GET /
sta/plugin/scalars/scalars/tag-losses%3Fregularization_loss&run=&experiment= HTTP
1.1" 200 -
```

Gambar 5.3 Penyedia server lokal



Gambar 5.4 Sapi yang terkena serangan parasit cacing namun salah deteksi



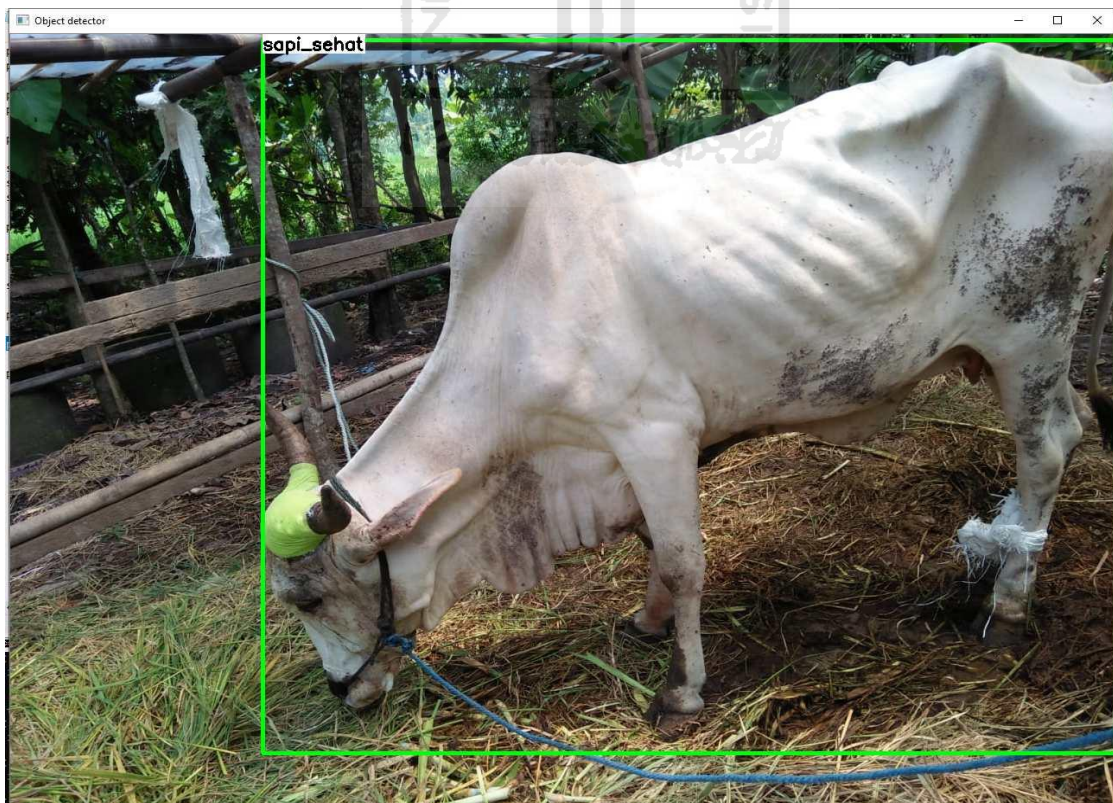
Gambar 5.5 Sapi yang terkena serangan parasit cacing namun salah deteksi



Gambar 5.6 Cacing yang terkena serangan parasit cacing dan terdeteksi benar



Gambar 5.7 Sapi yang terkena serangan parasit cacing namun salah deteksi



Gambar 5.8 Sapi yang terkena serangan parasit cacing namun salah deteksi



Gambar 5.9 Sapi yang terkena serangan parasit cacing namun salah deteksi



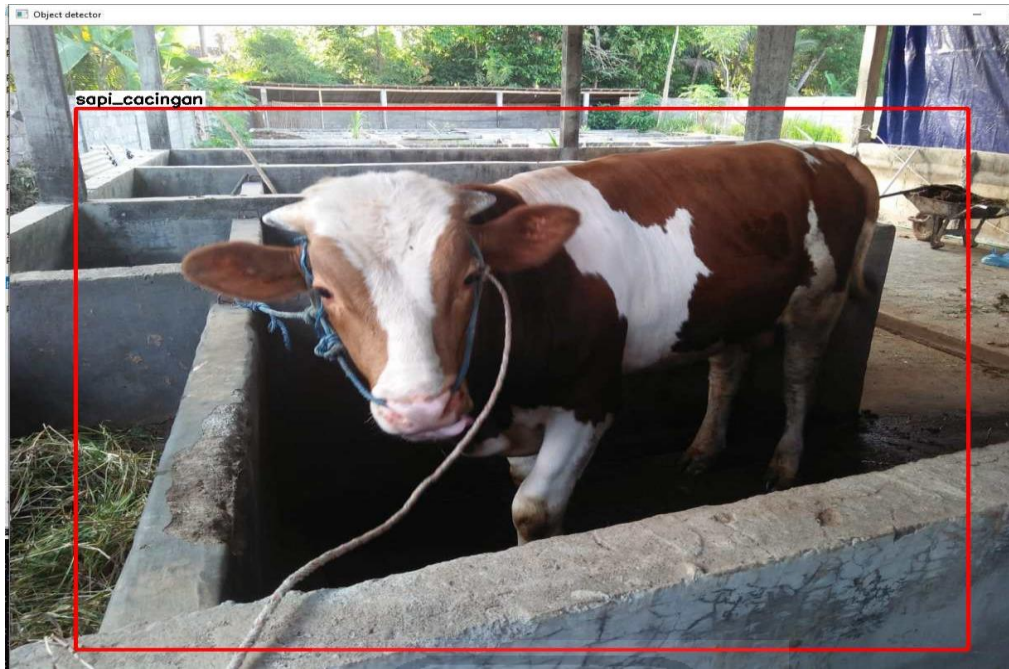
Gambar 5.10 Sapi yang tidak terkena serangan parasit cacing dan terdeteksi benar



Gambar 5.11 Sapi yang terkena serangan parasit cacing namun salah deteksi

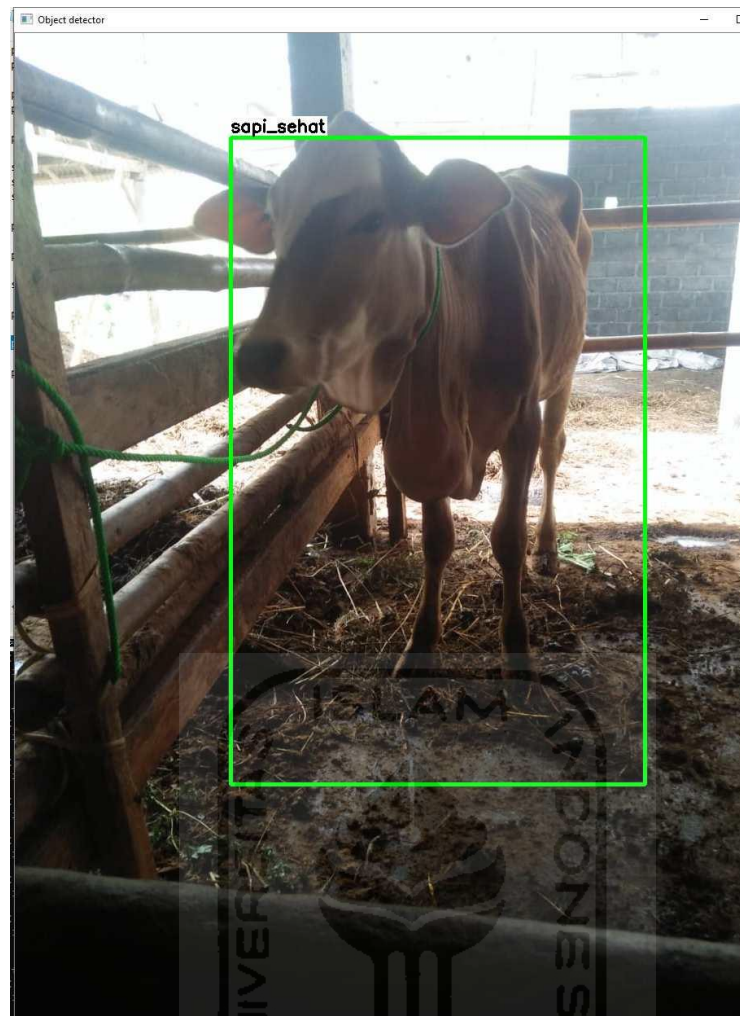


Gambar 5.12 Sapi yang tidak terkena serangan parasit cacing dan terdeteksi benar



Gambar 5.13 Sapi yang tidak terkena serangan parasit cacing namun salah deteksi





Gambar 5.14 Sapi yang terkena serangan parasit cacing namun salah deteksi



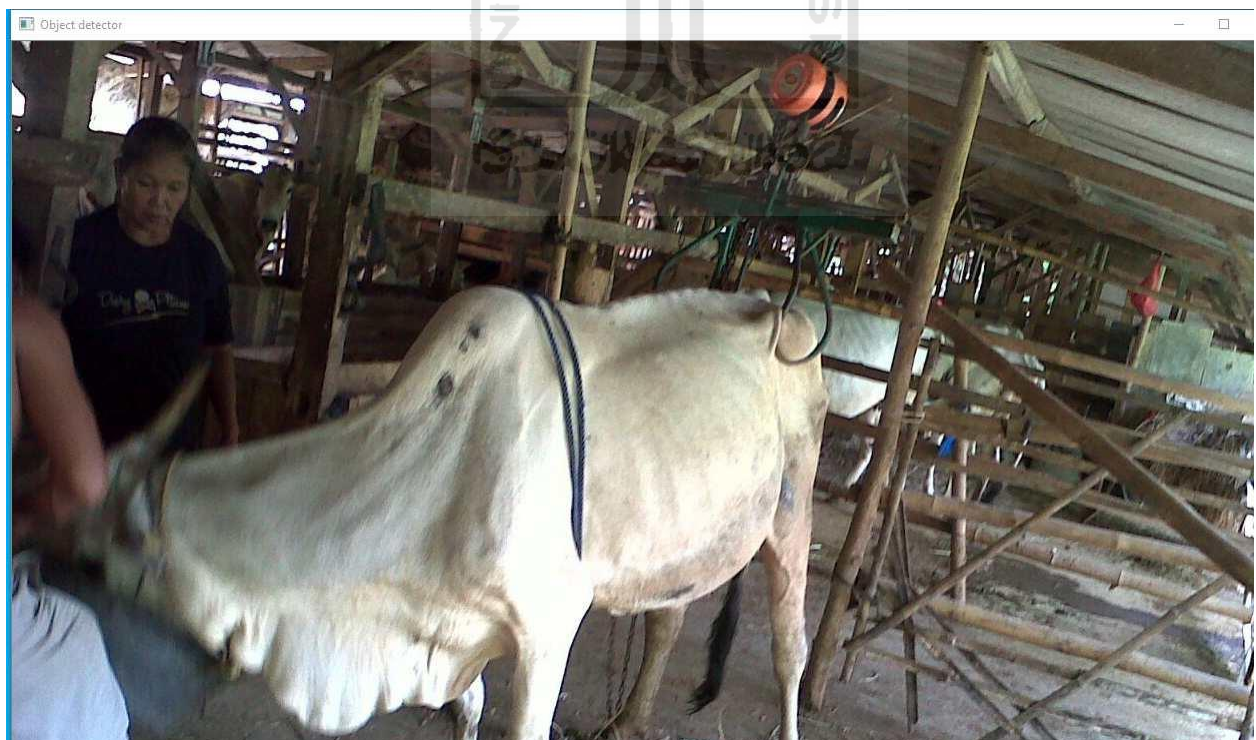
Gambar 5.15 Sapi yang terkena serangan parasit cacing namun salah deteksi



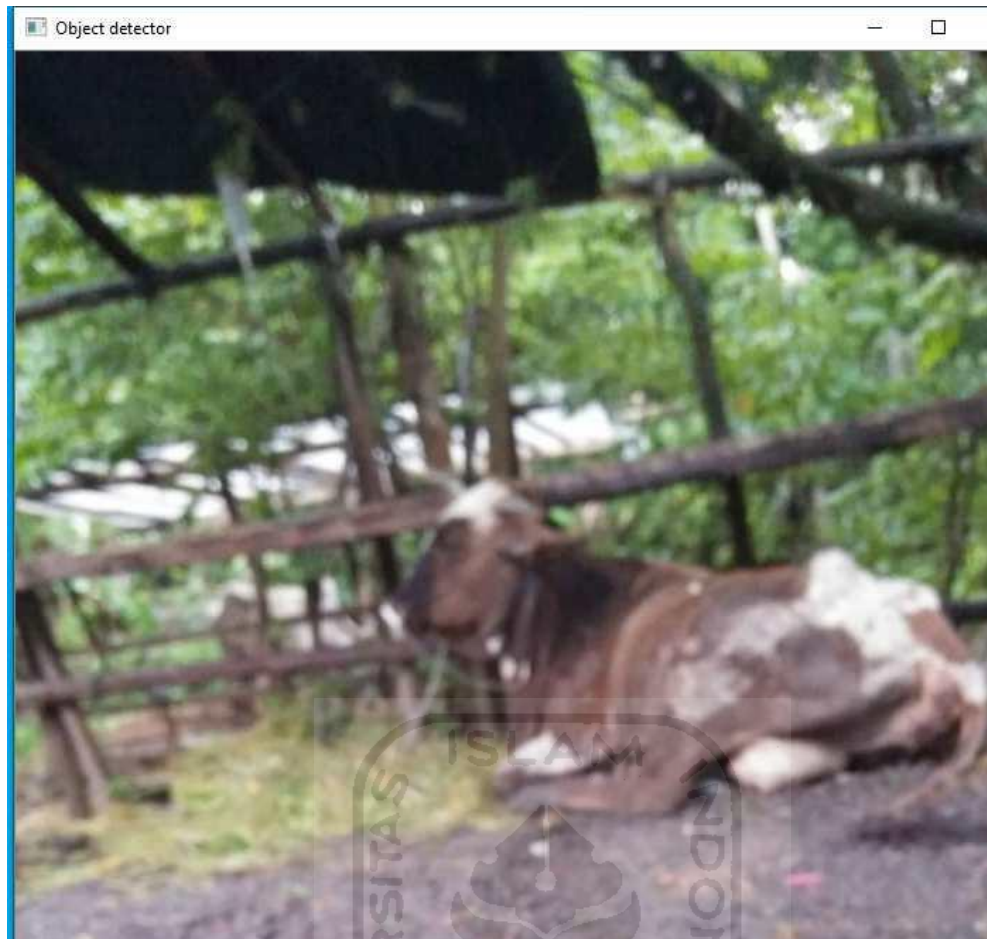
Gambar 5.16 Sapi yang terkena serangan parasit cacing dan terdeteksi benar



Gambar 5.17 Sapi yang terkena serangan parasit cacing namun salah deteksi



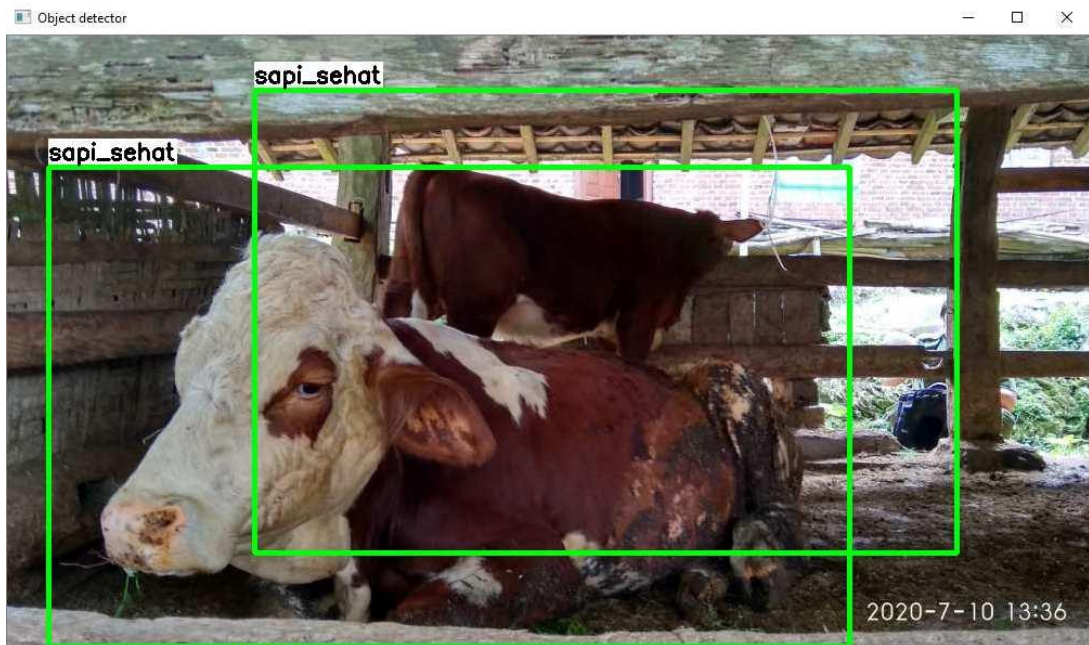
Gambar 5.18 Sapi yang terkena serangan parasit cacing namun tidak terdeteksi



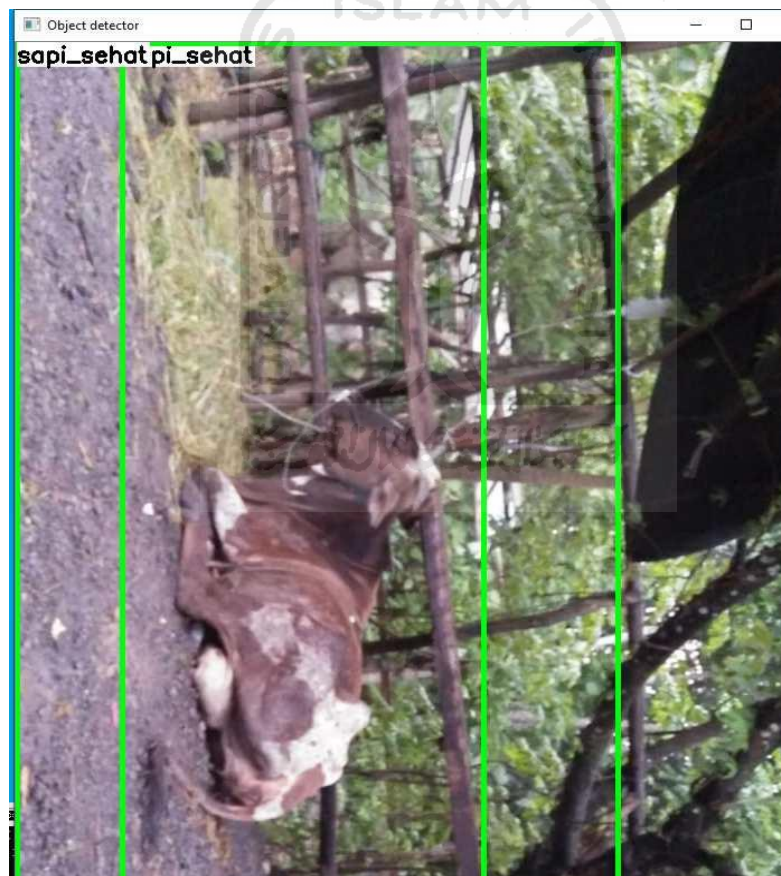
Gambar 5.19 Sapi yang terkena serangan parasit cacing namun tidak terdeteksi



Gambar 5.20 Sapi yang tidak terkena serangan parasit cacing dan terdeteksi



Gambar 5.21 Sapi yang tidak terkena serangan parasit cacing dan terdeteksi benar



Gambar 5.22 Sapi yang terkena serangan parasit cacing namun salah deteksi

2. Training Script

```
# Copyright 2017 The TensorFlow Authors. All Rights Reserved.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
# =====

r"""Training executable for detection models.

This executable is used to train DetectionModels. There are two ways of
configuring the training job:

1) A single pipeline_pb2.TrainEvalPipelineConfig configuration file
can be specified by --pipeline_config_path.

Example usage:
    ./train \
      --logtostderr \
      --train_dir=path/to/train_dir \
      --pipeline_config_path=pipeline_config.pbtxt

2) Three configuration files can be provided: a model_pb2.DetectionModel
configuration file to define what type of DetectionModel is being trained, an
input_reader_pb2.InputReader file to specify what training data will be used and
a train_pb2.TrainConfig file to configure training parameters.

Example usage:
    ./train \
      --logtostderr \
      --train_dir=path/to/train_dir \
      --model_config_path=model_config.pbtxt \
      --train_config_path=train_config.pbtxt \
      --input_config_path=train_input_config.pbtxt
"""

import functools
import json
import os
import tensorflow as tf
```

```

from object_detection.builders import dataset_builder
from object_detection.builders import graph_rewriter_builder
from object_detection.builders import model_builder
from object_detection.legacy import trainer
from object_detection.utils import config_util

tf.logging.set_verbosity(tf.logging.INFO)

flags = tf.app.flags
flags.DEFINE_string('master', '', 'Name of the TensorFlow master to use.')
flags.DEFINE_integer('task', 0, 'task id')
flags.DEFINE_integer('num_clones', 1, 'Number of clones to deploy per worker.')
flags.DEFINE_boolean('clone_on_cpu', False,
                    'Force clones to be deployed on CPU. Note that even if '
                    'set to False (allowing ops to run on gpu), some ops may '
                    'still be run on the CPU if they have no GPU kernel.')
flags.DEFINE_integer('worker_replicas', 1, 'Number of worker+trainer '
                    'replicas.')
flags.DEFINE_integer('ps_tasks', 0,
                    'Number of parameter server tasks. If None, does not use '
                    'a parameter server.')
flags.DEFINE_string('train_dir', '',
                    'Directory to save the checkpoints and training summaries.')

flags.DEFINE_string('pipeline_config_path', '',
                    'Path to a pipeline_pb2.TrainEvalPipelineConfig config '
                    'file. If provided, other configs are ignored')

flags.DEFINE_string('train_config_path', '',
                    'Path to a train_pb2.TrainConfig config file.')
flags.DEFINE_string('input_config_path', '',
                    'Path to an input_reader_pb2.InputReader config file.')
flags.DEFINE_string('model_config_path', '',
                    'Path to a model_pb2.DetectionModel config file.')

FLAGS = flags.FLAGS

@tf.contrib.framework.deprecated(None, 'Use object_detection/model_main.py.')
def main(_):
    assert FLAGS.train_dir, '`train_dir` is missing.'
    if FLAGS.task == 0: tf.gfile.MakeDirs(FLAGS.train_dir)
    if FLAGS.pipeline_config_path:
        configs = config_util.get_configs_from_pipeline_file(
            FLAGS.pipeline_config_path)
    if FLAGS.task == 0:
        tf.gfile.Copy(FLAGS.pipeline_config_path,
                      os.path.join(FLAGS.train_dir, 'pipeline.config'),
                      overwrite=True)
    else:

```

```

configs = config_util.get_configs_from_multiple_files(
    model_config_path=FLAGS.model_config_path,
    train_config_path=FLAGS.train_config_path,
    train_input_config_path=FLAGS.input_config_path)
if FLAGS.task == 0:
    for name, config in [('model.config', FLAGS.model_config_path),
                        ('train.config', FLAGS.train_config_path),
                        ('input.config', FLAGS.input_config_path)]:
        tf.gfile.Copy(config, os.path.join(FLAGS.train_dir, name),
                      overwrite=True)

model_config = configs['model']
train_config = configs['train_config']
input_config = configs['train_input_config']

model_fn = functools.partial(
    model_builder.build,
    model_config=model_config,
    is_training=True)

def get_next(config):
    return dataset_builder.make_initializable_iterator(
        dataset_builder.build(config)).get_next()

create_input_dict_fn = functools.partial(get_next, input_config)

env = json.loads(os.environ.get('TF_CONFIG', '{}'))
cluster_data = env.get('cluster', None)
cluster = tf.train.ClusterSpec(cluster_data) if cluster_data else None
task_data = env.get('task', None) or {'type': 'master', 'index': 0}
task_info = type('TaskSpec', (object,), task_data)

# Parameters for a single worker.
ps_tasks = 0
worker_replicas = 1
worker_job_name = 'lonely_worker'
task = 0
is_chief = True
master = ''

if cluster_data and 'worker' in cluster_data:
    # Number of total worker replicas include "worker"s and the "master".
    worker_replicas = len(cluster_data['worker']) + 1
if cluster_data and 'ps' in cluster_data:
    ps_tasks = len(cluster_data['ps'])

if worker_replicas > 1 and ps_tasks < 1:
    raise ValueError('At least 1 ps task is needed for distributed training.')

if worker_replicas >= 1 and ps_tasks > 0:

```

```

# Set up distributed training.
server = tf.train.Server(tf.train.ClusterSpec(cluster), protocol='grpc',
                        job_name=task_info.type,
                        task_index=task_info.index)

if task_info.type == 'ps':
    server.join()
    return

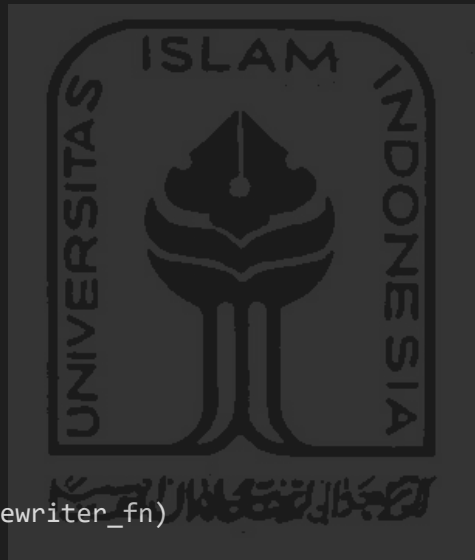
worker_job_name = '%s/task:%d' % (task_info.type, task_info.index)
task = task_info.index
is_chief = (task_info.type == 'master')
master = server.target

graph_rewriter_fn = None
if 'graph_rewriter_config' in configs:
    graph_rewriter_fn = graph_rewriter_builder.build(
        configs['graph_rewriter_config'], is_training=True)

trainer.train(
    create_input_dict_fn,
    model_fn,
    train_config,
    master,
    task,
    FLAGS.num_clones,
    worker_replicas,
    FLAGS.clone_on_cpu,
    ps_tasks,
    worker_job_name,
    is_chief,
    FLAGS.train_dir,
    graph_hook_fn=graph_rewriter_fn)

if __name__ == '__main__':
    tf.app.run()

```



3. XML to CSV Script

```

"""
Usage:
# Create train data:
python xml_to_csv.py -i [PATH_TO_IMAGES_FOLDER]/train -
o [PATH_TO_ANNOTATIONS_FOLDER]/train_labels.csv

# Create test data:
python xml_to_csv.py -i [PATH_TO_IMAGES_FOLDER]/test -
o [PATH_TO_ANNOTATIONS_FOLDER]/test_labels.csv

```

```

"""

import os
import glob
import pandas as pd
import argparse
import xml.etree.ElementTree as ET

def xml_to_csv(path):
    """Iterates through all .xml files (generated by labelImg) in a given directory and combines them in a single Pandas dataframe.

    Parameters:
    -----
    path : {str}
        The path containing the .xml files
    Returns
    -----
    Pandas DataFrame
        The produced dataframe
    """

    xml_list = []
    for xml_file in glob.glob(path + '/*.xml'):
        tree = ET.parse(xml_file)
        root = tree.getroot()
        for member in root.findall('object'):
            value = (root.find('filename').text,
                     int(root.find('size')[0].text),
                     int(root.find('size')[1].text),
                     member[0].text,
                     int(member[4][0].text),
                     int(member[4][1].text),
                     int(member[4][2].text),
                     int(member[4][3].text)
                    )
            xml_list.append(value)
    column_name = ['filename', 'width', 'height',
                   'class', 'xmin', 'ymin', 'xmax', 'ymax']
    xml_df = pd.DataFrame(xml_list, columns=column_name)
    return xml_df

def main():
    # Initiate argument parser
    parser = argparse.ArgumentParser(
        description="Sample TensorFlow XML-to-CSV converter")
    parser.add_argument("-i",
                        "--inputDir",

```

```

        help="Path to the folder where the input .xml files are s
tored",
        type=str)
    parser.add_argument("-o",
                        "--outputFile",
                        help="Name of output .csv file (including path)", type=st
r)
    args = parser.parse_args()

    if(args.inputDir is None):
        args.inputDir = os.getcwd()
    if(args.outputFile is None):
        args.outputFile = args.inputDir + "/labels.csv"

    assert(os.path.isdir(args.inputDir))

    xml_df = xml_to_csv(args.inputDir)
    xml_df.to_csv(
        args.outputFile, index=None)
    print('Successfully converted xml to csv.')

if __name__ == '__main__':
    main()

```

4. Generate TFRecord Script

```

"""
Usage:

# Create train data:
python generate_tfrecord.py --label=<LABEL> --
csv_input=<PATH_TO_ANNOTATIONS_FOLDER>/train_labels.csv --
output_path=<PATH_TO_ANNOTATIONS_FOLDER>/train.record

# Create test data:
python generate_tfrecord.py --label=<LABEL> --
csv_input=<PATH_TO_ANNOTATIONS_FOLDER>/test_labels.csv --
output_path=<PATH_TO_ANNOTATIONS_FOLDER>/test.record
"""

from __future__ import division
from __future__ import print_function
from __future__ import absolute_import

import os

```

```

import io
import pandas as pd
import tensorflow as tf
import sys
sys.path.append("../models/research")

from PIL import Image
from object_detection.utils import dataset_util
from collections import namedtuple, OrderedDict

flags = tf.app.flags
flags.DEFINE_string('csv_input', '', 'Path to the CSV input')
flags.DEFINE_string('output_path', '', 'Path to output TFRecord')
flags.DEFINE_string('label', '', 'Name of class label')
# if your image has more labels input them as
# flags.DEFINE_string('label0', '', 'Name of class[0] label')
# flags.DEFINE_string('label1', '', 'Name of class[1] label')
# and so on.
flags.DEFINE_string('img_path', '', 'Path to images')
FLAGS = flags.FLAGS

# TO-DO replace this with label map
# for multiple labels add more else if statements
def class_text_to_int(row_label):
    if row_label == 'sapi_sehat':
        return 1
    elif row_label == 'sapi_cacingan':
        return 2
    else:
        None

def split(df, group):
    data = namedtuple('data', ['filename', 'object'])
    gb = df.groupby(group)
    return [data(filename, gb.get_group(x)) for filename, x in zip(gb.groups.keys(), gb.groups)]

def create_tf_example(group, path):
    with tf.gfile.GFile(os.path.join(path, '{}'.format(group.filename)), 'rb') as fid:
        encoded_jpg = fid.read()
        encoded_jpg_io = io.BytesIO(encoded_jpg)
        image = Image.open(encoded_jpg_io)
        width, height = image.size

        filename = group.filename.encode('utf8')
        image_format = b'jpg'
        # check if the image format is matching with your images.

```

```

xmins = []
xmaxs = []
ymins = []
ymaxs = []
classes_text = []
classes = []

for index, row in group.object.iterrows():
    xmin = row['xmin'] / width
    xmax = row['xmax'] / width
    ymin = row['ymin'] / height
    ymax = row['ymax'] / height
    class_text = row['class'].encode('utf8')
    class_int = class_text_to_int(class_text)

    tf_example = tf.train.Example(features=tf.train.Features(feature={
        'image/height': dataset_util.int64_feature(height),
        'image/width': dataset_util.int64_feature(width),
        'image/filename': dataset_util.bytes_feature(filename),
        'image/source_id': dataset_util.bytes_feature(filename),
        'image/encoded': dataset_util.bytes_feature(encoded_jpg),
        'image/format': dataset_util.bytes_feature(image_format),
        'image/object/bbox/xmin': dataset_util.float_list_feature(xmins),
        'image/object/bbox/xmax': dataset_util.float_list_feature(xmaxs),
        'image/object/bbox/ymin': dataset_util.float_list_feature(ymins),
        'image/object/bbox/ymax': dataset_util.float_list_feature(ymaxs),
        'image/object/class/text': dataset_util.bytes_list_feature(classes_text),
        'image/object/class/label': dataset_util.int64_list_feature(classes),
    }))

return tf_example

def main(_):
    writer = tf.python_io.TFRecordWriter(FLAGS.output_path)
    path = os.path.join(os.getcwd(), FLAGS.img_path)
    examples = pd.read_csv(FLAGS.csv_input)
    grouped = split(examples, 'filename')
    for group in grouped:
        tf_example = create_tf_example(group, path)
        writer.write(tf_example.SerializeToString())

    writer.close()
    output_path = os.path.join(os.getcwd(), FLAGS.output_path)
    print('Successfully created the TFRecords: {}'.format(output_path))

if __name__ == '__main__':
    tf.app.run()

```

5. Testing Script

```
##### Webcam Object Detection Using Tensorflow-trained Classifier #####
#
# Author: Evan Juras
# Date: 10/27/19
# Description:
# This program uses a TensorFlow Lite model to perform object detection on a live
# webcam
# feed. It draws boxes and scores around the objects of interest in each frame fr
# om the
# webcam. To improve FPS, the webcam object runs in a separate thread from the ma
# in program.
# This script will work with either a Picamera or regular USB webcam.
#
# This code is based off the TensorFlow Lite image classification example at:
# https://github.com/tensorflow/tensorflow/blob/master/tensorflow/lite/examples/p
# ython/label_image.py
#
# I added my own method of drawing boxes and labels using OpenCV.

# Import packages
import os
import argparse
import cv2
import numpy as np
import sys
import time
from threading import Thread
import importlib.util

# Define VideoStream class to handle streaming of video from webcam in separate p
# rocessing thread
# Source - Adrian Rosebrock, PyImageSearch: https://www.pyimagesearch.com/2015/12
# /28/increasing-raspberry-pi-fps-with-python-and-opencv/
class VideoStream:
    """Camera object that controls video streaming from the Picamera"""
    def __init__(self,resolution=(480, 270),framerate=30):
        # Initialize the PiCamera and the camera image stream
        self.stream = cv2.VideoCapture(0)
        ret = self.stream.set(cv2.CAP_PROP_FOURCC, cv2.VideoWriter_fourcc(*'MJPG'
    ))

        ret = self.stream.set(3,resolution[0])
        ret = self.stream.set(4,resolution[1])

        # Read first frame from the stream
        (self.grabbed, self.frame) = self.stream.read()

        # Variable to control when the camera is stopped
        self.stopped = False
```

```

def start(self):
    # Start the thread that reads frames from the video stream
    Thread(target=self.update, args=()).start()
    return self

def update(self):
    # Keep looping indefinitely until the thread is stopped
    while True:
        # If the camera is stopped, stop the thread
        if self.stopped:
            # Close camera resources
            self.stream.release()
            return

        # Otherwise, grab the next frame from the stream
        (self.grabbed, self.frame) = self.stream.read()

def read(self):
    # Return the most recent frame
    return self.frame

def stop(self):
    # Indicate that the camera and thread should be stopped
    self.stopped = True

# Define and parse input arguments
parser = argparse.ArgumentParser()
parser.add_argument('--modeldir', help='Folder the .tflite file is located in',
                    required=True)
parser.add_argument('--graph', help='Name of the .tflite file, if different than detect.tflite',
                    default='detect.tflite')
parser.add_argument('--labels', help='Name of the labelmap file, if different than labelmap.txt',
                    default='labelmap.txt')
parser.add_argument('--threshold', help='Minimum confidence threshold for displaying detected objects',
                    default=0.5)
parser.add_argument('--resolution', help='Desired webcam resolution in WxH. If the webcam does not support the resolution entered, errors may occur.',
                    default='480x270')
parser.add_argument('--edgetpu', help='Use Coral Edge TPU Accelerator to speed up detection',
                    action='store_true')

args = parser.parse_args()

MODEL_NAME = args.modeldir

```

```

GRAPH_NAME = args.graph
LABELMAP_NAME = args.labels
min_conf_threshold = float(args.threshold)
resW, resH = args.resolution.split('x')
imW, imH = int(resW), int(resH)
use_TPU = args.edgetpu

# Import TensorFlow libraries
# If tfLite_runtime is installed, import interpreter from tfLite_runtime, else im
port from regular tensorflow
# If using Coral Edge TPU, import the load_delegate library
pkg = importlib.util.find_spec('tfLite_runtime')
if pkg:
    from tfLite_runtime.interpreter import Interpreter
    if use_TPU:
        from tfLite_runtime.interpreter import load_delegate
else:
    from tensorflow.lite.python.interpreter import Interpreter
    if use_TPU:
        from tensorflow.lite.python.interpreter import load_delegate

# If using Edge TPU, assign filename for Edge TPU model
if use_TPU:
    # If user has specified the name of the .tflite file, use that name, otherwis
e use default 'edgetpu.tflite'
    if (GRAPH_NAME == 'detect.tflite'):
        GRAPH_NAME = 'edgetpu.tflite'

# Get path to current working directory
CWD_PATH = os.getcwd()

# Path to .tflite file, which contains the model that is used for object detectio
n
PATH_TO_CKPT = os.path.join(CWD_PATH, MODEL_NAME, GRAPH_NAME)

# Path to label map file
PATH_TO_LABELS = os.path.join(CWD_PATH, MODEL_NAME, LABELMAP_NAME)

# Load the label map
with open(PATH_TO_LABELS, 'r') as f:
    labels = [line.strip() for line in f.readlines()]

# Have to do a weird fix for label map if using the COCO "starter model" from
# https://www.tensorflow.org/lite/models/object_detection/overview
# First label is '???' , which has to be removed.
if labels[0] == '???':
    del(labels[0])

# Load the Tensorflow Lite model.
# If using Edge TPU, use special load_delegate argument

```

```

if use_TPU:
    interpreter = Interpreter(model_path=PATH_TO_CKPT,
                             experimental_delegates=[load_delegate('libedgetpu.s
o.1.0')]))
    print(PATH_TO_CKPT)
else:
    interpreter = Interpreter(model_path=PATH_TO_CKPT)

interpreter.allocate_tensors()

# Get model details
input_details = interpreter.get_input_details()
output_details = interpreter.get_output_details()
height = input_details[0]['shape'][1]
width = input_details[0]['shape'][2]

floating_model = (input_details[0]['dtype'] == np.float32)

input_mean = 127.5
input_std = 127.5

# Initialize frame rate calculation
frame_rate_calc = 1
freq = cv2.getTickFrequency()

# Initialize video stream
videostream = VideoStream(resolution=(imW,imH),framerate=30).start()
time.sleep(1)

#for frame1 in camera.capture_continuous(rawCapture, format="bgr",use_video_port=
True):
while True:

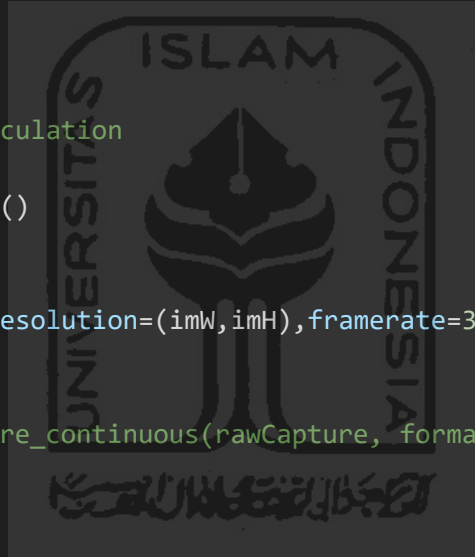
    # Start timer (for calculating frame rate)
    t1 = cv2.getTickCount()

    # Grab frame from video stream
    frame1 = videostream.read()

    # Acquire frame and resize to expected shape [1xHxWx3]
    frame = frame1.copy()
    frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    frame_resized = cv2.resize(frame_rgb, (width, height))
    input_data = np.expand_dims(frame_resized, axis=0)

    # Normalize pixel values if using a floating model (i.e. if model is non-
quantized)
    if floating_model:
        input_data = (np.float32(input_data) - input_mean) / input_std

```



```

# Perform the actual detection by running the model with the image as input
interpreter.set_tensor(input_details[0]['index'],input_data)
interpreter.invoke()

# Retrieve detection results
boxes = interpreter.get_tensor(output_details[0]['index'])[0] # Bounding box
coordinates of detected objects
classes = interpreter.get_tensor(output_details[1]['index'])[0] # Class index
of detected objects
scores = interpreter.get_tensor(output_details[2]['index'])[0] # Confidence o
f detected objects
#num = interpreter.get_tensor(output_details[3]['index'])[0] # Total number
of detected objects (inaccurate and not needed)

# Loop over all detections and draw detection box if confidence is above mini
mum threshold
for i in range(len(scores)):
    if ((scores[i] > min_conf_threshold) and (scores[i] <= 1.0)):

        # Get bounding box coordinates and draw box
        # Interpreter can return coordinates that are outside of image dimens
        ions, need to force them to be within image using max() and min()
        ymin = int(max(1,(boxes[i][0] * imH)))
        xmin = int(max(1,(boxes[i][1] * imW)))
        ymax = int(min(imH,(boxes[i][2] * imH)))
        xmax = int(min(imW,(boxes[i][3] * imW)))

        cv2.rectangle(frame, (xmin,ymin), (xmax,ymax), (10, 255, 0), 2)

        # Draw label
        object_name = labels[int(classes[i])] # Look up object name from "lab
        els" array using class index
        if object_name == "sapi_sehat":
            cv2.rectangle(frame, (xmin,ymin), (xmax,ymax), (10, 255, 0), 4)
        elif object_name == "sapi_cacingan":
            cv2.rectangle(frame, (xmin,ymin), (xmax,ymax), (0, 0, 255), 4)
        label = '%s' % (object_name) # Example: 'person'
        labelSize, baseLine = cv2.getTextSize(label, cv2.FONT_HERSHEY_SIMPLEX
        , 0.7, 2) # Get font size
        label_ymin = max(ymin, labelSize[1] + 10) # Make sure not to draw lab
        el too close to top of window
        cv2.rectangle(frame, (xmin, label_ymin-labelSize[1]-
        10), (xmin+labelSize[0], label_ymin+baseline-
        10), (255, 255, 255), cv2.FILLED) # Draw white box to put label text in
        cv2.putText(frame, label, (xmin, label_ymin-
        7), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 0), 2) # Draw label text

        # Draw framerate in corner of frame
        cv2.putText(frame, 'FPS: {0:.2f}'.format(frame_rate_calc),(30,50),cv2.FONT_HER
        SHEY_SIMPLEX,1,(255,255,0),2,cv2.LINE_AA)

```

```

# All the results have been drawn on the frame, so it's time to display it.
cv2.imshow('Object detector', frame)

# Calculate framerate
t2 = cv2.getTickCount()
time1 = (t2-t1)/freq
frame_rate_calc= 1/time1

# Press 'q' to quit
if cv2.waitKey(1) == ord('q'):
    break

# Clean up
cv2.destroyAllWindows()
videostream.stop()

```

6. Resizer Script



```

## Bulk image resizer

# This script simply resizes all the images in a folder to one-eighth their
# original size. It's useful for shrinking large cell phone pictures down
# to a size that's more manageable for model training.

# Usage: place this script in a folder of images you want to shrink,
# and then run it.

import numpy as np
import cv2
import os

dir_path = os.getcwd()

for filename in os.listdir(dir_path):
    # If the images are not .JPG images, change the line below to match the image
    # type.
    if filename.endswith(".JPG"):
        image = cv2.imread(filename)
        resized = cv2.resize(image, None, fx=0.25, fy=0.25, interpolation=cv2.INTER
_AREA)
        cv2.imwrite(filename, resized)

```