# PEMBUATAN MODEL DETEKSI GEJALA AWAL PENYAKIT MULUT DAN KUKU PADA SAPI BERBASIS CITRA MENGGUNAKAN METODE *CONVOLUTIONAL NEURAL NETWORK*

S K R I P S I

**HELVIANI ZEBUA**
**F1E119051**

**PROGRAM STUDI SISTEM INFORMASI**
**JURUSAN TEKNIK ELEKTRO DAN INFORMATIKA**

**FAKULTAS SAINS DAN TEKNOLOGI**
**UNIVERSITAS JAMBI**
**2023**

# SURAT PERNYATAAN

Dengan ini saya menyatakan bahwa skripsi ini benar-benar karya sendiri. Sepanjang pengetahuan saya tidak terdapat karya atau pendapat yang ditulis atau diterbitkan orang lain kecuali sebagai acuan atau kutipan dengan mengikuti tata penulisan karya ilmiah yang telah lazim.
Tanda tangan yang tertera dalam halaman pengesahan adalah asli. Jika tidak asli, saya siap menerima sanksi sesuai dengan peraturan yang berlaku.

Jambi, 16 Januari 2023
Yang menyatakan

HELVIANI ZEBUA
F1E119051

# PEMBUATAN MODEL DETEKSI GEJALA AWAL PENYAKIT MULUT DAN KUKU PADA SAPI BERBASIS CITRA MENGGUNAKAN METODE *CONVOLUTIONAL NEURAL NETWORK*

S K R I P S I

Diajukan sebagai salah satu syarat untuk memperoleh
Gelar Sarjana pada Program Studi Sistem Informasi

**HELVIANI ZEBUA**
**F1E119051**

**PROGRAM STUDI SISTEM INFORMASI**
**JURUSAN TEKNIK ELEKTRO DAN INFORMATIKA**

**FAKULTAS SAINS DAN TEKNOLOGI**
**UNIVERSITAS JAMBI**
**2023**

**PENGESAHAN**

**PENGESAHAN**

Skripsi dengan judul **PEMBUATAN MODEL DETEKSI GEJALA AWAL PENYAKIT MULUT DAN KUKU PADA SAPI BERBASIS CITRA MENGGUNAKAN METODE *CONVOLUTIONAL NEURAL NETWORK*** yang disusun oleh **HELVIANI ZEBUA, NIM: F1E119051** telah dipertahankan di depan penguji pada tanggal 06 Januari 2023 dan dinyatakan Lulus.

Susunan Tim Penguji:

| | | |
|---|---|---|
| Ketua | : | Edi Saputra, S.T., M.Sc. |
| Sekretaris | : | Pradita Eko Prasetyo Utomo, S.Pd., M.Cs. |
| Anggota | : | Ir. Indra Weni, M.Kom. |
| | | Ulfa Khaira, S.Komp., M.Kom. |
| | | Rizqa Raaiqa Bintana, S.T., M.Kom. |

Pembimbing Utama,

Edi Saputra, S.T., M.Sc.
NIP. 198501082015041003

Disetujui:
Pembimbing Pendamping,

Pradita Eko Prasetyo Utomo, S.Pd., M.Cs., CIISA.
NIP. 198710282019031010

Dekan
Fakultas Sains dan Teknologi,

Drs. Jefri Marzal, M.Sc., D.I.T.
NIP. 196806021993031004

Diketahui:
Ketua Jurusan
Teknik Elektro dan Informatika,

Nehru, S.Si., M.T.
NIP. 197602082001121002



i

**RINGKASAN**

Penyakit Mulut dan Kuku (PMK) merupakan penyakit yang sangat menular dan disebabkan oleh virus penyakit mulut dan kuku, yaitu *Foot and Mouth Disease Virus* (FMDV) pada hewan berkuku belah. Sejak tahun 1887, Indonesia beberapa kali mengalami wabah PMK. Tersebarnya PMK ke populasi hewan rentan di Indonesia dengan cepat disebabkan oleh benda yang terkontaminasi virus PMK serta lalu lintas hewan dan produknya. Untuk meminimalisir penyebaran PMK, diperlukan pendeteksian PMK sedini mungkin.

Dalam kasus pendeteksian objek, salah satu metode yang dapat digunakan adalah *Convolutional Neural Network* (CNN). CNN dapat bekerja lebih baik dalam pembuatan model dengan dataset citra dibandingkan dengan jenis algoritma neural network lainnya karena jaringan pada metode CNN mengkhususkan neuron untuk mengenali bentuk citra (pada tahap konvolusi). Penelitian ini dilakukan untuk menemukan arsitektur model deteksi gejala awal PMK pada sapi dengan metode CNN dengan akurasi model terbaik dalam pendeteksian gejala awal PMK pada sapi. Hasil penelitian ini menunjukkan bahwa penambahan jumlah *epoch* dan nilai *batch size* dapat mempengaruhi akurasi dari model CNN. Pada penelitian ini, jumlah *epoch* 20 dan nilai *batch size* 32 menghasilkan akurasi terbaik dengan nilai akurasi *training* sebesar 99% dan akurasi pengujian pada 22 citra adalah sebesar 91%.

**RIWAYAT HIDUP**

Helviani Zebua, dilahirkan di Jambi pada tanggal 17 November 2000. Anak kedua dari dua bersaudara pasangan dari Anotona Zebua dan Meriati Gulo. Penulis menyelesaikan pendidikan Sekolah Dasar di SD YPMM Tebing Tinggi pada tahun 2013. Pada tahun itu juga penulis melanjutkan pendidikan di SMP YPMM Tebing Tinggi selama satu tahun ajaran dan kembali melanjutkan pendidikan di SMP Xaverius 1 Kota Jambi dan tamat pada tahun 2016. Penulis kemudian melanjutkan pendidikan Sekolah Menengah Atas di SMA Negeri 3 Kota Jambi pada tahun 2016 dan menyelesaikan pendidikan SMA pada tahun 2019. Pada tahun 2019, penulis melanjutkan pendidian jenjang Strata 1 dan terdaftar sebagai mahasiswa di Program Studi Sistem Informasi, Jurusan Teknik Elektro dan Informatika, Fakultas Sains dan Teknologi, Universitas Jambi melalui jalur SBMPTN (Seleksi Bersma Masuk Perguruan Tinggi Negeri). Selama menempuh Pendidikan Strata 1, penulis aktif dalam bidang akademik maupun non akademik. Pada tahun 2020 penulis berhasil mendapatkan medali perunggu pada ajang Olimpiade Sains tingkat mahasiswa se-Indonesia di bidang ilmu komputer. Tahun 2021 penulis meraih juara 2 pada kompetisi debat Bahasa Inggris di tingkat fakultas dan di tahun yang sama penulis mewakili Himpunan Mahasiswa Universitas Jambi (HIMASI UNJA) menerima dana hibah dari Kemendikbudristek dalam Program Holistik Pembinaan dan Pemberdayaan Desa dan dalam tim tersebut penulis dipercaya sebagai sekretaris. Di tahun 2020-2021 penulis dipercaya dan diberi tanggungjawab sebagai koordinator Divisi Riset dan Teknologi HIMASI UNJA. Pada tahun 2022 penulis mengikuti program magang selama 6 bulan di Kementerian Agraria dan Tata Ruang/Badan Pertanahan Nasional tepatnya di Direktorat Jenderal Tata Ruang di Jakarta Selatan dan ditempatkan di divisi Data dan Informasi serta Studio Peta.

## PRAKATA

Puji dan syukur ke hadirat Tuhan Yang Maha Esa atas berkat dan rahmat serta karunia-Nya, sekaligus peran kedua orang tua dan saudara penulis yang senantiasa selalu mendoakan, memberikan motivasi dan pengorbanannya baik dari segi moril dan materiil sehingga skripsi yang berjudul "Pembuatan Model Deteksi Gejala Awal Penyakit Mulut dan Kuku pada Sapi Berbasis Citra Menggunakan Metode *Convolutional Neural Network*" ini dapat diselesaikan dengan baik.

Dalam penulisan skripsi ini, tidak terlepas dari kerja sama dan bantuan dari berbagai pihak, sehingga pada kesempatan ini diucapkan terima kasih yang sebesar-besarnya bagi semua pihak yang telah memberikan bantuan moril maupun materiil baik langsung maupun tidak langsung dalam penyusunan skripsi ini hingga selesai dan berkontribusi secara maksimal, terutama kepada:

1. Dekan Fakultas Sains dan Teknologi.
2. Ketua Jurusan Teknik Elektro dan Informatika Fakultas Sains dan Teknologi Universitas Jambi.
3. Ketua/Koordinator Program Studi Sistem Informasi.
4. Edi Saputra, S.T., M.Sc. dan Pradita Eko Prasetyo Utomo, S.Pd., M.Cs., CIISA. selaku Dosen Pembimbing yang telah memberikan arahan serta bimbingan secara langsung kepada penulis dalam proses penyusunan skripsi.
5. Ir. Indra Weni, M.Kom, Ulfa Khaira, S.Komp., M.Kom., dan Rizqa Raaiqa Bintana, S.T., M.Kom. selaku Tim Penguji Skripsi yang telah memberikan berbagai masukan dan saran untuk kesempurnaan skripsi ini.
6. Pradita Eko Prasetyo Utomo, S.Pd., M.Cs., CIISA. selaku Dosen Pembimbing Akademik yang selalu memberikan motivasi dan pengarahan selama masa studi.
7. Seluruh Dosen beserta Staf di Program Studi Sistem Informasi Universitas Jambi atas segala ilmu dan bimbingan selama masa studi.
8. Imma, Sisilia, Evelyn, Aurelia, Gira, Margaretha, Renatha, Fhirda, Indica yang selalu mendukung dan memberikan motivasi yang luar biasa kepada penulis selama masa studi.
9. Nurandini, Farah, Vira, Crysant, Iqbal, Raldi, dan Taufiq sebagai teman seperjuangan semasa kuliah serta telah membantu dan mendukung penulis selama masa studi hingga penyelesaian skripsi.
10. Keluarga besar Himpunan Mahasiswa Sistem Informasi Universitas Jambi.
11. Teman-teman anggota PHP2D Nasional HIMASI UNJA 2021.

12. Teman-teman Program Studi Sistem Informasi angkatan 2019.
13. Teman-teman di adorablelildove yang selalu memberikan semangat selama masa studi.

Akhir kata penulis mengucapkan terima kasih kepada semua pihak yang telah membantu dan memberikan dukungan. Penulis berharap skripsi ini dapat memberikan manfaat bagi banyak orang khususnya di bidang Sistem Informasi. Penulis menyadari bahwa skripsi ini masih jauh dari kesempurnaan. Oleh sebab itu, penulis mengharapkan kritik dan saran yang bersifat membangun dari semua pihak demi kesempurnaan skripsi ini.

Jambi, 16 Januari 2023

Helviani Zebua

F1E119051

**DAFTAR ISI**

Halaman

**DAFTAR TABEL**

Tabel Halaman

**DAFTAR GAMBAR**

Gambar Halaman

**DAFTAR LAMPIRAN**

Lampiran Halaman

# I. PENDAHULUAN

## 1.1 Latar Belakang

Penyakit Mulut dan Kuku (PMK) merupakan penyakit yang sangat menular pada hewan berkuku belah, seperti sapi, kerbau, domba, kambing, dan kijang. Penyakit ini disebabkan oleh beberapa varian genetik dari tujuh serotipe yang berbeda secara imunologis, sehingga untuk pemulihan dari infeksi yang berasal dari satu serotipe tidak memberikan dampak perlindungan apapun terhadap varian genetik pada enam serotipe lainnya (Kitching, 2002). Hewan yang terinfeksi virus ini akan mengalami demam dan beberapa tanda gejala lainnya, yaitu di dalam dan sekitar mulut, lidah, bibir, dan celah kuku pada hewan yang bersangkutan akan mengalami luka lepuh yang berisi cairan dan pada hewan betina gejala yang sama akan terjadi pada ambing dan puting susu (Pusat Data dan Analisa Tempo, 2020). Selain itu, juga akan muncul hipersalivasi, saliva terlihat menggantung, air liur berbusa di lantai kandang (Direktorat Kesehatan Hewan, 2022).

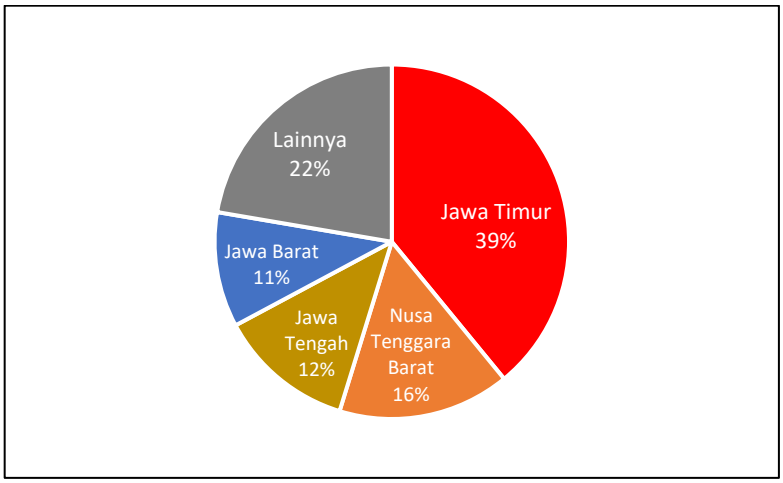
Sejak tahun 1887, Indonesia mengalami beberapa kali wabah PMK (Direktorat Kesehatan Hewan, 2022). Direktorat Kesehatan Hewan juga menjelaskan bahwa di tahun 1887, penyakit ini mulai masuk ke daerah Malang, Jawa Timur melalui impor sapi dari Belanda dan kasus terakhir wabah PMK terjadi di pulau Jawa pada tahun 1983 yang kemudian dapat diberantas melalui program vaksinasi massal. Namun, hingga saat ini beberapa negara di kawasan Asia Tenggara masih mengalami wabah PMK, sehingga hal ini dapat memungkinkan masuknya PMK ke Indonesia.

Wabah PMK kembali menyerang ribuan sapi di Indonesia dan ditemukan pertama kali di Kabupaten Gresik pada 28 April 2022 sebanyak 402 ekor sapi potong (Sutawi, 2022). Sutawi juga mengatakan bahwa kasus kedua ditemukan 102 ekor sapi di Kabupaten Lamongan dan 595 ekor sapi perah dan kerbau di Kabupaten Sidoarjo terinfeksi PMK pada 1 Mei 2022. Dilansir dari laman Siaga PMK, hingga 11 Juli 2022 tercatat sebanyak 219.821 sisa kasus PMK di Indonesia, dengan data tambahan sebanyak 361.207 ekor sakit, 135.504 sembuh, dan 2.342 ekor mati (Siaga PMK, 2022).

Tersebarnya PMK ke populasi hewan rentan di Indonesia dengan cepat disebabkan oleh kendaraan dan benda yang terkontaminasi virus PMK serta lalu lintas hewan dan produknya. Untuk meminimalisir dampak dan penyebaran PMK, maka diperlukan pendeteksian PMK sedini mungkin dan pengendalian lalu lintas hewan rentan serta produknya ke daerah lain yang masih bebas. Strategi utama yang perlu diterapkan pada saat wabah PMK terjadi di Indonesia adalah

melaksanakan *stamping out* dengan sistem *zoning* atau perwilayahan sehingga daerah lain yang tidak terlular tetap dapat bebas dan kegiatan perdagangan di daerah tersebut dapat terus berjalan (Direktorat Kesehatan Hewan, 2022). Meskipun PMK memiliki angka mordibitas tinggi dan mortalitas rendah, penyakit ini tetap berpotensi menurunkan produksi ternak dan hal ini akan berdampak pada aspek ekonomi peternak terlebih ekonomi negara.

Sebaran kasus PMK pada provinsi yang paling terkena dampak di Indonesia divisualisasikan pada Gambar 1.




**Gambar 1.** Sebaran Kasus PMK di Indonesia (Siaga PMK, 2022)

Penanggulangan dan pengendalian virus PMK dapat dilakukan dengan beberapa tindakan, yaitu tindakan karantina dan pengawasan lalu lintas, pemusnahan atau *stamping out*, penelusuran, surveilans, pengobatan hewan tertular, dan kontrol hewan liar (Direktorat Kesehatan Hewan, 2022). Selain itu Direktorat Kesehatan Hewan juga mengatakan bahwa solusi pasti untuk menghentikan produksi virus PMK oleh hewan tertular belum ada, namun beberapa tindakan dapat diterapkan untuk menghilangkan virus PMK pada lingkungan yang terkontaminasi, yaitu dekontaminasi kandang, peralatan, kendaraan, dan bahan-bahan permanen lainnya yang berpotensi menularkan PMK serta disposal bahan-bahan dan peralatan tidak permanen yang terkontaminasi atau dengan memusnahkan hewan tertular (*stamping out*).

Untuk mewujudkan tindakan cepat dalam mendeteksi gejala awal PMK, diperlukan pengembangan sistem untuk membantu para peternak dalam pendeteksian gejala luka PMK terhadap sapinya. Seperti yang telah dijelaskan sebelumnya, salah satu gejala yang akan terlihat pada hewan terinfeksi PMK adalah luka lepuh pada beberapa bagian tubuh hewan. Luka lepuh atau vesikel ini akan terlihat setelah masa inkubasi dan terdapat pada sekitar mulut, lidah, gusi, nostril, kulit sekitar teracak, dan ambing. Oleh karena itu, diperlukan sistem yang dapat mendeteksi dan mengenali gejala awal PMK terutama gejala

timbulnya vesikel pada hewan peka agar peternak dapat langsung melakukan pertolongan pertama pada hewan yang terinfeksi agar dapat menghambat penyebaran virus ke daerah lain mengingat belum adanya solusi untuk menyembuhkan hewan yang terjangkit PMK.

Dalam pendeteksian objek atau *object detection* digunakan *dataset* berupa gambar sebagai *input*. Salah satu metode yang dapat digunakan dalam pendeteksian objek ini adalah *Convolutional Neural Network* (CNN). CNN merupakan metode yang banyak digunakan dalam penelitian yang berkaitan dengan *object detection* dan *image classification* karena pada penelitian terdahulu penggunaan metode ini menghasilkan nilai akurasi yang relatif tinggi dan memiliki hasil yang signifikan dalam pengenalan gambar atau citra. CNN merupakan algoritma yang dapat bekerja lebih baik dalam pembuatan model dengan *dataset* citra dibandingkan dengan jenis algoritma *neural network* lainnya karena jaringan pada metode CNN mengkhususkan neuron untuk mengenali bentuk citra (pada tahap konvolusi) (Mueller & Massaron, 2021b).

Penelitian dengan studi kasus pendeteksian penyakit dengan menggunakan metode CNN telah dilakukan oleh Prastika dan Zuliarso, 2021 dengan judul "Deteksi Penyakit Kulit Wajah Menggunakan *Tensorflow* dengan Metode *Convolutional Neural Network*". Penelitian ini menggunakan data *train* sebanyak 700 citra dan dikelompokkan menjadi 20 jenis penyakit kulit wajah. Kesimpulan dari penelitian ini adalah performa dari model pendeteksian penyakit kulit wajah mendapat nilai akurasi tertinggi sebesar 99,91%.

Penelitian dengan studi kasus pembuatan model untuk pendeteksian penyakit juga telah dilakukan oleh Saputra et al., 2021. Pada penelitian ini dilakukan percobaan dengan menerapkan algoritma CNN dan MobileNet pada model yang dibangun. Data yang digunakan pada penelitian ini adalah data citra penyakit daun padi berukuran 224x224 piksel dengan hasil akurasi pada *Confusion Matrix* sebesar 92%.

Selain itu, penelitian yang membangun model untuk mendeteksi gejala PMK pada hewan ternak telah dilakukan oleh Kuhamba, 2020 dengan judul "*A Deep Learning Based Approach for Foot and Mouth Disease Detection*". Kuhamba menggunakan data citra hewan ternak yang didapatkan melalui beberapa sumber dan citra yang didapatkan merupakan citra pada area lidah, gusi, ambing, mulut, dan kaki hewan. Penelitian ini melakukan perbandingan akurasi yang dihasilkan pada beberapa arsitektur *deep learning*, yaitu DenseNet, Resnet, Google Net, VGGNet, dan Alex Net. Didapatkan akurasi terbaik dengan menggunakan DenseNet yaitu sebesar 93,75%.

Beberapa penelitian tersebut menggunakan CNN dalam mendeteksi objek dan mendapatkan hasil akurasi yang sangat baik. Meski metode CNN telah banyak digunakan dalam pendeteksian penyakit, namun belum ada studi penelitian terkait pendeteksian gejala awal PMK yang mengkhususkan hewan sapi sebagai objek. Penggunaan *dataset* hewan sapi diberlakukan untuk meminimalisir kesalahan sistem dalam mendeteksi gejala awal PMK. Penggunaan data gambar hewan ternak yang berbeda-beda akan menimbulkan kesalahan sistem dalam mempelajari data karena tidak semua hewan ternak memiliki bentuk anggota tubuh yang sama. Agar proses pendeteksian dapat dilakukan secara berkelanjutan oleh pengguna yang terkait, maka digunakan *library Flask* untuk membuat *web interface* dan *web service* dengan tujuan agar model dapat digunakan pengembang sistem di berbagai *platform*.

Berdasarkan uraian di atas, dalam penelitian ini akan dilakukan pembuatan model untuk mendeteksi gejala awal PMK pada sapi yang terinfeksi virus PMK. Algoritma yang akan digunakan pada pembuatan model ini adalah algoritma CNN. Oleh karena itu, penulis membuat penelitian yang berjudul **"Pembuatan Model Deteksi Gejala Awal Penyakit Mulut dan Kuku pada Sapi Berbasis Citra Menggunakan Metode *Convolutional Neural Network*"** dengan studi kasus deteksi gejala awal PMK pada hewan sapi yang terinfeksi virus PMK.

Diharapkan adanya penelitian ini mampu membantu pengendalian dan penanggulangan wabah PMK yang terjadi di Indonesia dan memberikan informasi yang berguna bagi peternak yang belum mengetahui hal-hal mengenai gejala penyakit PMK. Harapan lainnya dari penelitian ini yaitu model yang dikembangkan mampu mendeteksi apakah gejala luka yang terdapat pada tubuh hewan peka merupakan gejala awal PMK atau tidak dengan baik sehingga informasi yang dihasilkan dapat berguna bagi pihak yang membutuhkan.

## 1.2 Rumusan Masalah

Dari latar belakang di atas, dirumuskan beberapa permasalahan sebagai berikut.

1. Bagaimana pembuatan model deteksi gejala awal PMK berbasis citra menggunakan metode CNN?
2. Bagaimana nilai akurasi model yang dihasilkan dalam mendeteksi gejala awal PMK?
3. Bagaimana membangun aplikasi deteksi gejala awal PMK yang diintegrasikan dengan model berupa REST API?

## 1.3 Tujuan

Tujuan dari penelitian ini adalah :

1. Untuk mengetahui bagaimana pembuatan model deteksi gejala awal PMK berbasis citra menggunakan metode CNN.
2. Untuk mengetahui nilai akurasi model yang telah dibangun dalam mendeteksi gejala awal PMK.
3. Untuk membangun sebuah aplikasi deteksi gejala awal PMK yang diintegrasikan dengan model berupa REST API sehingga dapat digunakan pada berbagai *platform*.

## 1.4 Batasan Masalah

Batasan masalah dari penelitian ini adalah :

1. *Dataset* yang digunakan merupakan citra berupa luka sapi yang terjangkit PMK dan sehat.
2. *Dataset* didapatkan dari hasil teknik *scraping* gambar, data citra dari Fakultas Peternakan Universitas Jambi, dan data citra dari Dinas Pertanian dan Ketahanan Pangan Kota Jambi.
3. Klasifikasi citra mencakup delapan kelas, yaitu Air Liur, Gusi, Kaki, dan Lidah untuk Gejala PMK dan Air Liur, Gusi, Kaki, dan Lidah untuk yang Sehat.
4. Model dapat memberikan hasil prediksi benar pada objek yang termasuk ke dalam delapan kelas yang telah ditetapkan.

## 1.5 Manfaat Penelitian

Penelitian ini diharapkan dapat memberikan informasi mengenai proses pembuatan model deteksi gejala PMK berbasis citra menggunakan metode CNN, nilai akurasi terbaik yang dihasilkan saat pengujian model PMK, dan model yang dibangun sebagai Rest API, sehingga dapat dikembangkan untuk penelitian pembuatan model deteksi gejala PMK atau penyakit hewan lainnya.

# II. TINJAUAN PUSTAKA

## 2.1 Penyakit Mulut dan Kuku (PMK)

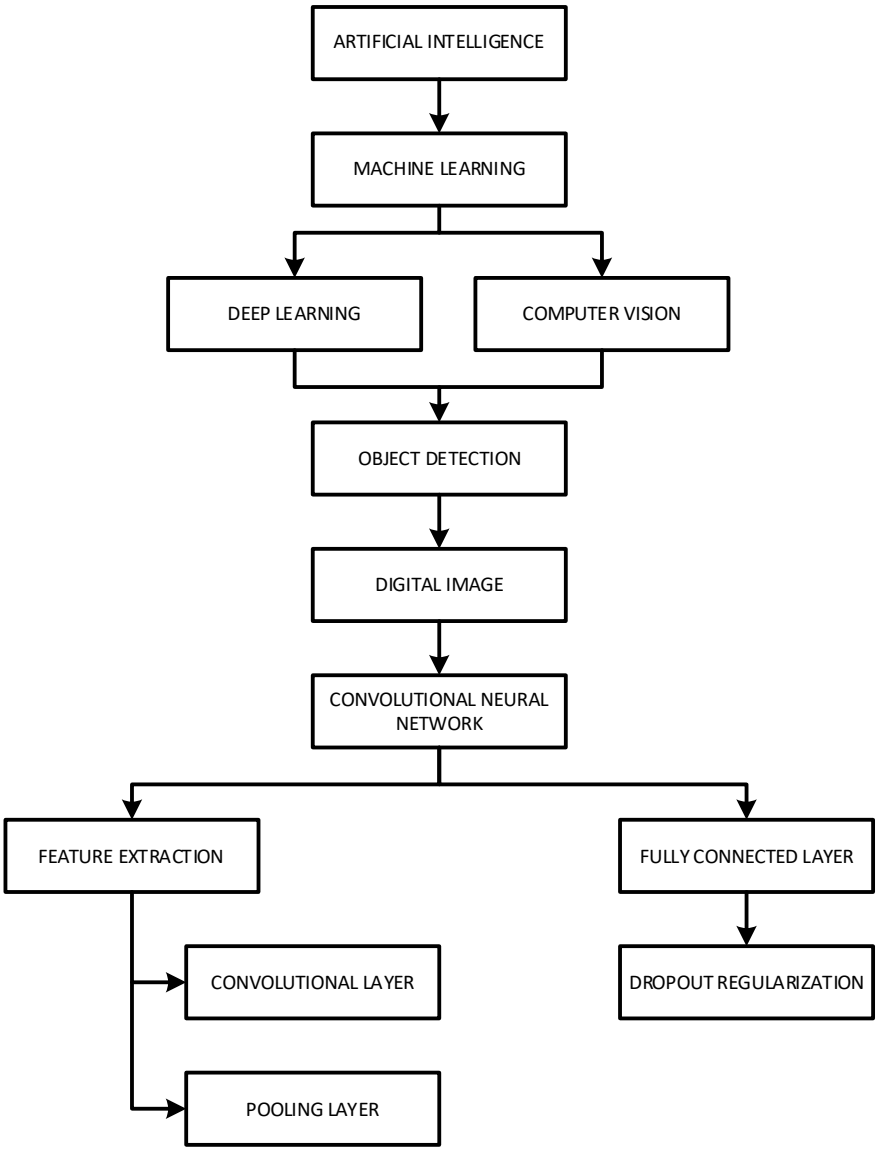
Penyakit Mulut dan Kuku (PMK) merupakan penyakit hewan menular yang disebabkan oleh virus penyakit mulut dan kuku, yaitu *Foot and Mouth Disease Virus* (FMDV). PMK memiliki masa inkubasi selama 2 – 7 hari dan selama masa inkubasi ini, virus bereplikasi dalam nasofaring (Hamjaya Putra Hamdu, 2019). Hamjaya juga mengatakan bahwa masuknya virus ke dalam aliran darah hewan di area otot, kelenjar limfa, sumsum tulang, dan beberapa bagian lainnya atau disebut juga dengan *viraemia* terjadi setelah beberapa jam terinfeksi, tetapi biasanya tidak lebih dari 24 sampai 26 jam pasca terinfeksi.

PMK biasanya sangat menular dan menyerang hewan-hewan peka atau berkuku belah seperti sapi, babi, kambing, domba, kerbau, dan beberapa hewan liar seperti rusa, babi hutan, antelop. Meskipun PMK tidak menyebabkan angka kematian yang tinggi pada hewan dewasa, penyakit ini dapat menimbulkan gejala klinis yang bervariasi tergantung serotipe virus PMK yang menyerang. Gejala klinis yang akan segera muncul sejak pertama kali terinfeksi, yaitu kenaikan suhu tubuh dan diikuti lemas, nafsu makan menurun, terdapat luka lepuh pada tubuh, *salivasi* akan meningkat dan disertai busa di sekitar bibir serta saliva yang menggantung di area mulut (Grubman & Baxt, 2004).

Indonesia sudah beberapa kali mengalami wabah PMK sejak pertama kali masuk pada tahun 1887 melalui impor sapi dari Belanda (Direktorat Kesehatan Hewan, 2022). Direktorat Kesehatan Hewan juga menyebutkan bahwa wabah PMK terakhir terjadi di pulau Jawa di tahun 1983 dan kemudian dapat dimusnahkan melalui program vaksinasi massal, sehingga pada tahun 1986 Indonesia dinyatakan sebagai negara bebas PMK melalui Surat Keputusan Menteri Pertanian Nomor 260 Tahun 1986 dan pada tahun 1990 diakui OIE dengan Resolusi nomor XI. Namun, Wabah PMK kembali menyerang ribuan sapi di Indonesia dan ditemukan pertama kali di Kabupaten Gresik pada 28 April 2022 sebanyak 402 ekor sapi potong. Kasus kedua ditemukan 102 ekor sapi di Kabupaten Lamongan dan 595 ekor sapi perah dan kerbau di Kabupaten Sidoarjo terinfeksi PMK pada 1 Mei 2022 (Sutawi, 2022).

## 2.2 Peta Konsep

Berikut merupakan peta konsep yang berupa sebuah bagan skematis yang berfungsi untuk memberikan gambaran secara umum mengenai hubungan antara beberapa konsep yang berkaitan dengan penelitian.




**Gambar 2.** Peta Konsep

## 2.3 *Artificial Intelligence*

*Artificial Inteligence* (AI) berarti kecerdasan buatan yang banyak dimanfaatkan untuk memudahkan pekerjaan manusia. AI sendiri dengan cerdas dapat meniru perilaku manusia dalam mengerjakan suatu hal. AI dapat diklasifikasikan ke dalam tiga hal, yaitu *analytical*, *human-inspired*, dan *humanized* AI, tergantung pada tipe atau jenis kecerdasan yang ditunjukkan, seperti kecerdasan kognitif, emosional, dan sosial atau ke dalam *Artificial Narrow Intelligence* (kecerdasan buatan yang terbatas), dan *Artificial Super Intelligence* berdasarkan tahap evolusinya (Haenlein & Kaplan, 2019).

AI dapat dikategorikan ke dalam 4 (empat) cara, yaitu (1) *acting humanly*, yaitu ketika sebuah komputer berperilaku seperti manusia. (2) *thinking humanly*, yaitu ketika komputer berfikir seperti manusia dan komputer melakukan tugas

yang membutuhkan kecerdasan dari manusia. (3) *thinking rationally*, yaitu komputer mempelajari bagaimana cara manusia berfikir menggunakan beberapa standar yang menggambarkan perilaku manusia. (4) *acting rationally*, yaitu komputer mempelajari bagaimana manusia berperilaku dalam situasi tertentu dan pada batasan tertentu (Mueller & Massaron, 2021).

## 2.4 *Machine Learning*

AI akan menghasilkan sistem yang mampu meniru kecerdasan manusia untuk melakukan berbagai tugas, sedangkan *machine learning* merupakan proses memberikan pembelajaran pada mesin dengan mengembangkan algoritma untuk melakukan tugas-tugas berdasarkan data yang tersedia, yang artinya *machine learning* merupakan bagian dari AI.

*Machine learning* bergantung pada algoritma yang digunakan untuk menganalisis *dataset* yang besar (Mueller & Massaron, 2021b). *Machine learning* merupakan istilah yang diakui secara universal yang biasanya mengacu pada ilmu dan teknik untuk mengembangkan mesin yang mampu melakukan berbagai hal yang berguna (Burkov, 2019). Burkov juga mengatakan bahwa *machine learning* juga dapat didefinisikan sebagai proses pemecahan masalah praktis dengan mengumpulkan *dataset* dan secara algoritma membangun model statistik berdasarkan *dataset* tersebut dan model statistik yang dibangun diasumsikan dapat digunakan untuk memecahkan masalah praktis.

Implementasi *machine learning* akan dibutuhkan ketika tugas yang akan dikerjakan terlalu kompleks untuk diprogramkan (Shalev-Shwartz & Ben-David, 2014). Shalev-Shwartz dan Ben-David juga menjelaskan bahwa terdapat dua aspek masalah yang memerlukan program yang dapat melakukan pembelajaran terhadap data, aspek permasalahan tersebut adalah masalah kompleksitas dan kebutuhan akan adaptasi. Di dalam *machine learning*, kita perlu mengenal dan mengetahui tingkat kompleksitas data atau sampel yang digunakan untuk mengevaluasi ukuran sampel yang dibutuhkan algoritma untuk mempelajari data (Schapire & Freund, 2012).

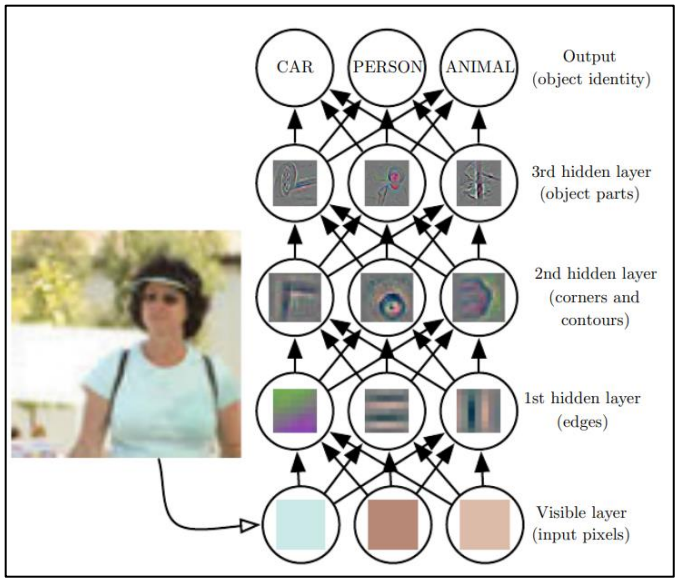
Kita dapat membagi *machine learning* ke dalam tiga kategori berdasarkan tujuannya (Mueller & Massaron, 2021b), yaitu (1) *supervised learning*, berlaku ketika sebuah algoritma belajar dari data yang terdiri dari nomor dan label, seperti kelas, *tag*, yang kemudian memprediksi dengan benar ketika diberikan data baru. (2) *unsupervised learning*, berlaku ketika sabuah algoritma belajar dari data yang biasa yang artinya tidak memiliki label atau kelas sehingga algoritma akan menentukan pola datanya sendiri. (3) *reinforcement learning*, berlaku ketika secara berurutan menyajikan algoritma dengan data yang tidak memiliki label. Namun, Mueller dan Massaron di dalam bukunya juga mengatakan bahwa setiap

data perlu disertai *feedback* positif atau negatif sesuai dengan solusi yang diajukan algoritma.

***Deep Learning***

*Deep learning* memanfaatkan *artificial neural network* sebagai algoritma untuk meniru cara kerja pikiran manusia dan *deep learning* ini merupakan pengembangan dari *machine learning. Deep learning* dapat membuat dan menghasilkan sebuah penghubung terkait dengan kemampuan dari beberapa algoritma untuk membuat keputusan sebaik mungkin menggunakan seluruh data yang dibutuhkan, yang formatnya tidak dapat dipahami oleh algoritma *machine learning* (Mueller & Massaron, 2021a).

Dalam *deep learning*, sebagian besar parameter model dipelajari tidak langsung dari *feature* pada data *training*, tetapi dari *output* pada lapisan sebelumnya (Burkov, 2019). Burkov juga menyebutkan bahwa *Deep learning* mengacu pada pelatihan *neural network* dengan lebih dari dua *non-output layers* dan menggunakan peralatan algoritmik dan matematis yang modern terlepas dari seberapa dalam *neural network* tersebut.




**Gambar 3.** Ilustrasi Model *Deep Learning* (Goodfellow et al., 2016)

Ketika *deep learning* diimplementasikan, awalnya system komputer akan kesulitan untuk memahami arti dari data *input* yang diberikan, seperti data gambar yang direpresentasikan dengan kumpulan nilai *pixel* dan *deep learning* akan mengatasi kesulitan ini dengan memecah pemetaan *pixel* yang kompleks ke dalam beberapa pemetaan yang lebih sederhana dan setiap pemetaan tersebut dituangkan ke dalam lapisan yang berbeda-beda pada model yang digunakan (Goodfellow et al., 2016).

***Computer Vision***

*Computer vision* adalah sebuah ilmu dari *machine learning* yang bertujuan untuk memberikan pembelajaran pada sistem dalam hal memahami atau menginterpretasi gambar dan video. *Computer vision* telah diterapkan di beberapa inovasi, seperti *autonomous driving*, *industrial inspection*, dan *Augmented Reality* (AR). Penggunaan *deep learning* pada *computer vision* dapat dikategorikan ke dalam beberapa kategori, yaitu klasifikasi, deteksi, segmentasi, dan generasi, baik pada gambar maupun video (Shanmugamani & Moore, 2018).

*Vision* atau penglihatan memungkinkan manusia untuk mendeteksi dan memahami dunia di sekitarnya, sementara *computer vision* bertujuan untuk menduplikasi efek penglihatan manusia dengan memahami gambar secara elektronik (Sonka et al., 2014). Memberikan kemampuan pada komputer untuk melihat bukan merupakan tugas yang mudah. Ketika komputer mencoba menganalisis objek dalam ruang tiga dimensi (3D), sensor visual seperti kamera dan televisi biasanya hanya menghasilkan dua dimensi (2D) gambar, dan proyeksi ke jumlah dimensi yang lebih rendah ini menyebabkan hilangnya informasi yang sangat besar (Sonka et al., 2014).

## 2.5 *Object Detection*

*Object detection* merupakan tugas yang dimiliki oleh komputer yang berhubungan dengan mendeteksi contoh objek visual pada kelas tertentu, seperti manusia, hewan, dan sebagainya dalam gambar atau citra digital dan tujuan dari *object detection* adalah untuk mengembangkan model dan teknik komputasi (Zou et al., 2019). Beberapa sistem pendeteksian objek akhir-akhir ini menggunakan pengklasifikasian untuk melakukan deteksi. Untuk mendeteksi objek, sistem melakukan klasifikasi pada objek tersebut dan mengevaluasinya di berbagai lokasi dan skala pada gambar uji atau *test image* (Redmon et al., 2016).

Deteksi objek dapat dikelompokkan ke dalam dua jenis, yaitu deteksi hal tertentu secara spesifik (*detection of specific instances*) dan deteksi pada kategori yang luas (*detection of broad categories*) (Liu et al., 2020). Liu et al menjelaskan dua jenis deteksi objek tersebut, yaitu *detection of specific instances* yang bertujuan untuk mendeteksi objek tertentu, seperti wajah manusia, spesies hewan, dan sebagainya, dan *detection of broad categories* yang bertujuan untuk mendeteksi sesuatu dari beberapa kategori objek yang telah ditentukan sebelumnya, seperti manusia dan benda-benda di sekitar.

## 2.6 *Digital Image*

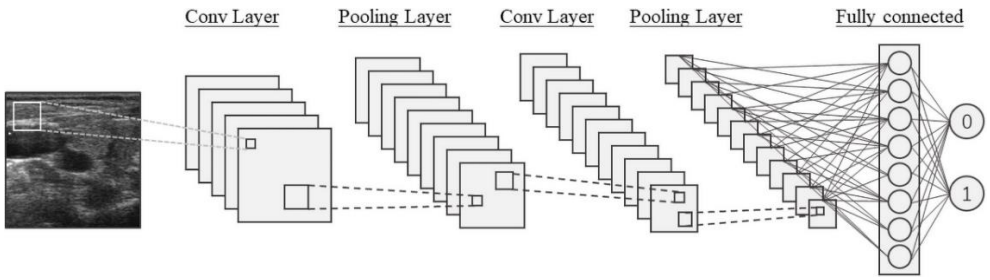
Dalam ilmu *computer vision*, data gambar yang diproses umumnya merupakan *digital image* atau citra digital yang mengacu pada larik dua dimensi dan larik ini disebut piksel. *Digital image* merupakan sebuah citra $f(x,y)$ yang telah didiskritisasi (proses memecah daerah perhitungan menjadi beberapa daerah kecil yang disebut dengan *grid, mesh,* atau *cell*) baik dari segi koordinat spasial maupun kecerahannya (Petrou & Petrou, 2010). Petrou dkk juga menjelaskan bahwa pada *digital image*, untuk setiap pita warna (*colour band*) direpresentasikan oleh *array* dua dimensi (2D) bertipe *integer* atau serangkaian *array* 2D.

Langkah pertama dalam pemrosesan citra digital pada *low-level image processing techniques* adalah citra atau gambar ditangkap oleh sensor (seperti kamera) dan didigitalkan, kemudian komputer mengurangi *noise* pada citra (*image pre-processing*) dan meningkatkan beberapa fitur objek yang relevan untuk mendeteksi gambar seperti melakukan *edge extraction* atau ekstraksi tepi (Sonka et al., 2014).

## 2.7 *Convolutional Neural Network* (CNN)

*Convolutional Neural Network* (CNN) merupakan salah satu algoritma *deep learning* yang merupakan pengembangan dari *Multilayer Perceptron* (MLP) dan dirancang untuk mengolah data dalam bentuk dua dimensi, seperti suara dan gambar (Ilahiyah & Nilogiri, 2018). CNN memiliki cara kerja yang sama dengan MLP, namun dalam CNN tiap neuron direpresentasikan dalam bentuk dua dimensi dan MLP dalam satu dimensi.

*Neural* Network terdiri dari berbagai *layer* dan beberapa neuron pada tiap *layer*. Jenis-jenis *layer* dan jumlah neuron yang digunakan tidak dapat ditentukan dengan aturan yang pasti dan akan berlaku aturan yang berbeda-beda pada data yang berbeda pula (Stathakis, 2009). Utamanya, CNN terdiri dari beberapa *layer*, yaitu *input layer*, *convolutional layer*, *pooling layer*, *fully connected layer*, dan *output layer* dan CNN terbagi lagi menjadi dua proses, yaitu *forward* dan *back propagation* (Yu et al., 2019).




**Gambar 4.** Arsitektur Single-channel CNN (Zhang et al., 2022)

*Input layer* merupakan lapisan yang berguna untuk menampung nilai pixel dari gambar yang diinputkan dan pada umumnya adalah sebuah matriks $N \times r \times r$ (Yu et al., 2019). Komputer memiliki sebuah mesin yang hanya dapat membaca angka. Sebuah gambar yang terdapat dalam komputer merupakan matriks yang berisi nilai dari setiap pixel pada gambar. Sebagai contoh, diinputkan gambar dengan ukuran $128 \times 128$ pixel dengan tiga kanal warna RGB, maka akan dimasukkan menjadi matriks berukuran $128 \times 128 \times 3$.

***Feature Extraction***

*Feature extraction* merupakan *stage* pertama dalam proses algoritma CNN. Pada bagian ini dilakukan "*encoding*" pada citra menjadi *features* berupa angka-angka yang merepresentasikan citra tersebut. Pada proses ini terdapat beberapa *layer*, yaitu *convolutional layer* dan *pooling layer*.

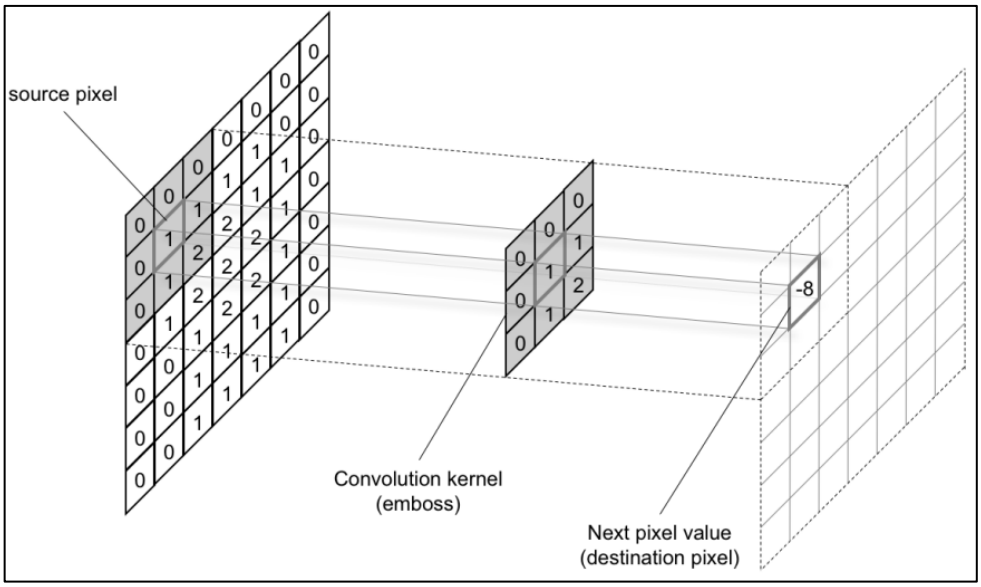
***Convolutional Layer.*** *Convolutional Layer* berfungsi untuk mengenali atribut-atribut unik pada objek. *Convolutional layer* akan melakukan operasi konvolusi pada *output* dari *layer* sebelumnya. Prinsip kerja dari *convolutional layer* adalah sebagai berikut :

$$x_j^l = f(\Sigma_{i \epsilon S_j} x_i^{l-1} * \omega_{ij}^l + b_j^l) \quad (1)$$

Pada rumus (1), $S_j$ merepresentasikan gambar yang diinput, $x_i^{l-1}$ merepresentasikan sebuah *feature map* dari input ke-$i$ layer ke-$(l-1)$, $\omega_{ij}^l$ merepresentasikan kernel konvolusi dari sebuah *input feature map* ke-$i$ layer ke-$(l-1)$ hingga *output feature map* ke-$j$ layer ke-$l$, $b_j^l$ merepresentasikan nilai bias sesuai dengan *output feature map* ke-$j$ layer ke-$l$, $f()$ merepresentasikan fungsi aktivasi, * merepresentasikan operasi konvolusi dan akan menghasilkan *feature map* $x_i^l$ ke-$j$ layer ke-$l$ (Yu et al., 2019).

*Convolutional* layers merupakan bagian utama dari arsitektur CNN. Pada tahap ini dilakukan operasi konvolusi pada *output* fungsi lain secara berulang (Hidayat et al., 2019). Hidayat et al juga menjelaskan bahwa operasi konvolusi menerapkan fungsi *output* sebagai *feature map* dari citra dan tujuan dari konvolusi pada data citra adalah untuk mengekstrak *features* dari cita masukan, selanjutnya konvolusi akan menghasilkan transformasi linier dari data masukan sesuai dengan informasi spasial pada data tersebut. Bobot atau *weight* pada *layer* menentukan kernel konvolusi mana yang digunakan sehingga kernel konvolusi dapat dilatih berdasarkan *input* (Hidayat et al., 2019).

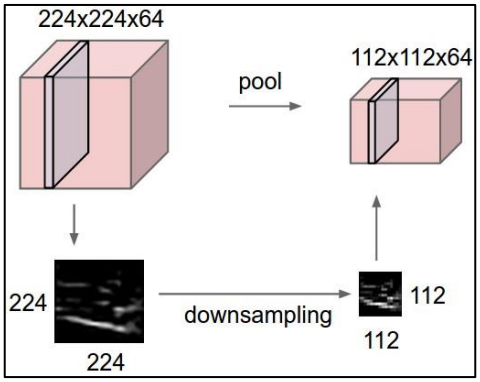
**Kernel.** *Kernel* merupakan parameter dari *convolutional layer* yang biasa digunakan untuk memproses (*convolve*) gambar dan *kernel* memiliki dua parameter, yaitu *stride* dan *size* (Shanmugamani & Moore, 2018).



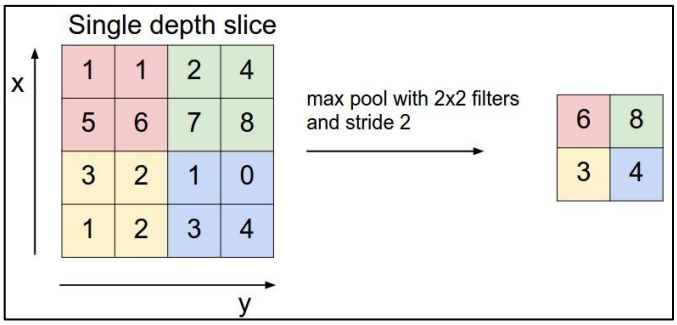

**Gambar 5.** Operasi Konvolusi (Shanmugamani & Moore, 2018)

Shanmugamani dan Moore menjelaskan bahwa *size* dapat berupa dimensi persegi panjang apapun dan *stride* merupakan jumlah *pixel* yang dipindahkan setiap waktu. Sebuah *stride* dengan panjang (*length*) 1 menghasilkan gambar dengan *size* yang hampir sama dan *stride* dengan panjang 2 akan menghasilkan setengah *size* dan untuk penambahan *padding* pada gambar atau citra akan membantu dalam pencapaian ukuran yang sama dengan gambar input (Shanmugamani & Moore, 2018).

***Pooling Layer.*** *Pooling Layer* atau juga disebut dengan *down-pooling layer* pada umumnya terdiri dari dua jenis, yaitu *max pooling* dan *average pooling* (Andrej Karpathy et al., 2015). Andrej Karpathy et al menjelaskan bahwa prinsip kerja *layer* ini persis seperti prinsip kerja *convolutional layer*, bedanya adalah fungsi aktivasi pada *convolutional layer* diubah menjadi *pooling function* yang merupakan sebuah proses untuk mengurangi resolusi gambar dengan tetap mempertahankan informasi pada gambar.




**Gambar 6.** *Pooling Layer* (Andrej Karpathy et al., 2015)




**Gambar 7.** *Max Pool in Pooling Layer* (Andrej Karpathy et al., 2015)

Seperti pada Gambar 6 dan Gambar 7, *pooling* dengan operasi *down sampling* berfungsi untuk mengurangi volume input secara spasial (mengurangi jumlah parameter) namun tetap mempertahankan kedalaman volume (Andrej Karpathy et al., 2015).

***Batch Normalization.*** Proses pelatihan dalam *deep neural network* sangat kompleks karena pendistribusian nilai *input* pada setiap *layer* berubah-ubah pada saat pelatihan karena parameter dari lapisan sebelumnya berubah. Proses distribusi nilai *input* yang selalu berubah ini menyebabkan proses pelatihan menjadi lambat dengan mengharuskan *learning rate* yang lebih rendah dan proses inisialisasi parameter yang lebih hati-hati, serta mempersulit pelatihan model dengan meningkatnya nonlinearitas (Ioffe & Szegedy, 2015).

Ioffe & Szegedy, 2015 mengajukan sebuah teknik yang disebut dengan *batch normalization* untuk mengatasi masalah *vanishing/exploding gradients*. Teknik ini terdiri dari menambahkan sebuah operasi pada model sebelum atau setelah fungsi aktivasi pada setiap *hidden layer, zero-centering* dan menormalkan setiap nilai *input*, lalu menskalakan dan menggeser hasilnya menggunakan dua vektor parameter pada tiap *layer*, satu untuk penskalaan, dan lainnya untuk penggeseran (Géron, 2019). Dengan kata lain, operasi ini dapat memungkinkan model untuk mempelajari skala dan rata-rata yang optimal pada nilai *input* setiap *layer.*

Secara umum, *batch normalization* merupakan mekanisme yang bertujuan untuk menstabilkan distribusi (melalui *mini batch*) nilai *input* ke *layer* tertentu pada saat pelatihan (Santurkar et al., 2018). *Batch normalization* mempengaruhi stabilitas pada proses pembelajaran dan mengurangi jumlah *epoch* yang diperlukan pada saat pelatihan untuk melatih model (Purnomo & Tjandrasa, 2021).

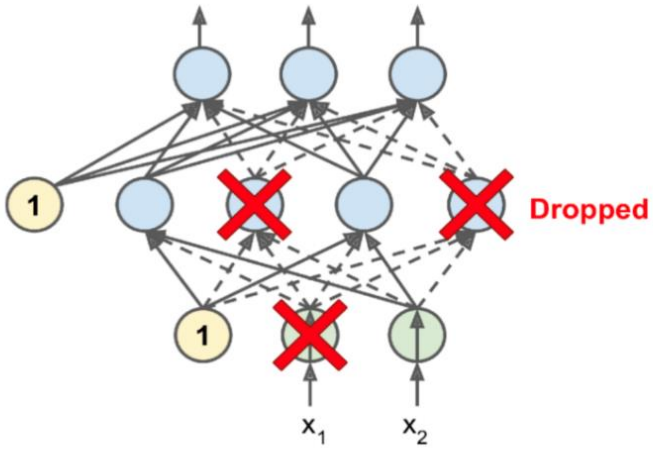
***Fully Connected Layer***

*Fully connected layer* akan menjadikan *output* pada *pooling layer* yang berupa *feature map* sebagai *input*. *Feature map* dari *pooling layer* masih berupa

multidimensional array sehingga *fully connected layer* akan melakukan *reshape* terhadap *feature map* dan akan menghasilkan vektor sebanyak n-dimensi, di mana n merupakan jumlah kelas *output* yang harus dipilih program (Andrej Karpathy et al., 2015). Proses *reshape* ini juga dikenal sebagai *flattening*, yang juga dapat berarti satu atau lebih *convolutional layer* yang berupa *multidimensional array* diubah menjadi sebuah larik konvolusi 1D (Jin et al., 2014).

*Fully connected layer* merupakan sebuah *layer* di mana semua neuron aktivasi dari *layer* sebelumnya telah terhubung semua atau *all connected* dengan neuron pada *layer* berikutnya (Hidayat et al., 2019). Hidayat et al juga menjelaskan bahwa setiap aktivasi pada *layer* sebelumnya perlu dikonversikan ke dalam data satu dimensi sebelum dapat terhubung ke seluruh neuron pada *fully connected layer.* Proses transformasi pada dimensi data dilakukan agar data dapat diklasifikasikan secara linear.

***Dropout Regularization.*** *Dropout* merupakan salah satu teknik regulasi dalam *deep neural network*. Teknik regulasi merupakan teknik yang digunakan untuk mengurangi *overfitting*. *Overfitting* dapat dikurangi dengan menggunakan teknik *dropout*. Dalam prosesnya, *dropout* akan menghilangkan *neuron* secara acak dengan probabilitas setiap *neuron* yang diberikan bernilai antara 0 hingga 1, sehingga nantinya setiap *neuron* yang *hidden* maupun *visible* tidak dapat bergantung satu sama lain (Hinton et al., 2012).




**Gambar 8.** *Dropout Regularization* (Géron, 2019)

Mengacu pada Gambar 8, dengan menggunakan *dropout*, pada setiap langkah pelatihan, setiap *neuron* memiliki probabilitas p untuk "keluar" sementara, yang berarti *neuron* tersebut akan diabaikan selama pelatihan terjadi, namun terdapat kemunkginan *neuron* tersebut aktif pada langkah pelatihan selanjutnya. *Hyperparameter* p disebut dengan *dropout rate*, biasanya bernilai 0,5

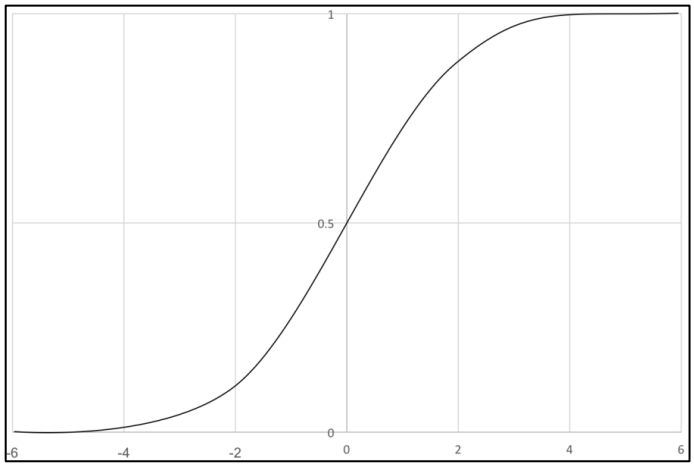
atau 50% dan setelah proses pelatihan, *neuron* yang dikeluarkan sebelumnya akan kembali seperti semula.

***Klasifikasi***

***Activation Function.*** *Activation function* akan membuat *neural network* menjadi non-linear (Shanmugamani & Moore, 2018). Pada saat proses *training*, *activation function* memiliki peran yang sangat penting dalam proses penyesuaian gradien. *Dalam neural network programming*, *activation* atau *transfer function* dapat menggunakan beberapa fungsi aktivasi yang berbeda dalam menetapkan batas pada *output* dari *neuron* dan *neural networks* (Heaton, 2015). Heaton juga menyebutkan bahwa pemilihan fungsi aktivasi bagi *neural networks* merupakan hal yang sangat penting untuk dipertimbangkan karena dapat mempengaruhi bagaimana cara memberikan format data *input*.

Fungsi aktivasi berfungsi sebagai penyaring yang akan membantu dalam pemilihan kelas yang relevan dan akan menghindari kelas yang memiliki probabilitas lemah (Mueller & Massaron, 2021b). Beberapa fungsi aktivasi yang biasa digunakan, yaitu Sigmoid, Tanh, ReLU, dan Softmax.

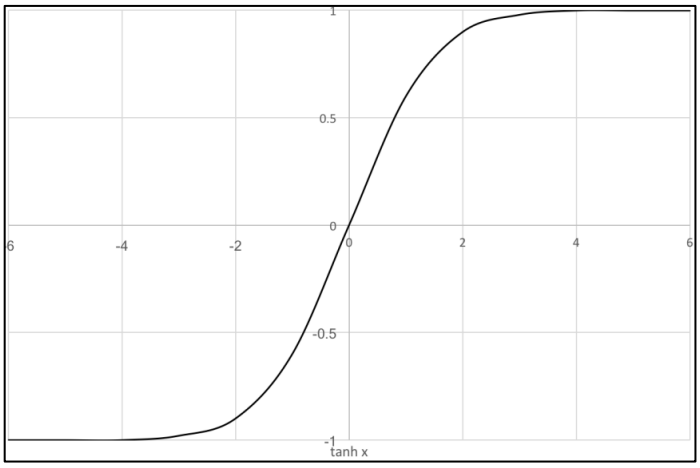
Fungsi aktivasi sigmoid akan mengubah nilai *input* menjadi sebuah angka antara 0 sampai 1 seperti grafik berikut:




**Gambar 9.** Grafik Fungsi Aktivasi Sigmoid

Perubahan nilai y terhadap x semakin mengecil, dan jika gradien memiliki nilai yang sangat kecil, sigmoid akan menghilangkan gradien (Shanmugamani & Moore, 2018). Hilangnya gradien ini merupakan kekurangan dari fungsi aktivasi sigmoid.

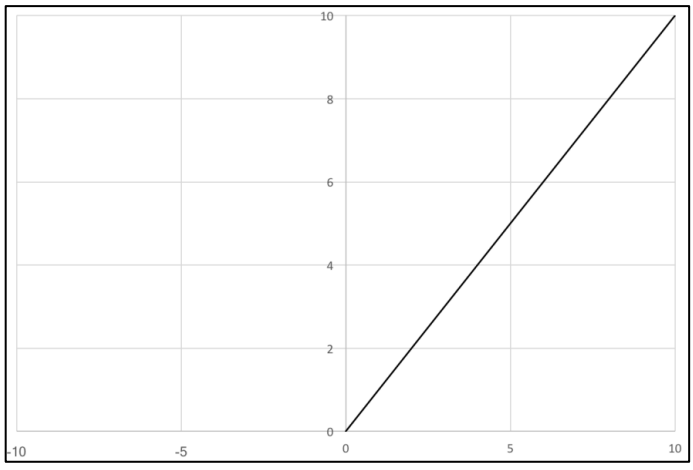
Fungsi aktivasi TanH atau *Hyperbolic Tangent* merupakan fungsi aktivasi sigmoid yang dikembangkanm hanya saja perbedaannya terdapat pada pemetaan nilai *input* ke dalam rentang nilai antara -1 hingga 1 seperti grafik berikut:




**Gambar 10.** Grafik Fungsi Aktivasi TanH

Gradien pada fungsi aktivasi TanH lebih stabil dibandingkan dengan sigmoid dan karena hal tersebut, gradien yang dihilangkan lebih sedikit, namun ini fungsi aktivasi TanH tidak menyelesaikan *vanishing gradient* yang sering terjadi pada *sigmoid* (Nwankpa et al., 2018). Baik fungsi aktivasi sigmoid maupun TanH, keduanya dapat menyebabkan proses *training* menjadi berat.

Fungsi aktivasi ReLU merupakan fungsi aktivasi yang paling sering digunakan dalam *Deep Learning*. Dalam prosesnya, ReLU memetakan *input* $(0, x)$, di mana *input* yang bernilai negatif akan menghasilkan keluaran bernilai 0, sedangkan jika *input* bernilai positif, maka nilai tersebut tidak mengalami perubahan pada keluaran (Shanmugamani & Moore, 2018). Pemetaan nilai *input* dengan fungsi aktivasi ReLU dapat divisualisasikan seperti pada grafik berikut:




**Gambar 11.** Grafik Fungsi Aktivasi ReLU

Keuntungan dari penggunaan fungsi aktivasi ReLU ini adalah proses komputasi yang lebih cepat dibandingkan dengan fungsi aktivasi lainnya. Selain itu, fungsi ini memperbaiki nilai *input* yang kurang dari 0, dengan demikian 'memaksa' nilai *input* tersebut menjadi nilai 0 dan menghilangkan masalah *vanishing gradient*

yang biasa ditemukan jika menggunakan fungsi aktivasi sigmoid dan TanH (Nwankpa et al., 2018).

Fungsi aktivasi Softmax dalam *deep learning* biasanya digunakan pada *fully connected layer* sebagai *output* dari klasifikasi model. Fungsi aktivasi ini digunakan untuk menghitung distribusi probabilitas dari sebuah vektor bilangan riil (Nwankpa et al., 2018). Fungsi aktivasi ini menghasilkan *output* dengan rentang nilai antara 0 hingga 1, dan dengan jumlah probabilitas sama dengan 1. Fungsi aktivasi *softmax* dapat dihitung dengan persamaan berikut :

$$f(x_i) = \exp(x_i) / \Sigma_j \exp(x_j) \quad (1)$$

Keterangan :

$x$ = nilai *input* lapisan sebelumnya

$i$ = indeks unit

$j$ = lapisan

Fungsi aktivasi *softmax* juga biasanya digunakan dalam pemodelan klasifikasi *multiclass*. Dalam penggunaannya pada lapisan *output* dari arsitektur *deep learning,* fungsi aktivasi *Sigmoid* digunakan dalam *binary classification,* sedangkan *Softmax* digunakan dalam *multiclass classification* (Nwankpa et al., 2018).

***Callbacks***

*Callback* merupakan sebuah fungsi yang dapat dimanfaatkan jika pada saat *training* model mengalami beberapa permasalahan, terutama jika proses *training* membutuhkan waktu yang cukup lama dan biasanya masalah ini ditemukan pada *dataset* yang besar (Géron, 2019). Selain itu, *callback* juga dapat digunakan jika ingin menyimpan *trained model* pada saat *training,* baik pada interval tertentu maupun pada interval dengan akurasi terbaik. Berikut beberapa contoh *callbacks* yang terdapat pada *Keras Library:*

**Tabel 1.** *Callbacks* dalam *Keras Library*

| No | *Callback* | Deskripsi |
|---|---|---|
| 1. | *Model Checkpoint* | Menyimpan model atau bobot model dalam beberapa interval |
| 2. | *Backup and Restore* | Memulihkan pelatihan dari gangguan yang terjadi pada saat eksekusi |
| 3. | *TensorBoard* | Memberikan *output* berupa visualisasi hasil dari evaluasi model selama pelatihan berlangsung |
| 4. | *Early Stopping* | Menghentikan proses pelatihan jika nilai yang dimonitor tidak mengalami peningkatan |
| 5. | *ReduceLROnPlateau* | Mengurangi nilai *learning rate* jika nilai yang dimonitor tidak mengalami peningkatan |

**Evaluasi Performa**

Dalam tahap evaluasi performa model, pengukuran performa dapat dilakukan dengan menggunakan berbagai *evaluation metrics,* namun *evaluation*

*metric* yang paling umum digunakan dalam model kalsifikasi adalah *confusion matriks*. *Confusion matrix* akan menunjukkan hasil analisa terhadap seberapa baik *classifier* dapat mengenali *record* data dari kelas yang berbeda-beda.

***Confusion Matrix.*** Dalam proses pengklasifikasian, hasil akurasi yang baik merupakan salah satu hal yang harus diperhatikan. Sebuah confusion matrix dengan ukuran n×n menunjukkan hasil prediksi dan performa dari proses klasifikasi, yang mana n merupakan jumlah kelas (Visa et al., 2011). Confusion matrix dengan n=2 dapat ditunjukkan sebagai berikut.

**Tabel 2.** *Confusion Matrix* pada Klasifikasi Dua Kelas (Visa et al., 2011)

| | PREDICTED NEGATIVE | PREDICTED POSITIVE |
|---|---|---|
| ACTUAL NEGATIVE | *a* | *b* |
| ACTUAL POSITIVE | *c* | *d* |

Pada Tabel 2, nilai *a* (*True Negative*) merupakan angka prediksi negatif yang benar, nilai *b* (*False Positive*) merupakan angka prediksi positif yang salah, nilai *c* (*False Negative*) merupakan angka prediksi negatif yang salah, dan nilai *d* (*True Positive*) merupakan angka prediksi positif yang benar.

Keakuratan dan *error* dari prediksi pada pengklasifikasian dari tabel *confusion matrix* dapat dihitung dengan:

$$accuracy = \frac{a + d}{a + b + c + d} \tag{2}$$

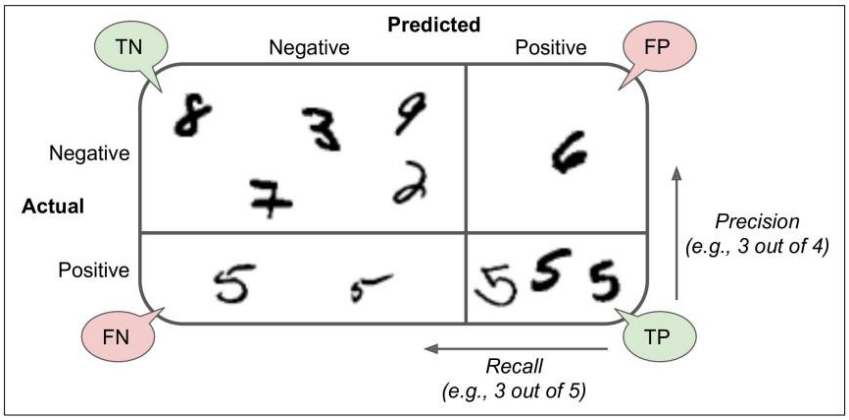
$$error = \frac{b + c}{a + b + c + d} \tag{3}$$

*Confusion matrix* memberikan berbagai informasi, namun informasi tersebut dapat diringkas dengan melihat hasil *precision, recall,* dan *f1 score*. Penjelasan mengenai *precision*, *recall*, dan *f1 score* disajikan pada tabel berikut:

**Tabel 3.** *Precision, Recall*, dan *F1 Score* (Géron, 2019)

| No | *Callback* | Perhitungan | Deskripsi |
|---|---|---|---|
| 1. | *Precision* | $\frac{d}{d + b}$ | Rasio dari perbandingan antara jumlah prediksi positif yang benar dengan seluruh data yang diprediksi positif |
| 2. | *Recall* | $\frac{d}{d + c}$ | Rasio dari perbandingan antara jumlah prediksi positif yang benar dengan banyaknya data yang sebenarnya positif |
| 3. | *F1 Score* | $2 \times \frac{precision \times recall}{precision + recall}$ | *Harmonic mean* dari *precission* dan *recall*. *F1 Score* akan menghasilkan nilai yang baik (skor terbaik 1,0 dan terburuk 0) jika skor *recall* dan *precission* juga baik |

Berikut merupakan ilustrasi sederhana dari *confusion matrix*:




**Gambar 12.** Ilustrasi *Confusion Matrix* (Géron, 2019)

## 2.8 *TensorFlow Library*

Tensorflow merupakan *library open source* untuk pengembangan dan penerapan model *deep learning*. TensorFlow menggunakan grafik komputasi untuk aliran data atau *data flow* dan komputasi numerik (Shanmugamani & Moore, 2018). Dengan kata lain, data atau *tensor* mengalir melalui grafik sehingga dinamakan TensorFlow.

Di dalam TensorFlow, setiap data gambar yang dimasukkan, biasanya direpresentasikan sebagai *tensor* 3D dalam bentuk [tinggi, lebar, *channel*] dan bias pada *convolutional layer* sederhananya direpresentasikan sebagai *tensor* 1D dengan bentuk [$f_n$] (Géron, 2019). Géron juga menyebutkan bahwa nilai *pixel* pada setiap warna *channel* direpresentasikan sebagai sebuah *byte* antara 0 sampai 255, sehingga kita perlu menskalakan data fitur dengan membagi 255 untuk mendapatkan nilai *float* atau nilai desimal mulai dari 0 hingga 1.

## 2.9 REST API

*Web service* merupakan sebuah metode komunikasi antara dua perangkat komputer melalui sebuah jaringan dan komunikasi ini terjadi dalam integrasi yang terstandarisasi dan terspesifikasi pada berbagai jenis aplikasi web menggunakan XML/JSON, SOAP, WSDL, dan UDDI (Subramanian & Raj, 2019). XML/JSON merupakan format data yang menyediakan *metadata* untuk data yang ada di dalamnya; SOAP untuk mentransfer data; WSDL digunakan untuk mendefinisikan layanan yang tersedia untuk dikonsumsi; UDDI memberikan daftar layanan yang tersedia.

*Web service* bertujuan untuk membangun *server* web yang mendukung kebutuhan situs atau aplikasi lainnya. Program *client* menggunakan *application programming interfaces* (APIs) untuk berkomunikasi dengan *web service* (Masse, 2011). Masse juga menjelaskan bahwa umunya API memiliki sekumpulan data

dan fungsi untuk memfasilitasi proses interaksi antara setiap program komputer dan mengizinkan program-program tersebut untuk saling bertukar informasi.

**Tabel 4.** Komponen HTTP *Request* pada REST API (Richardson et al., 2013)

| No | Komponen | Deskripsi |
|---|---|---|
| 1. | *GET* | Mendapatkan representasi dari sumber daya |
| 2. | *POST* | Membuat sumber daya baru di bawahnya, berdasarkan representasi yang diberikan |
| 3. | *PUT* | Menggantikan status sumber daya sesuai dengan representasi yang diberikan |
| 4. | *DELETE* | Menghapus sumber daya |

## 2.10 Flask

Flask merupakan *web framework* dan termasuk ke dalam jenis *microframework* yang ditulis dengan bahasa pemrograman Python (*Flask's Documentation*, 2022). Flask berfungsi sebagai *framework* sekaligus tampilan dari suatu *website* dan dengan menggunakan Flask, sebuah *website* dapat dikembangkan secara terstruktur dan dapat mengatur *behaviour website* dengan lebih mudah (Irsyad, 2018).

## 2.11 Penelitian Terdahulu

Dalam melakukan penelitian, perlu adanya acuan penelitian terdahulu sehingga dapat diketahui kontribusi penelitian yang dilakukan terhadap ilmu pengetahuan. Penelitian yang berkaitan dengan deteksi penyakit menggunakan metode CNN sudah pernah diteliti dalam penelitian lain, tetapi pendeteksian PMK pada sapi dengan metode CNN dan TensorFlow *library* belum dilakukan dalam penelitian lain terutama di Indonesia. Berikut merupakan penelitian yang berkaitan atau serupa dengan penelitian yang akan dilaksanakan oleh peneliti.

**Tabel 5.** Daftar Penelitian Terdahulu

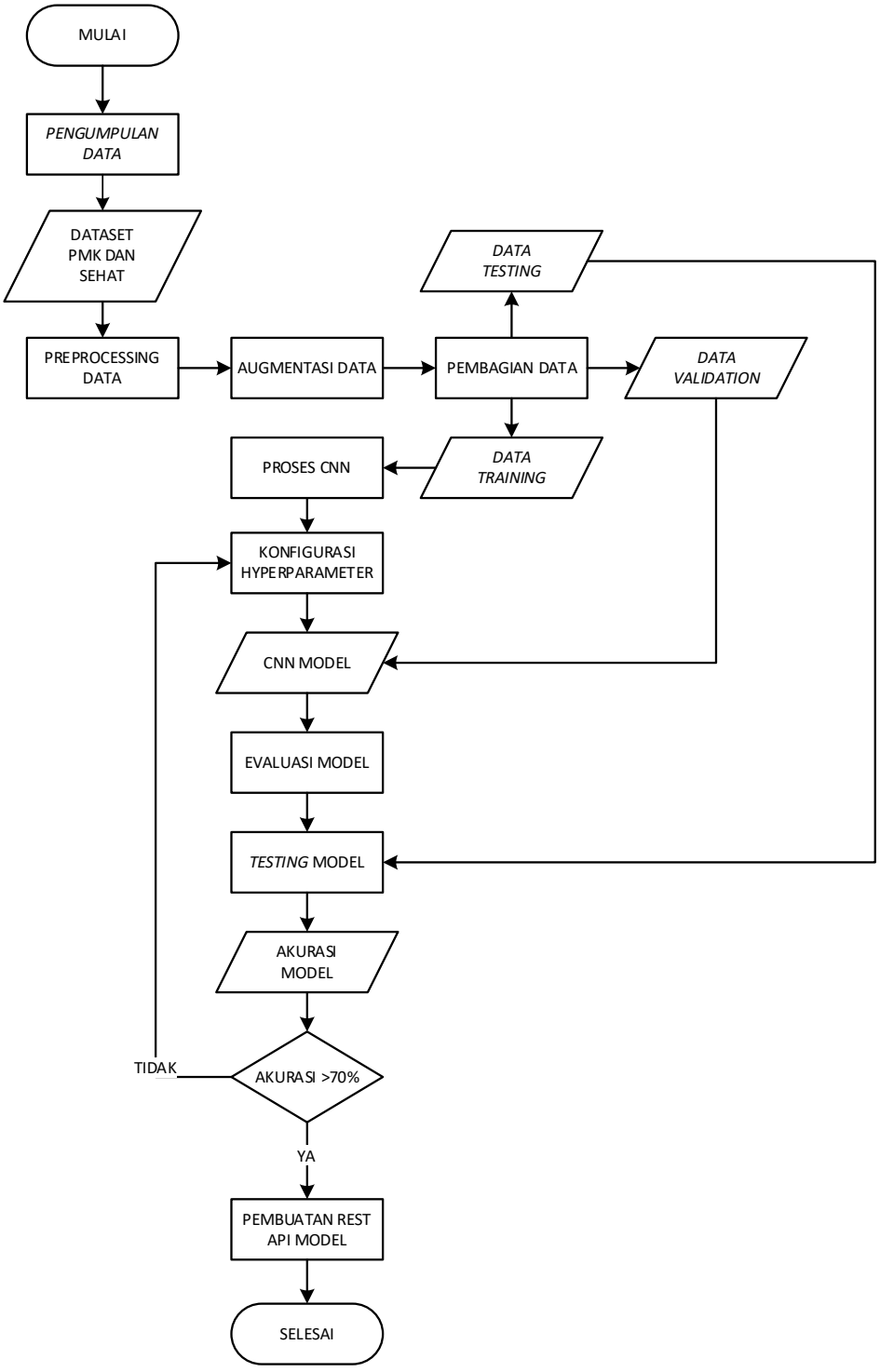
| No | Penulis/Tahun | Judul Penelitian | Metode | Hasil Penelitian |
|---|---|---|---|---|
| 1. | Kuhamba, 2020 | A Deep Learning Based Approach for Foot and Mouth Disease Detection | VGG16, Inception v3, Denset 201, Resnet 15v2, Resnet50, Inception Resnet v2 | Arsitektur terbaik berdasarkan hasil uji adalah Densenet 201 yang menghasilkan akurasi tertinggi yaitu sebesar 93.75%. |
| 2. | Prastika & Zuliarso, 2021 | Deteksi Penyakit Kulit Wajah Menggunakan TensorFlow dengan Metode Convolutional Neural Network | Convolutional Neural Network (CNN) | Hasil uji citra penyakit kulit menunjukan persentase akurasi yang berbeda-beda dan menghasilkan tingkat akurasi tertinggi mencapai 99,91%. |

| | | | | |
|---|---|---|---|---|
| 3. | Saputra dkk., 2021 | Penerapan Algoritma Convolutional Neural Network dan Arsitektur MobileNet pada Aplikasi Deteksi Penyakit Daun Padi | Convolutional Neural Network (CNN) | Hasil akurasi *training* mencapai 1.0 dan nilai *validation* mencapai 0.8333. Nilai akurasi pada Confussion Matrix yaitu sebesar 92%. |
| 5. | Ilahiyah & Nilogiri, 2018 | Implementasi Deep Learning pada Identifikasi Jenis Tumbuhan Berdasarkan Citra Daun Menggunakan Convolutional Neural Network | Convolutional Neural Network (CNN) | Rata-rata akurasi dari hasil klasifikasi mencapai 85%. Sedangkan akurasi dari pengujian berhasil mencapai 90% yang didapatkan dari pengujian 40 citra. |
| 7. | Weni dkk., 2021 | Detection of Cataract Based on Image Features Using Convolutional Neural Networks | Convolutional Neural Network (CNN) | Akurasi tertinggi adalah 95%. Setelah pengujian secara akurat, 10 gambar mencapai akurasi rata-rata 88%. |
| 8. | Putri dkk., 2022 | Identification of Incung Characters (Kerinci) to Latin Characters Using Convolutional Neural Network | Convolutional Neural Network (CNN) | Akurasi data training mencapai 99% dan akurasi data testing mencapai 91%. |

# III. METODOLOGI PENELITIAN

## 3.1 Tahapan Penelitian

Berikut merupakan diagram alir dari tahapan-tahapan penelitian yang akan dilakukan.

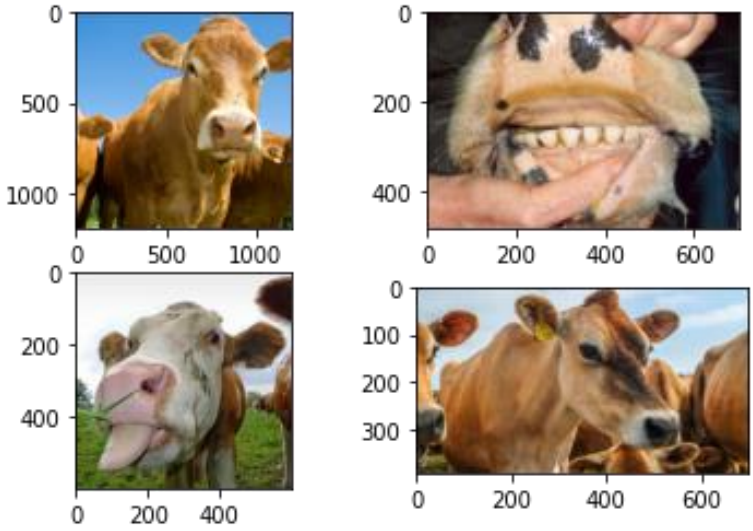



**Gambar 13.** *Flowchart* Penelitian

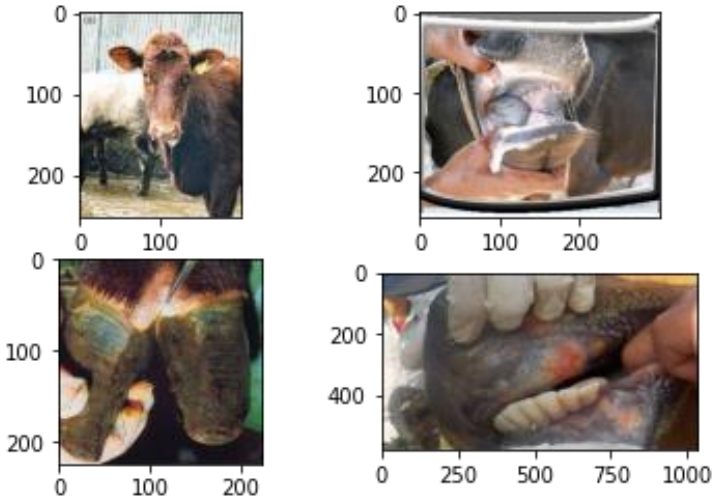
***Pengumpulan Data***

Pada penelitian ini, *dataset* citra diperoleh dari beberapa sumber, yaitu:

1. Gambar sapi yang terjangkit PMK yang diunduh melalui internet. Proses pengambilan data dilakukan dengan *scraping* gambar dan data gambar dari Dinas Pertanian dan Ketahanan Pangan Kota Jambi.
2. Gambar sapi yang tidak terjangkit PMK yang diambil di lingkungan Fakultas Peternakan Universitas Jambi.



**Gambar 14.** *Dataset* Sapi Sehat



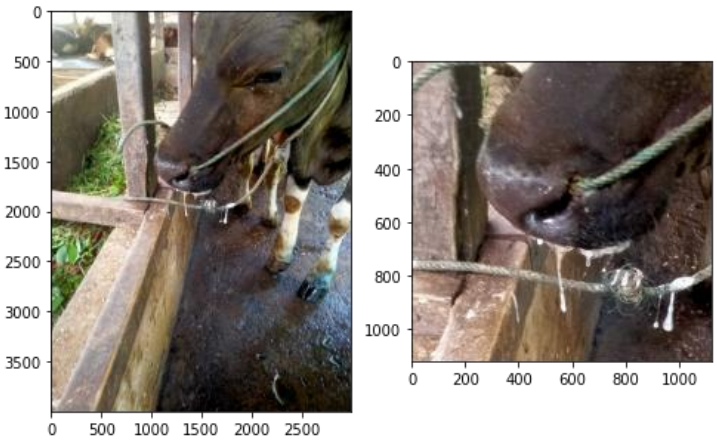
**Gambar 15.** *Dataset* Sapi Terjangkit PMK

***Image Preprocessing dan Augmentation***

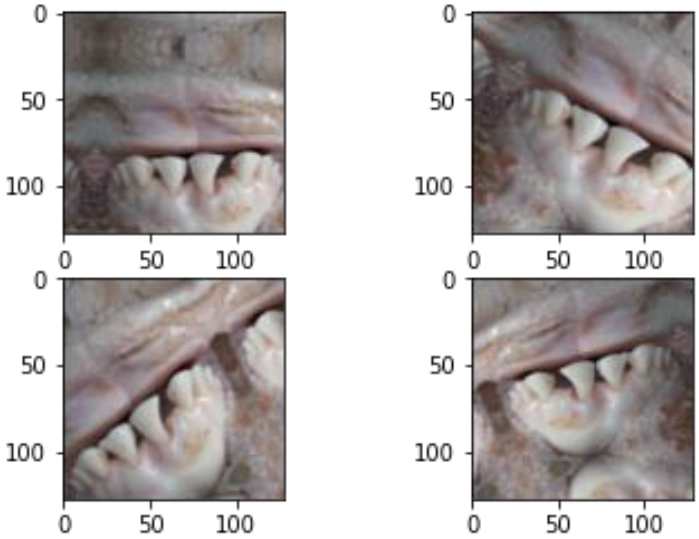
Pada tahap ini data gambar diolah lebih lanjut agar data siap dipakai dalam pengembangan *machine learning*. Gambar atau citra perlu diklasifikasikan ke dalam dua kelas yang berbeda karena jenis yang digunakan adalah *supervised*

*learning*. *Dataset* dibagi dan dikelompokkan ke dalam dua bagian, yaitu *data training* dan *data validation*. *Data training* dan *data validation* masing-masing kemudian dibagi lagi menjadi 8 kelas, yaitu Air Liur, Gusi, Kaki, dan Lidah untuk Gejala PMK dan Air Liur, Gusi, Kaki, dan Lidah untuk yang Sehat.

Data yang sudah ada kemudian masuk ke tahap *preprocessing* dengan melakukan *resize* untuk menyamakan ukuran gambar dan penyeleksian *region of interest* (ROI). Setelah diproses, selanjutnya gambar diaugmentasi. Augmentasi gambar dilakukan untuk memodifikasi dan memanipulasi citra yang asli. Proses ini akan memperbanyak data gambar sehingga mesin dapat mengenali dan mempelajari bentuk citra yang berbeda-beda. Beberapa teknik augmentasi yaitu *rotation, flip, zoom,* dan *fill mode*.



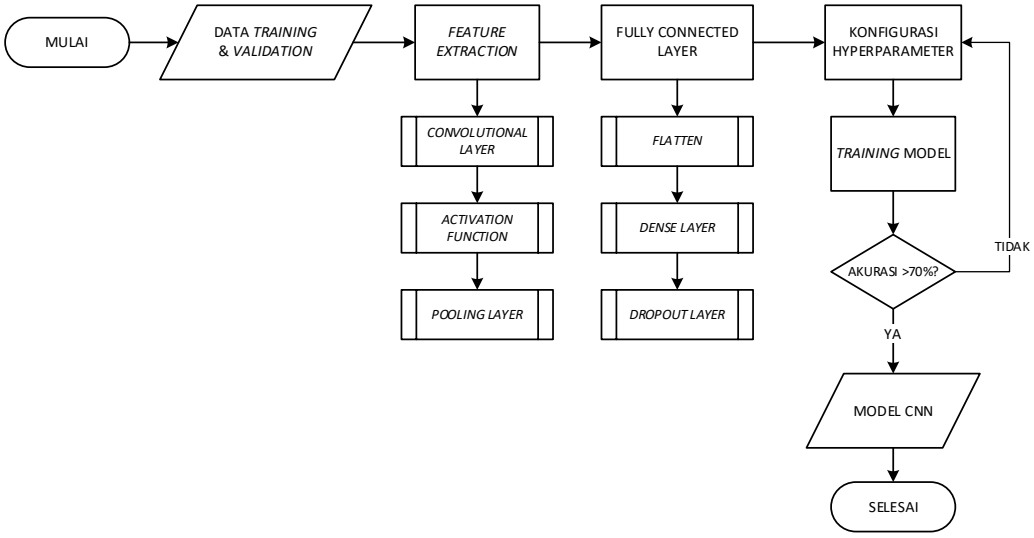

**Gambar 16**. Sebelum *Resize* dan Menentukan ROI (kiri), Setelah Resize dan ROI (kanan)




**Gambar 17.** Hasil Augmentasi Gambar

***Model Selection***

Pada tahap ini model CNN akan dipakai dan dilakukan optimasi parameter dari model yang digunakan. Dilakukan pelatihan menggunakan *data training* sehingga terbentuk sebuah model dari proses latihan *dataset* tersebut. Kemudian akan dilakukan prediksi dari model yang didapatkan. Berikut merupakan proses dalam pembuatan model CNN:




**Gambar 18.** Proses Pembuatan Model CNN

***Evaluasi Model***

Pada tahap ini evaluasi terhadap model dilakukan dan nilai performa yang dihasilkan terhadap *training* diamati dengan bantuan *confusion matrix*. Pada tahap ini juga dilakukan pengecekan apakah model yang digunakan *underfit* atau *overfit*.

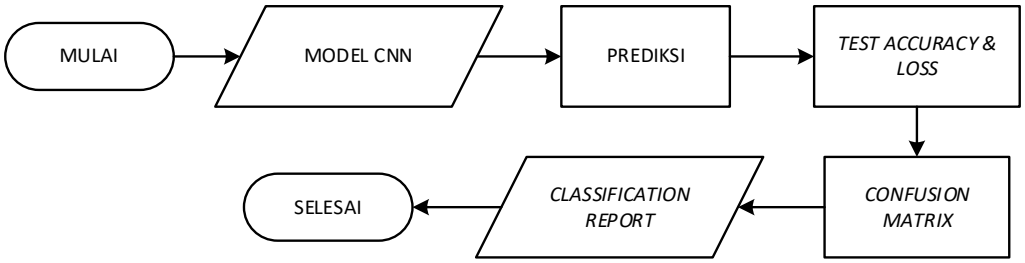
*Underfit* akan terjadi jika model yang dikembangkan terlalu sederhana dan tidak mampu menyesuaikan pola yang terdapat pada *data training*. Sebuah model dapat disebut sebagai *underfit* jika model tersebut menghasilkan nilai *error* yang tinggi pada *data training*. *Underfitting* menandakan bahwa model yang dikembangkan tersebut belum cukup baik dalam mengenali pola data yang terdapat pada *data training*.

*Overfit* akan terjadi jika model menghasilkan nilai prediksi terlalu baik pada *data training*, namun nilai prediksinya buruk pada *data testing*. Ketika model yang dikembangkan mengalami *overfit*, model akan membuat banyak kesalahan dalam memprediksi data-data baru yang ditemukan selama tahap produksi.

## 3.2 Pengujian Model dalam Deteksi Gejala Awal PMK

Pada bagian ini akan dilakukan pengujian performa pada model yang telah dibuat. Pengujian ini menggunakan data *testing* dan bertujuan untuk

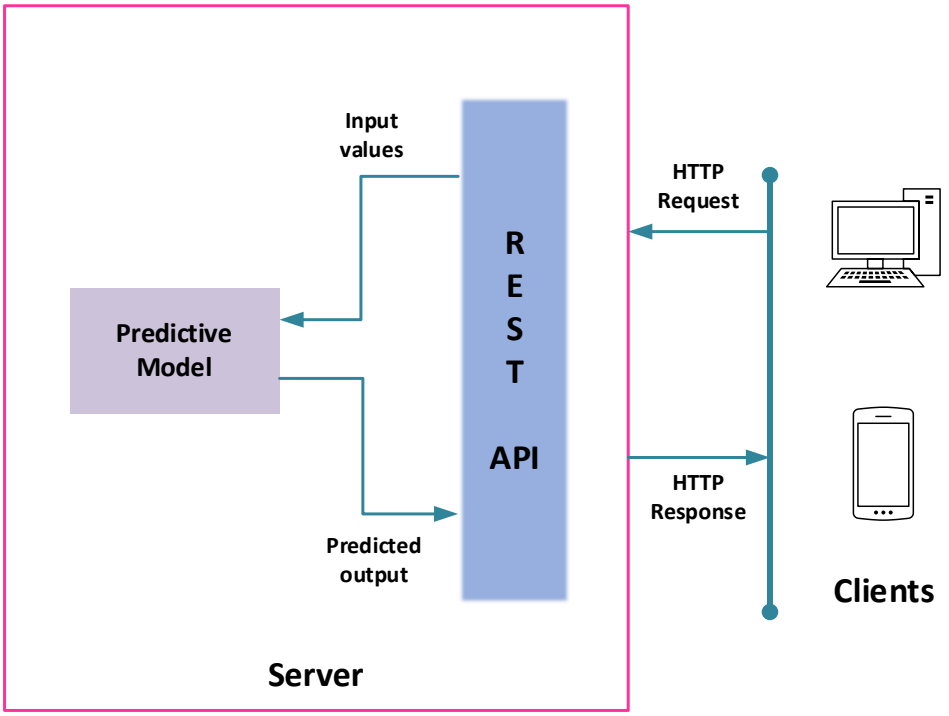
mendapatkan informasi detail mengenai akurasi prediksi pada seluruh data *testing*. Dari pengujian ini, akan didapatkan informasi mengenai akurasi, *loss*, *confusion matrix*, termasuk nilai *precision*, *recall*, dan *f1 score*. Berikut merupakan diagram alir pada proses pengujian performa model:




**Gambar 19.** *Flowchart* Proses Pengujian Model

### 3.3 Menghubungkan Model ke API

Pada tahap ini, model CNN yang telah dibuat dihubungkan dengan *web service* dan *interface* yang berupa API yang dapat melakukan simulasi pendeteksian gejala PMK. Dalam pengintegrasian ini, digunakan Flask sebagai *framework web*.




**Gambar 20**. Arsitektur Implementasi Model *Machine Learning* Sebagai *Service* (Balamurugan, 2020).

### 3.4 Bahan dan Alat Penelitian

Penelitian ini membutuhkan data gambar sapi yang terjangkit PMK dan yang tidak terjangkit PMK. Data gambar yang digunakan diperoleh melalui

internet dengan teknik *scraping* pada Google dan diambil secara langsung dengan menggunakan *device* seperti telepon seluler.

Alat yang dibutuhkan dalam penelitian ini adalah sebagai berikut.

1. Perangkat keras (*Hardware*)

**Tabel 6.** Spesifikasi Komputer

| Spesifikasi | Keterangan |
|---|---|
| *Processor* | Intel(R) Core(TM) i3-7020U CPU @ 2.30GHz 2.30 GHz |
| RAM | 12,0 GB |
| *Storage* | 1TB |
| *Graphic Card* | Intel® HD Graphics 620 |
| *Operating System* | Windows 11, 64-bit operating system, x64-based processor |

2. Perangkat lunak (*Software*)

**Tabel 7.** Spesifikasi Google Colab

| Spesifikasi | Keterangan |
|---|---|
| Python *Version* | Python 3.8.15 |
| RAM | 13 GB |
| *Storage* | 100 GB |

### 3.5 Waktu dan Tempat Penelitian

Penelitian ini dilakukan pada awal semester ganjil tahun ajaran 2022/2023 dan data yang didapatkan berasal dari lingkungan Fakultas Peternakan Universitas Jambi dan Internet dengan waktu yang dibutuhkan yaitu ± 6 bulan.

**Tabel 8.** *Timeline* Penelitian

| Kegiatan | Jun | Jul | Agu | Sep | Okt | Nov | Des |
|---|---|---|---|---|---|---|---|
| *Literature review* | | | | | | | |
| Penulisan skripsi | | | | | | | |
| *Data acquisition* | | | | | | | |
| *Preprocessing data* | | | | | | | |
| Perancangan model | | | | | | | |
| Pengujian model | | | | | | | |
| Pengujian klasifikasi gambar | | | | | | | |
| Implementasi model ke GUI | | | | | | | |

# IV. HASIL DAN PEMBAHASAN

Pada bab ini, penulis akan menguraikan tahap-tahap dalam memperolah hasil akhir penelitian berdasarkan metode yang telah dijelaskan pada bab sebelumnya.

## 4.1 Pengumpulan *Dataset*

Pada tahap awal penelitian, penulis mengumpulkan *dataset* gejala luka pada sapi yang terjangkit PMK dan sehat. Jumlah keseluruhan data yang berhasil dikumpulkan penulisadalah sebanyak 160 data gambar dan data gambar ini diperoleh dari beberapa sumber, yaitu mengunduh gambar dari Google dengan teknik *scraping*, Dinas Pertanian dan Ketahanan Pangan Kota Jambi, dan Lingkungan Fakultas Peternakan Universitas Jambi.

### *Web Scraping*

*Scraping* merupakan sebuah teknik yang digunakan untuk mengumpulkan beberapa data pada sebuah laman *website*. Dalam melakukan *scraping*, terjadi proses ekstraksi informasi dengan menggunakan HTTP (*Hypertext Transfer Protocol*).

Pada penelitian ini, *scraping* dilakukan secara otomatis menggunakan program dengan bahasa pemrograman Python. Program akan mengunduh gambar pada Google sesuai dengan *keyword* yang dimasukkan. Beberapa hal yang perlu dipersiapkan sebelum melakukan *scraping* menggunakan program ini, yaitu melakukan instalasi *Selenium Library, Pillow Library*, dan *WebDriver for Chrome*.

**Tabel 9.** Keyword yang Digunakan Saat Scraping

| ***Keyword*** | **Jumlah Gambar** |
|---|---|
| *foot and mouth disease cattle* | 63 |
| *foot and mouth disease cow* | 30 |
| *foot and mouth disease cow symptoms* | 28 |
| **Jumlah** | **121** |

Seluruh data yang didapatkan dari hasil proses *scraping* ditunjukkan pada Tabel 9. Gambar yang telah didapatkan dari proses *scraping* dengan beberapa *keyword* tersebut kemudian diseleksi. Penyeleksian atau *filtering* ini dilakukan untuk memperoleh gambar yang baik untuk digunakan sebagai *dataset* dan sesuai dengan kelas yang telah ditentukan. Data yang didapatkan dari proses *scraping* adalah berjumlah 14 gambar.

**Gambar 21.** Data Gambar Hasil *Scraping*

***Dinas Pertanian dan Ketahanan Pangan Kota Jambi***

Data yang didapatkan dari Dinas Pertanian dan Ketahanan Pangan Kota Jambi merupakan data gambar gejala luka pada sapi yang terjangkit PMK. Data gambar diambil oleh dokter hewan di dinas setempat saat kasus PMK sedang mewabah di kota Jambi, pada awal hingga pertengahan tahun 2022. Data yang didapatkan dari Dinas Pertanian dan Ketahanan Pangan Kota Jambi adalah sebanyak 60 gambar.

**Gambar 22.** Data Gambar dari Dinas Pertanian dan Ketahanan Pangan Kota Jambi

***Lingkungan Fakultas Peternakan Universitas Jambi***

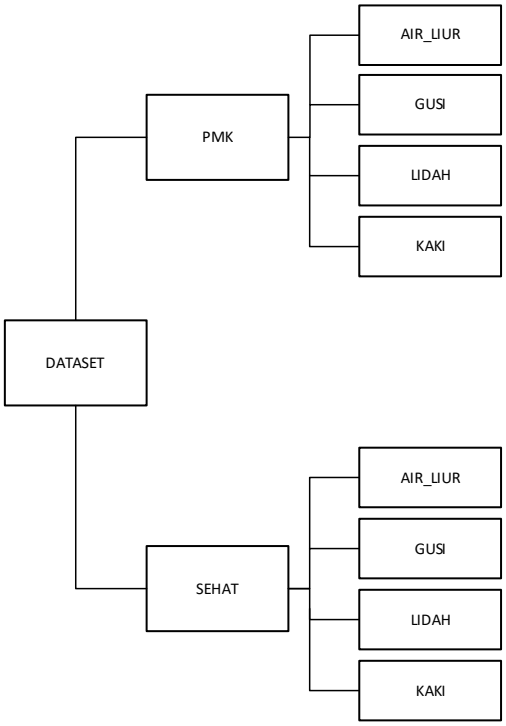
Data yang didapatkan dari Dinas Pertanian dan Ketahanan Pangan Kota Jambi merupakan data gambar area kaki dan mulut sapi yang sehat atau tidask terjangkit PMK. Data gambar diambil secara langsung oleh penulis pada tanggal

05 November 2022 menggunakan kamera *smartphone* dan data yang didapatkan adalah sebanyak 86 gambar.

**Gambar 23.** Data Gambar dari Lingkungan Fakultas Peternakan Universitas Jambi

## 4.2 Pembagian *Dataset*

Sebelum dilakukan pembagian, *dataset* yang akan digunakan memiliki struktur direktori seperti berikut.




**Gambar 24.** Struktur Direktori *Dataset*

Pada setiap *folder*, jumlah *dataset* yang dimiliki berbeda-beda. Sebaran jumlah data citra pada tiap *folder* ditunjukkan pada

**Tabel 10.** Sebaran Jumlah Data Citra

| No | Direktori | Jumlah Data |
|---|---|---|
| 1. | Dataset/pmk/air_liur | 26 |
| 2. | Dataset/pmk/gusi | 19 |
| 3. | Dataset/pmk/lidah | 15 |
| 4. | Dataset/pmk/kaki | 18 |
| 5. | Dataset/sehat/air_liur | 30 |
| 6. | Dataset/sehat/gusi | 11 |
| 7. | Dataset/sehat/lidah | 17 |
| 8. | Dataset/sehat/kaki | 24 |
| **Total** | | **160** |

***Pengunduhan Dataset***

Dataset yang telah dikumpulkan dan telah diproses dengan melakukan seleksi ROI, kemudian diunggah di akun Kaggle milik penulis sehingga *dataset* ini dapat diunduh langsung di program pada saat ingin membuat model. Berikut merupakan kode program yang dibuat untuk mengunduh *dataset*.

```
os.environ['KAGGLE_USERNAME'] = "XXXXXXXXXX"
os.environ['KAGGLE_KEY'] = "XXXXXXXXXX"
!kaggle datasets download -d helvianizebua/roiselected
from zipfile import ZipFile
file_name = "/content/roiselected.zip"
with ZipFile(file_name, 'r') as zp:
  zp.extractall(path='/content/')
  print('done')
```

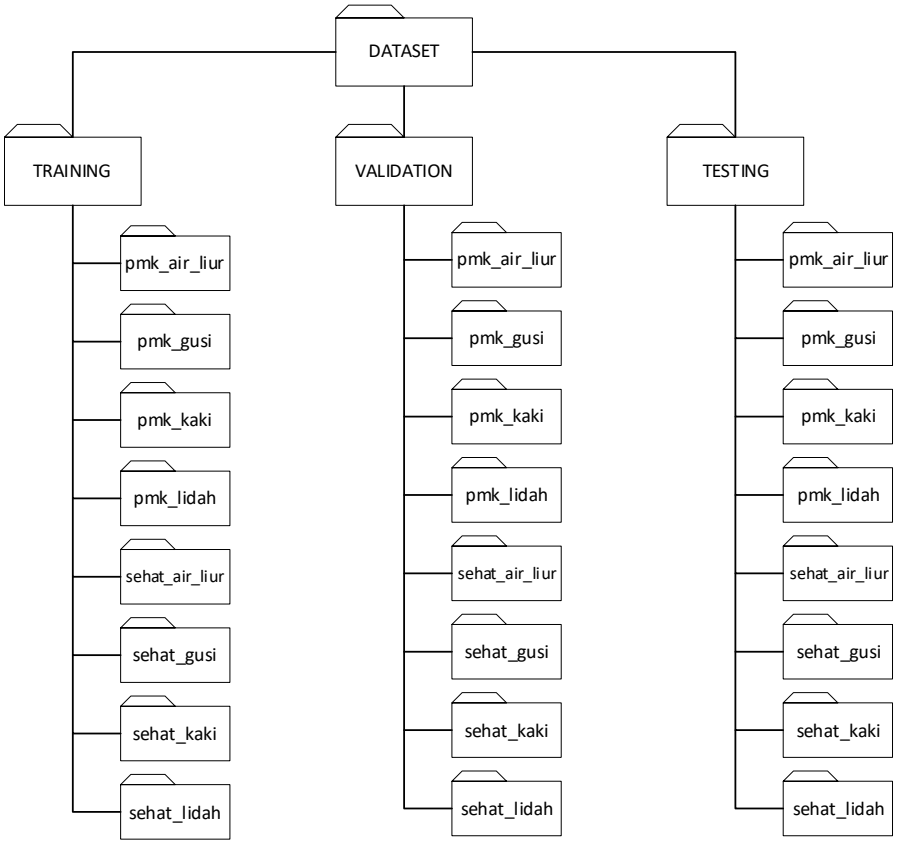
**Gambar 25.** Kode Program Pengunduhan *Dataset*

***Skenario Pembagian Data Training, Validation, dan Testing***

Setelah *dataset* berhasil diunduh, maka dataset akan tersimpan ke dalam folder yang bernama Dataset yang berisi 160 data citra. Folder ini memiliki struktur direktori seperti yang ditunjukkan pada Gambar 24. Sebelum ditentukan jumlah pembagian *dataset* pada data *training, validation,* dan *testing*, terlebih dahulu dibuat folder baru untuk menyimpan *dataset* yang akan dibagi nantinya. Kode program dan struktur direktori dalam pembuatan folder ini adalah sebagai berikut:

```
classes =  [
    'pmk_air_liur', 'pmk_gusi', 'pmk_kaki', 'pmk_lidah','sehat_air_liur', 'sehat_gusi',
'sehat_kaki', 'sehat_lidah']
base_dir = '/content/dataset'
train_dir = '/content/dataset/train/'
val_dir = '/content/dataset/validation/'
test_dir = '/content/dataset/test/'
os.mkdir(base_dir)
os.mkdir(train_dir)
os.mkdir(val_dir)
os.mkdir(test_dir)
for i in classes:
  os.mkdir(os.path.join(train_dir,i))
for j in classes:
  os.mkdir(os.path.join(val_dir,j))
for k in classes:
  os.mkdir(os.path.join(test_dir,k))
```

**Gambar 26.** Kode Program Membuat Folder *Training, Validation,* dan *Test*




**Gambar 27.** Struktur Direktori untuk Pembagian Dataset

*Dataset* yang tersedia kemudian akan ditentukan jumlah datanya yang dibagi menjadi data *training*, *validation*, dan *testing*.

**Tabel 11.** Skenario Pembagian Data

| **Skenario 70%:20%:10%** | | |
|---|---|---|
| *Training* | *Validation* | *Testing* |
| 70 | 20 | 10 |

## 4.3 *Augmentation* dan *Resize*

*Dataset* yang telah dibagi, kemudian diproses dengan melakukan *augmentation* dan *resize image*. Proses augmentasi ini bertujuan untuk memperbanyak data citra. Selain itu proses *resizing* bertujuan untuk menyamakan ukuran data citra.

Pada penelitian ini dalam proses augmentasi digunakan fungsi ImageDataGenerator dengan menerapkan beberapa teknik augmentasi, yaitu *rotation, width shift, height shift, shear, zoom, horizontal flip,* dan *fill mode*. Berikut merupakan deskripsi dari beberapa teknik yang digunakan:

**Tabel 12.** Teknik Augmentasi

| No | Teknik Augmentasi | Deskripsi |
|---|---|---|
| 1. | *rotation range* | Rentang derajat untuk melakukan rotasi pada citra secara acak |
| 2. | *width shift* | Menggeser gambar ke kiri atau ke kanan (secara horizontal) |
| 3. | *height shift* | Menggeser gambar ke atas atau ke bawah (secara vertikal) |
| 4. | *shear range* | Intensitas pergeseran sudut pada citra (sudut bergeser berlawanan arah jarum jam dalam derajat) |
| 5. | *Zoom range* | Rentang *zoom* secara acak |
| 6. | *horizontal flip* | membalikkan citra secara acak dan horizontal |
| 7. | *fill mode* | Aturan untuk bagian atau area piksel yang sudah digeser |

```
datagen = ImageDataGenerator(
        rotation_range=30,
        width_shift_range=0.2,
        height_shift_range=0.2,
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=True,
        fill_mode='constant')
```

**Gambar 28.** Kode Program Fungsi ImageDataGenerator

Pada proses *resizing*, data citra akan di-*resize* menjadi ukuran 150x150 piksel. Setelah dilakukan proses augmentasi dan *resizing*, jumlah data yang diperoleh sebanyak 3.757 data *training* dan 548 data *validation*.

```
def aug_img(input_dir, output_dir, nums):
  itr_num = nums
  itr = 0
  for each_data in classes:
    dataset = []
    image_directory = input_dir+each_data+'/'
    my_images = os.listdir(image_directory)
    for i, image_name in enumerate(my_images):
        if (image_name.split('.')[1] == 'jpg' or 'jpeg'):
            image = io.imread(image_directory + image_name)
            image = Image.fromarray(image, 'RGB')
            image = image.resize((IMG_WIDTH,IMG_HEIGHT))
            dataset.append(np.array(image))
    x = np.array(dataset)
    aug_dir = output_dir+each_data
    os.makedirs(aug_dir)
    i = 1
    for batch in datagen.flow(x, batch_size=16,
                              save_to_dir=aug_dir,
                              save_prefix='aug'+image_name,
                              save_format='png'):
        i += 1
        if i > itr_num[itr]:
            break
    itr +=1
    print('images are: '.format(i)+str(len(os.listdir(aug_dir))))
  print()
```

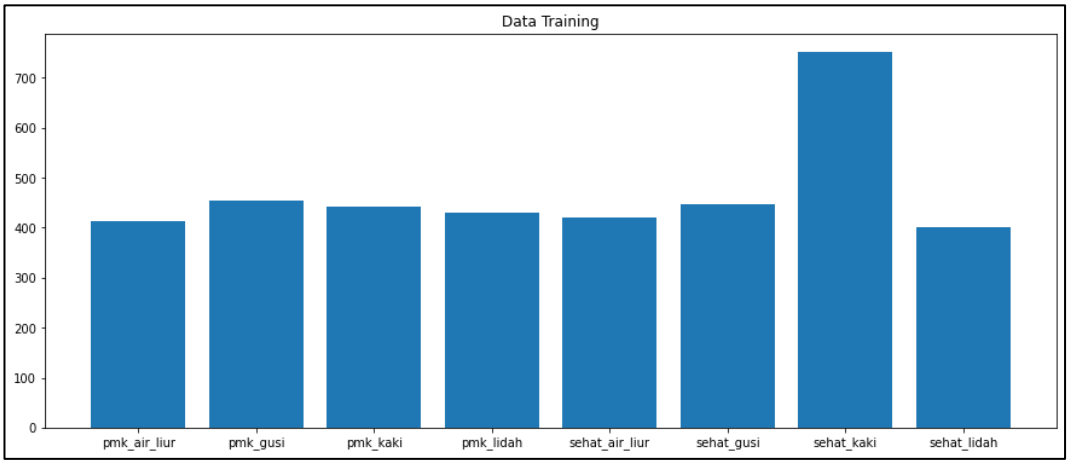
**Gambar 29.** Kode Program Fungsi untuk Proses Augmentasi

Pada kode program yang ditunjukkan Gambar 29. Kode Program Fungsi untuk Proses Augmentasi , untuk menjalankan proses augmentasi gambar, dilakukan pemanggilan fungsi ’aug_img’ dengan tiga parameter *input,* dengan deskripsi setiap parameter sebagai berikut:

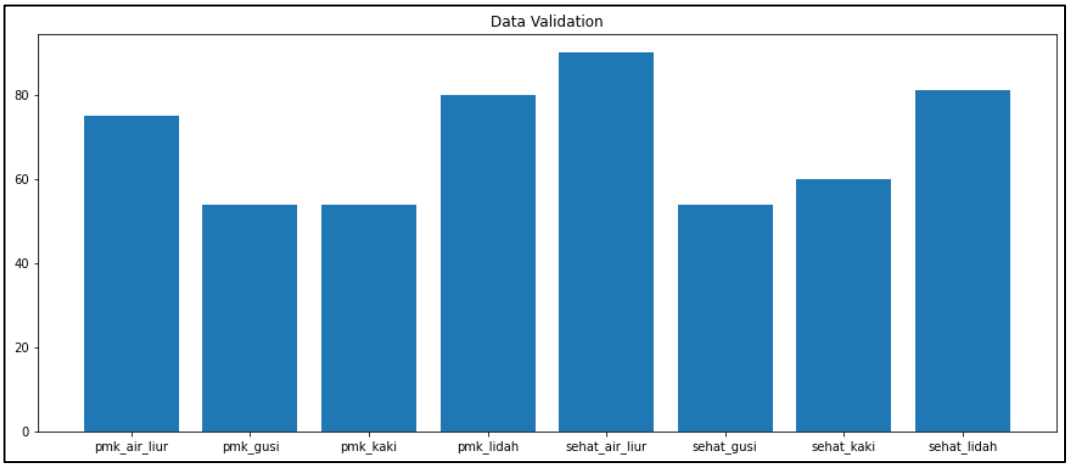
**Tabel 13.** Deskripsi Parameter pada Fungsi 'aug_img'

| No | Parameter | Deskripsi |
|---|---|---|
| 1. | input_dir | *Path* direktori di mana data gambar yang akan diaugmentasi disimpan |
| 2. | output_dir | *Path* direktori di mana data gambar yang telah melalui proses augmentasi disimpan |
| 3. | nums | Variabel bertipe *array* yang berisi serangkaian nilai yang mewakili tiap kelas. Nilai-nilai ini akan menentukan pada iterasi berapa proses augmentasi akan dihentikan dan melanjutkan proses augmentasi gambar pada kelas selanjutnya. |

Berikut merupakan visualisasi data diagram batang sebaran jumlah data *training* dan *validation* setelah proses augmentasi gambar:




**Gambar 30.** Visualisasi Jumlah Data *Training*




**Gambar 31.** Visualisasi Jumlah Data *Validation*

Berikut merupakan sebaran data citra pada direktori *training* dan *validation* setelah dilakukan proses augmentasi:

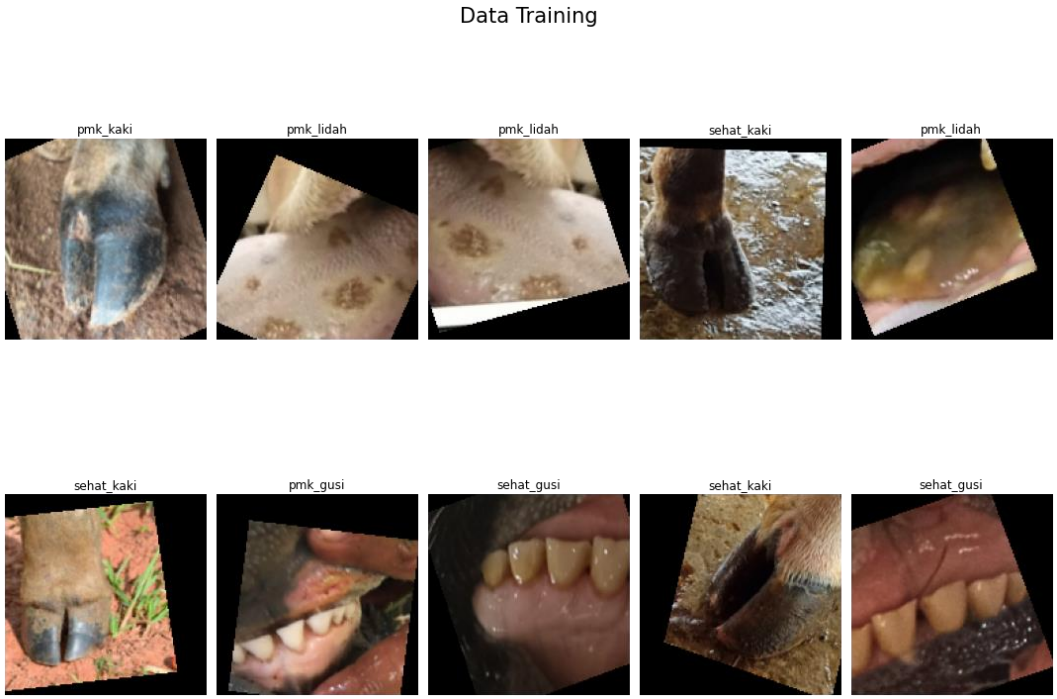
**Tabel 14.** Jumlah Data Citra pada *Training* Setelah Augmentasi

| No | Direktori | Jumlah Data |
|---|---|---|
| 1. | /content/dataset/train/pmk/air_liur | 412 |
| 2. | /content/dataset/train/pmk/gusi | 455 |
| 3. | /content/dataset/train/pmk/lidah | 429 |
| 4. | /content/dataset/train/pmk/kaki | 443 |
| 5. | /content/dataset/train/sehat/air_liur | 420 |
| 6. | /content/dataset/train/sehat/gusi | 447 |
| 7. | /content/dataset/train/sehat/lidah | 400 |
| 8. | /content/dataset/train/sehat/kaki | 751 |
| **Total** | | **3.757** |

**Tabel 15.** Jumlah Data Citra pada *Validation* Setelah Augmentasi

| No | Direktori | Jumlah Data |
|---|---|---|
| 1. | /content/dataset/validation/pmk/air_liur | 75 |
| 2. | /content/dataset/validation/pmk/gusi | 54 |
| 3. | /content/dataset/validation/pmk/lidah | 80 |
| 4. | /content/dataset/validation/pmk/kaki | 54 |
| 5. | /content/dataset/validation/sehat/air_liur | 90 |
| 6. | /content/dataset/validation/sehat/gusi | 54 |
| 7. | /content/dataset/validation/sehat/lidah | 81 |
| 8. | /content/dataset/validation/sehat/kaki | 60 |
| **Total** | | **548** |

Berikut merupakan sampel data *training* yang telah diaugmentasi:




**Gambar 32.** Sampel Data *Training*

Selanjutnya, label setiap kelas dibuat berdasarkan nama *folder* yang terdapat pada *folder train*. Berikut hasil pelabelan kelas:

```
0 : pmk_air_liur
1 : pmk_gusi
2 : pmk_kaki
3 : pmk_lidah
4 : sehat_air_liur
5 : sehat_gusi
6 : sehat_kaki
7 : sehat_lidah
```

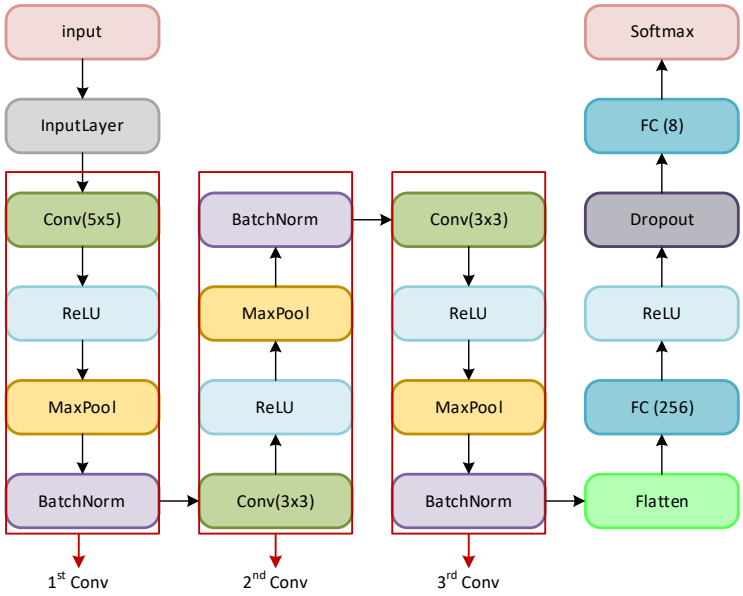
**Gambar 33.** Pemetaan Label

## 4.4 Perancangan Arsitektur

Setelah melalui beberapa tahapan, maka selanjutnya data yang telah diproses akan di-*training*. Arsitektur yang digunakan dalam pembuatan model deteksi gejala awal PMK pada sapi dengan metode CNN ini sangat mempengaruhi hasil dari akurasi model.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 146, 146, 128)     9728

activation (Activation)      (None, 146, 146, 128)     0

max_pooling2d (MaxPooling2D  (None, 73, 73, 128)       0
)

batch_normalization (BatchN  (None, 73, 73, 128)       512
ormalization)

conv2d_1 (Conv2D)            (None, 71, 71, 64)        73792

activation_1 (Activation)    (None, 71, 71, 64)        0

max_pooling2d_1 (MaxPooling  (None, 35, 35, 64)        0
2D)

batch_normalization_1 (Batc  (None, 35, 35, 64)        256
hNormalization)

conv2d_2 (Conv2D)            (None, 33, 33, 32)        18464

activation_2 (Activation)    (None, 33, 33, 32)        0

max_pooling2d_2 (MaxPooling  (None, 16, 16, 32)        0
2D)

batch_normalization_2 (Batc  (None, 16, 16, 32)        128
hNormalization)

flatten (Flatten)            (None, 8192)              0

dense (Dense)                (None, 256)               2097408

dropout (Dropout)            (None, 256)               0

dense_1 (Dense)              (None, 8)                 2056

=================================================================
Total params: 2,202,344
Trainable params: 2,201,896
Non-trainable params: 448
```

**Gambar 34.** *Model Summary*




**Gambar 35.** Arsitektur Model Deteksi Gejala Awal PMK pada Sapi dengan Metode CNN

*Input* gambar yang digunakan dalam penelitian ini adalah gambar yang berukuran 150x150x3. Pemodelan dengan metode CNN dapat dijelaskan sebagai berikut:

***Konvolusi Pertama***

Pada tahap konvolusi ini digunakan kernel berukuran 5x5 dengan jumlah filter 128. Selain itu juga digunakan *valid padding* yang tidak akan menambahkan nilai di sekitar *border input* sehingga nantinya dimensi *input* akan sama dengan dimensi *output*. Tahap ini merupakan proses kombinasi antara dua matriks yang berbeda dan tahap ini akan menghasilkan sebuah matriks baru. Setelah tahap konvolusi selesai, ditambahkan sebuah aktivasi yaitu RELU (*Retrified Linear Unit*) yang berfungsi untuk mengubah nilai negatif menjadi nol atau menghilangkan nilai negatif pada matriks hasil dari proses konvolusi.

Selanjutnya adalah tahap *pooling*. Pada tahap ini, terjadi pengurangan matriks dari hasil proses konvolusi. *Pooling layer* ini terdiri dari sebuah filter berukuran 2x2 yang mana ukuran nilai matriksnya akan bergeser secara bergantian pada area *feature map*. Proses *pooling* ini akan menggunakan aktivasi *maxpooling* yang bekerja dengan cara mengambil nilai maksimum pada setiap *pooling window* berukuran 2x2 di tiap pergeseran kernel. Pada konvolusi pertama ini juga diterapkan *batch normalization layer* untuk melakukan normalisasi pada *activation* di layer sebelumnya. *Batch normalization* ini menerapkan transformasi pada nilai *activation* agar mendekati nilai 0 dan nilai standar deviasi pada *activation* agar mendekati nilai 1.

***Konvolusi Kedua dan Ketiga***

Tahap konvolusi yang kedua merupakan tahap yang akan memproses hasil dari *pooling layer* sebelumnya dengan *input* matriks yang telah dinormalisasikan, yaitu 71x71 dengan jumlah *filter* 64 dan ukuran kernel 3x3. Pada tahap ini kembali digunakan RELU sebagai *activation function*. Selanjutnya pada tahap *pooling* yang memiliki proses yang sama dengan konvolusi sebelumnya dan menghasilkan *output* yang berukuran 30x30.

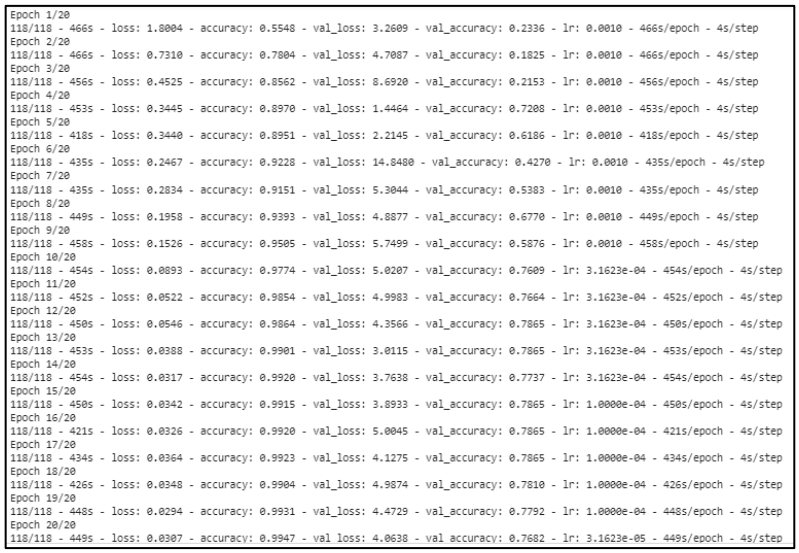
Tahap konvolusi yang ketiga juga memiliki proses yang sama dengan tahap konvolusi sebelumnya. *Input* matriks yang diproses pada konvolusi ketiga ini merupakan hasil dari *pooling layer* kedua yang telah dinormalisasikan yang berukuran 33x33 dengan jumlah *filter* 32 dan kernel berukuran 3x3. Pada tahap ini juga digunakan RELU sebagai *activation function*. *Pooling layer* ini akan menghasilkan *output* yang berukuran 16x16.

***Fully-Connected Layer***

Setelah melalui berbagai tahapan yang dijelaskan sebelumnya, berikutnya adalah tahap *flatten* atau *fully connected*. Di tahap ini hanya akan terdapat satu buah *hidden layer* yang akan mengubah *output* dari *pooling layer* di tahap sebelumnya yang berupa *array* dua dimensi menjadi data satu dimensi *single vector*.

Tahap berikutnya yaitu proses Dense yang berfungsi untuk menambah *hidden layer* dengan *node* yang berjumlah 256 dan untuk fungsi aktivasi digunakan RELU. Selanjutnya digunakan teknik regulasi jaringan syaraft tiruan yaitu *dropout regularization* yang akan menghilangkan *neuron* di dalam jaringan. Pada proses *dropout* ini diberikan probabilitas bernilai 0,5 pada setiap *neuron* dan proses ini akan mencegah *overfitting* sekaligus mempercepat proses pembelajaran.

Setelah proses *dropout*, dilakukan inisialisasi *layer output* yang berupa 8 *node* karena dalam pengkalsifikasian di penelitian ini terdapat 8 kelas, yaitu 'pmk_air_liur', 'pmk_gusi', 'pmk_kaki', 'pmk_lidah', 'sehat_air_liur', 'sehat_gusi', 'sehat_kaki', dan 'sehat_lidah'. Karena lebih dari dua kelas atau *multi-class classification*, maka akan digunakan fungsi aktivasi *softmax classifier*. Fungsi aktivasi *softmax* akan mengembalikan probabilitas dari masing-masing kelas dan kelas target akan memiliki nilai probabilitas lebih tinggi dibandingkan dengan kelas yang lain.




**Gambar 36.** *Training History*

***Inisialisasi Parameter Learning***

Tujuan dari perancangan serta melakukan pelatihan algoritma CNN ini adalah agar sistem dapat mengenali ciri dari setiap gambar dan kemudian menandai seluruh *neuron* serta menentukan dari setiap *neuron* tersebut yang mana yang akan diaktifkan ketika gambar diklasifikasi.

Sebelum dilakukan proses *training*, data *training* yang telah melalui proses augmentasi gambar perlu dipanggil terlebih dahulu dan perlu dilakukan inisialisasi terhadap beberapa parameter *learning*. Beberapa parameter *learning* yang perlu diinisialisasi untuk proses *training* ada tiga, yaitu *learning rate, batch size,* dan *epoch*.

**Tabel 16.** Inisialisasi Parameter

| Parameter | Deskripsi |
|---|---|
| *Learning rate* | Digunakan untuk menghitung nilai koreksi bobot pada saat proses training. Nilai learning rate yang diterapkan pada penelitian ini adalah 0,001 dengan *optimizer* Adam. |
| *Batch Size* | Digunakan untuk menentukan berapa jumlah sampel data yang disebarkan. Dalam penelitian ini nilai *batch size* yang digunakan adalah 32. |
| *Epoch* | Digunakan untuk penentu jumlah iterasi yang harus dilakukan dalam proses *training dataset*. Jumlah *epoch* pada penelitian ini adalah 20. |

```
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=np.sqrt(0.1), patience=5)
optimizer = Adam(learning_rate=0.001)
cnn_model.compile(optimizer=optimizer, loss=CategoricalCrossentropy(),
metrics=['accuracy'])
history = cnn_model.fit(train_generator, epochs=20,
validation_data=validation_generator,
                        verbose=2,
```

**Gambar 37.** Kode Program Inisialisasi Parameter

Pada kode program inisialisasi parameter yang ditunjukkan pada Gambar 37, ditambahkan fungsi *callback*, yaitu *ReduceLROnPlateau* dari *library* Keras. Fungsi ini akan memamntau atau memonitor sebuah kuantitas dan akan mengurangi atau memperkecil nilai *learning rate* jika *metric* atau nilai *loss* tidak mengalami perkembangan dalam sejumlah '*patience*' dari *epoch*. Pada fungsi ini digunakan beberapa parameter, yaitu :

**Tabel 17.** Parameter pada Fungsi ReduceLROnPlateau

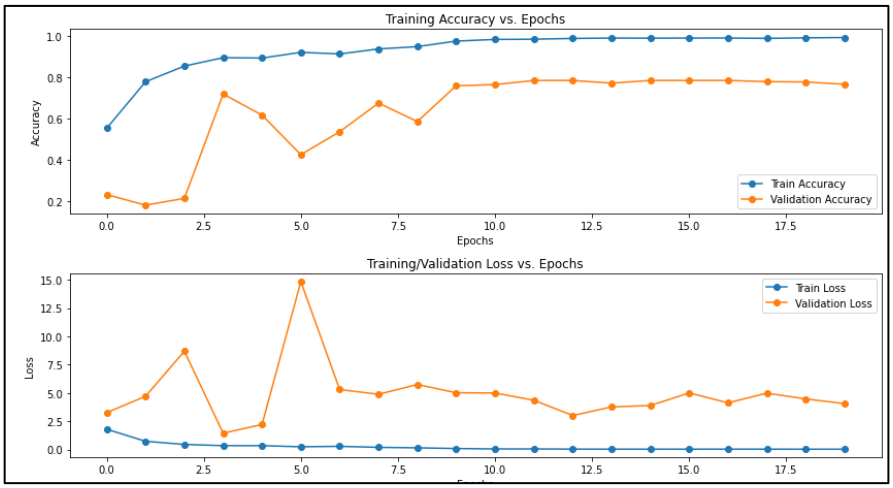
| Parameter | Deskripsi |
|---|---|
| *Monitor* | Kuantitas yang akan dimonitor. Pada penelitian ini kuantitas yang akan dipantau adalah nilai *validation loss.* |
| *Factor* | Faktor di mana nilai *learning rate* akan dikurangi ($new_{lr} = lr \times factor$). Nilai *factor* yang digunakan adalah $\sqrt{0{,}1}$. |
| *Patience* | Jumlah *epoch* yang tidak mengalami peningkatan dan setelah mencapai jumlah tersebut, *learning rate* dikurangi. Pada penelitian ini nilai *patience* yang digunakan adalah 5. |

Selain itu, dalam penelitian ini juga digunakan algoritma optimasi yaitu *Adam* (*Adaptive Moment Estimation*) *Optimizer*. Algoritma optimasi *Adam* ini merupakan salah satu algoritma optimasi yang banyak digunakan dalam *deep learning*. Beberapa penelitian mengenai *deep learning* telah menunjukkan bahwa dengan menggunakan *Adam* menghasilkan akurasi terbaik dibandingkan dengan algoritma lainnya (Bera & Shrivastava, 2020; Irfan et al., 2022; Wikarta et al., 2020). Kingma & Ba, 2014 memperkenalkan *Adam optimizer* pertama kali. Dalam *paper* tersebut, dapat disimpulkan beberapa keunggulan dari algoritma ini, yaitu efisien secara komputasi, persyaratan memori kecil, dan beberapa kelebihan lainnya.

Penelitian ini menggunakan salah satu jenis *multi-class classification loss function* yaitu *categorical cross entropy*. *Categorical cross entropy* merupakan *loss function cross entropy* yang digunakan untuk permasalahan klasifikasi multi-kelas. Dalam prosesnya, *cross entropy* akan meminimalkan nilai *loss* (nilai *cross entropy* yang sempurna adalah 0) dengan menghitung skor perbedaan antara distribusi probabilitas aktual dan prediksi untuk semua kelas (Géron, 2019). Berdasarkan hal tersebut, maka digunakan *categorical cross entropy* sebagai *loss function* karena pada penelitian ini klasifikasi yang digunakan lebih dari dua kelas.

## 4.5 Hasil Proses *Training*

Sesuai dengan skenario yang telah dijelaskan sebelumnya, pada bagian ini dilakukan percobaan dengan membagi seluruh data dengan data *training* sebesar 70%, data *validation* 20%, dan data *testing* 10%.




**Gambar 38.** Visualisasi Akurasi *Training* dan *Validation* pada Skenario 70:20:10

Gambar 38 merupakan grafik dari hasil akurasi *train, validation, loss,* dan *learning rate* selama proses *training* dengan skenario pembagian data 70:20:10.

***Training Accuracy***

Akurasi dari *training* dengan 20 *epochs* mengalami peningkatan pada setiap *epoch*. Dari proses *training* yang dilakukan, dapat disimpulkan bahwa rata-rata akurasi dari *train* adalah sebesar 93%. Akurasi tertinggi yang diperoleh terjadi pada iterasi ke-20, yaitu sebesar 99% dan akurasi terendah terjadi pada iterasi pertama, yaitu sebesar 55%.

***Validation Accuracy***

Akurasi dari validasi dengan 20 *epochs* mengalami naik turun bahkan mengalami *overfit* pada setiap iterasi di 10 *epochs* pertama. Setelah *epoch* ke 10, akurasi pada setiap *epoch* hampir selalu konsisten hingga *epoch* terakhir. Akurasi tertinggi yang didapatkan terdapat pada iterasi ke-17, yaitu sebesar 78%.

***Loss***

Nilai dari *loss* pada *training* mengalami penurunan hingga iterasi terakhir. Nilai *training loss* tertinggi terjadi pada *epoch* pertama, yaitu 1,8 dan terendah pada *epoch* terakhir, yaitu 0,03. Nilai akurasi pada *validation loss* mengalami naik turun hingga *epoch* ke-5 dan perlahan mengalami penurunan nilai *loss* hingga *epoch* terakhir. Nilai *validation loss* tertinggi terdapat pada *epoch* ke-6 sebesar 14 dan untuk yang terendah terjadi pada iterasi ke-4, yaitu sebesar 1,4.

## 4.6 Pengaruh *Hyperparameters* terhadap Akurasi

Proses *training* pada jaringan diatur dengan mendefinisikan beberapa variabel yang disebut *hyperparameter*, seperti *learning rate*, jumlah *hidden layer*, jumlah *neuron*, fungsi aktivasi, jumlah *training epoch* dan lainnya (Atteia et al., 2022). *Hyperparameter* sangat memberikan dampak pada pelatihan serta kinerja

model. Oleh karena itu, menetapkan *hyperparameter* terbaik merupakan langkah yang penting dalam membuat model yang akan menghasilkan akurasi dan prediksi yang baik.

***Jumlah Training Epoch***

*Epoch* merupakan sebuah siklus algoritma *learning* dari seluruh data *training*. Satu *epoch* menandakan sebuah algoritma *machine learning* yang telah 'mempelajari' data *training* secara keseluruhan (Wibawa, 2017). Wibawa mengatakan bahwa nilai *epoch* yang sesuai tidak dapat diketahui dan hingga saat ini belum ada penelitian yang melakukan klaim mengenai jumlah *epoch* terbaik dalam melakukan proses pembelajaran pada data *training*. Pada penelitian ini akan diujikan beberapa nilai *epoch* agar dapat diketahui nilai akurasi *training* yang terbaik.

**Tabel 18.** Pengaruh Jumlah *Epoch* (Percobaan 1)

| ***Epoch*** | **Akurasi *Training*** | **Akurasi *Validation*** | **Akurasi *Testing*** | ***Loss Value*** | **Waktu** |
|---|---|---|---|---|---|
| 10 | 95% | 66% | 68% | 0,01 | 1 jam 14 menit |
| 20 | 99% | 77% | 73% | 0,02 | 1 jam 26 menit |
| 30 | 99% | 79% | 73% | 0,01 | 3 jam 28 menit |

**Tabel 19.** Pengaruh Jumlah Epoch (Percobaan 2)

| ***Epoch*** | **Akurasi *Training*** | **Akurasi *Validation*** | **Akurasi *Testing*** | ***Loss Value*** | **Waktu** |
|---|---|---|---|---|---|
| 10 | 98% | 62% | 77% | 0,048 | 1 jam 20 menit |
| 20 | 99% | 78% | 91% | 0,03 | 2 jam 46 menit |
| 30 | 99% | 70% | 77% | 0,01 | 3 jam 30 menit |

**Tabel 20.** Pengaruh Jumlah Epoch (Percobaan 3)

| ***Epoch*** | **Akurasi *Training*** | **Akurasi *Validation*** | **Akurasi *Testing*** | ***Loss Value*** | **Waktu** |
|---|---|---|---|---|---|
| 10 | 96% | 75% | 54% | 0,1 | 1 jam 16 menit |
| 20 | 99% | 76% | 82% | 0,02 | 2 jam 7 menit |
| 30 | 99% | 81% | 77% | 0,01 | 2 jam 25 menit |

Berdasarkan tabel di atas, dengan menggunakan nilai *learning rate* 0,001, dimensi citra 150x150, dan nilai *batch size* 32, didapatkan akurasi yang paling optimal adalah dengan jumlah *epoch* 20. Jumlah *epoch* 20 dan 30 memiliki nilai akurasi *training* dan nilai *loss* yang hampir sama tetapi terdapat perbedaan pada nilai akurasi *validation* dan *testing* sehingga dapat disimpulkan bahwa akurasi

terbaik didapatkan ketika model menerapkan jumlah *epoch* sebanyak 20 pada saat proses *training*.

***Nilai Batch Size***

Salah satu *hyperparameter* utama yang perlu disesuaikan sebelum proses *training* dilakukan adalah nilai *batch size*, yang mana *batch size* merupakan jumlah gambar yang akan digunakan dalam satu kali *epoch* atau iterasi. Beberapa peneliti telah melakukan penelitian mengenai pengaruh dari *batch size* terhadap performa model agar dapat menentukan nilai *batch size* terbaik.

Nilai *batch size* yang kecil dapat memproses lebih cepat dibandingkan dengan nilai *batch size* yang besar, namun nilai *batch size* yang besar dapat mencapai nilai optimum yang mana nilai ini tidak dapat dicapai jika menggunakan nilai *batch size* yang kecil (Kandel & Castelli, 2020). Kandel dan Castelli juga mengatakan bahwa nilai *batch size* yang kecil dapat memiliki efek regularisasi yang signifikan karena variansi yang tinggi, namun nilai yang kecil akan membutuhkan nilai *learning rate* yang kecil pula untuk menghindari terjadinya *overshooting* atau lonjakan pada nilai.

Pada penelitian ini akan dilakukan pengujian dengan beberapa nilai *batch size*, yaitu 10, 16, dan 32.

**Tabel 21.** Pengaruh Nilai *Batch Size*

| ***Batch Size*** | **Akurasi *Training*** | **Akurasi *Validation*** | **Akurasi *Testing*** | ***Loss Value*** | **Waktu** |
|---|---|---|---|---|---|
| 10 | 99% | 71% | 60% | 0,03 | 2 jam 30 menit |
| 16 | 99% | 76% | 63% | 0,03 | 1 jam 42 menit |
| 32 | 99% | 78% | 91% | 0,03 | 2 jam 46 menit |

Berdasarkan tabel di atas, dengan menggunakan nilai *learning rate* 0,001, dimensi citra 150x150, dan jumlah *epoch* 20, didapatkan akurasi yang paling optimal adalah dengan nilai *batch size* 32. Nilai *batch* size 16 dan 32 memiliki nilai akurasi *training, validation,* dan nilai *loss* yang hampir sama tetapi terdapat perbedaan pada nilai akurasi *testing* sehingga dapat disimpulkan bahwa akurasi terbaik didapatkan ketika model menerapkan nilai *batch size* sebesar 32 pada saat proses *training*.

## 4.7 Hasil Proses *Testing*

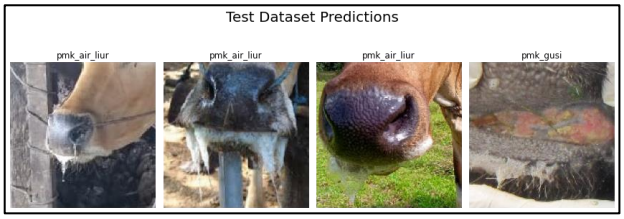
Setelah arsitektur model ditentukan, dilakukan proses *training,* hingga mendapatkan arsitektur dengan *hyperparameter* terbaik, kemudian dilanjutkan dengan pengujian model. Pengujian ini menggunakan data uji sebanyak 22 citra dengan jumlah setiap kelas berbeda-beda. Seperti yang telah dijelaskan sebelumnya, jumlah data uji pada tiap kelas ini didapatkan dari hasil pembagian

data *training*, *validation,* dan *testing* dengan skenario 70:20:10. Sebaran data uji pada tiap kelas ditnjukkan pada tabel 9.

**Tabel 22.** Jumlah Data Uji pada Tiap Kelas

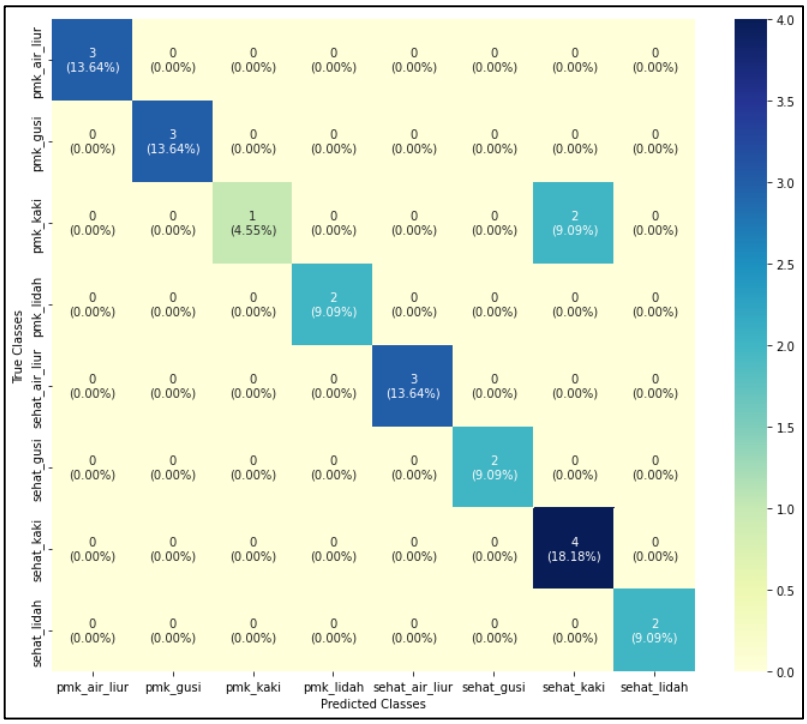
| No | Direktori | Jumlah Data |
|---|---|---|
| 1. | Dataset/pmk/air_liur | 3 |
| 2. | Dataset/pmk/gusi | 3 |
| 3. | Dataset/pmk/lidah | 2 |
| 4. | Dataset/pmk/kaki | 3 |
| 5. | Dataset/sehat/air_liur | 3 |
| 6. | Dataset/sehat/gusi | 2 |
| 7. | Dataset/sehat/lidah | 2 |
| 8. | Dataset/sehat/kaki | 4 |
| **Total** | | **22** |

Berikut merupakan sampel data pengujian yang digunakan:




**Gambar 39.** Sampel Data *Testing*

Dari hasil perhitungan data uji, didapatkan akurasi *test* sebesar 91% dan nilai *loss* 1,9. Dari pengujian data tersebut juga didapatkan *confusion matrix* sebagai berikut:




**Gambar 40.** *Confusion Matrix*

Dari hasil *confusion matrix* tersebut, didapatkan hasil perhitungan *accuracy, precision, recall,* dan *f1 score* sebagai berikut:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| pmk_air_liur | 1.00 | 1.00 | 1.00 | 3 |
| pmk_gusi | 1.00 | 1.00 | 1.00 | 3 |
| pmk_kaki | 1.00 | 0.33 | 0.50 | 3 |
| pmk_lidah | 1.00 | 1.00 | 1.00 | 2 |
| sehat_air_liur | 1.00 | 1.00 | 1.00 | 3 |
| sehat_gusi | 1.00 | 1.00 | 1.00 | 2 |
| sehat_kaki | 0.67 | 1.00 | 0.80 | 4 |
| sehat_lidah | 1.00 | 1.00 | 1.00 | 2 |
| accuracy | | | 0.91 | 22 |
| macro avg | 0.96 | 0.92 | 0.91 | 22 |
| weighted avg | 0.94 | 0.91 | 0.90 | 22 |

**Gambar 41.** *Precision, Recall,* dan *F1 Score*

Berdasarkan Gambar 40 dan Gambar 41, dapat disimpulkan bahwa didapatkan hasil prediksi model yang sangat baik pada proses *testing* dengan menggunakan sebanyak 22 data uji. Hasil akurasi yang didapatkan dari model dengan ukuran citra 150x150 piksel, jumlah *epoch* 20, *batch size* 32, dan *learning rate* 0,001 adalah sebesar 91%. Berikut contoh perhitungan untuk mendapatkan nilai *precision, recall,* dan *f1 score* pada kelas 'sehat_kaki':

$$Precision = \frac{TP}{TP + FP} = \frac{4}{4 + 2} = 0{,}67$$

$$Recall = \frac{TP}{TP + FN} = \frac{4}{4 + 0} = 1$$

$$F1\ Score = 2 \times \left(\frac{precision \times recall}{precision + recall}\right) = 2 \times \left(\frac{1 \times 0{,}67}{1 + 0{,}67}\right) = 0{,}80$$

Dari perhitungan di atas, dapat disimpulkan bahwa rasio prediksi benar positif dibandingkan dengan keseluruhan hasil yang diprediksi positif adalah 0,67. Selanjutnya rasio prediksi benar positif dibandingkan dengan keseluruhan data yang benar positif adalah 1. Selain itu, dapat diketahui bahwa dari 4 data *test* kelas 'sehat_kaki' terdapat 2 data yang diprediksi 'sehat_kaki' namun kenyataannya data tersebut bukan termasuk ke dalam kelas 'sehat_kaki' (*False Positive*).

Pada *classification report* (Gambar 41), seluruh kelas mendapatkan akurasi *precision, recall,* dan *f1-score* di atas 80% kecuali kelas 'pmk_kaki' dan 'sehat_kaki. Padahal total data gambar untuk kelas 'sehat_kaki' yang digunakan untuk *training* (setelah proses augmentasi) memiliki jumlah data terbanyak dibandingkan dengan data gambar yang dimiliki oleh kelas lain. Menurut penulis, hal ini disebabkan karena beberapa data gambar kaki sapi saat pengambilan data gambar dilakukan dalam keadaan tidak bersih (terdapat kotoran seperti tanah/debu yang menutupi sebagian luka) sehingga luka pada kaki sapi hanya sedikit terlihat dan sistem kesulitan mempelajari setiap piksel pada gambar yang

terdapat luka lepuh serta model kesulitan untuk mendeteksi apakah kaki sapi sehat atau tidak.

**Gambar 42.** Contoh Data Gambar Kaki Sapi yang Tidak Bersih

Dari hasil *testing* yang telah dilakukan, terdapat beberapa data yang terprediksi salah. Kesalahan prediksi ini terjadi pada data dengan label 'pmk_kaki' dan 'sehat_kaki'. Berikut data yang mengalami kesalahan prediksi:


**Gambar 43.** Data yang Mengalami Kesalahan Prediksi

## 4.8 Pembuatan REST API Model Deteksi PMK

Pada tahap ini penulis membuat rancangan *request* dan *response* yang akan menjelaskan aturan dalam proses permintaan sumber data dan hasil kembalian dari layanan. *Request* prediksi gambar dilakukan untuk meminta hasil probabilitas beberapa kelas terhadap gambar yang dikirimkan. Berikut merupakan rancangan *request* dan *response* dalam prediksi gambar:

**Tabel 23.** Rancangan *Request* dan *Response* Prediksi Gambar

| | |
|---|---|
| ***Method*** | POST |
| **URI *Scheme*** | http |
| **URI *Host*** | 127.0.0.1:3000 |
| ***Endpoint*** | /predict |
| ***Headers*** | - |
| ***Body*** | file |
| **Keterangan** | Pada *body*, dilakukan penyisipan data dengan key bernama *file* saat *request* dilakukan. Parameter *file* ini memiliki *value* gambar berformat 'jpg', 'jpeg','png' atau 'jfif' dan gambar ini akan diproses oleh REST API untuk mendapatkan hasil dari pendeteksian gambar. |

| | |
|---|---|
| ***Format Response*** | JSON |
| ***Response*** | <pre>{<br>  "prediction": [<br>  [<br>  "pmk_air_liur",<br>  "sehat_air_liur",<br>  "pmk_gusi"<br>  ],<br>  [<br>  71.46,<br>  28.54,<br>  0.0<br>  ]<br>  ]<br>}</pre> |

REST API dibangun dengan menggunakan *microframework* Python, yaitu Flask. Dalam *preprocessing* hingga menampilkan *output* probabilitas kelas dari hasil deteksi, digunakan modul *tensorflow*. Untuk *preprocessing* gambar, digunakan modul *tensorflow.keras.preprocessing.image* dan untuk penggunaan *trained model* dalam memprediksi gambar, digunakan fungsi *model.predict()* yang berasal dari modul *tf.keras.Model*. Berikut penjelasan kode program dan setiap fungsi yang digunakan pada program REST API :

```
model = tf.keras.models.load_model('Flask\PMK_Detection\modelPMK.h5')
classes = ['pmk_air_liur', 'pmk_gusi', 'pmk_kaki', 'pmk_lidah',
'sehat_air_liur', 'sehat_gusi', 'sehat_kaki', 'sehat_lidah']
```

**Gambar 44.** Pendefinisian Model dan Label Tiap Kelas

Pada REST API ini akan digunakan model CNN yang telah dibuat dan berformat h5. Selain itu, didefinisikan variabel '*classes*' yang akan digunakan untuk pelabelan setiap kelas.

```
def prepare_image(img):
    img = load_img(io.BytesIO(img), target_size = (150 , 150))
    img = img_to_array(img)
    img = img.reshape(1 , 150 ,150 ,3)
    img = img.astype('float32')
    img = img/255.0
    return img
```

**Gambar 45.** Fungsi *prepare_image*()

Pada Gambar 45, terdapat fungsi *prepare_image()* yang berfungsi untuk mempersiapkan gambar yang akan diprediksi oleh model. Fungsi ini menerima sebuah parameter gambar dan gambar tersebut akan melalui beberapa proses, seperti ukuran yang diubah menjadi 150x150 piksel, mengubah setiap piksel

gambar menjadi *array*, melakukan *reshape* dengan nilai *batch size* 1 dan jumlah *channel* 3. Setiap piksel diberi sebuah nilai bertipe *float* sesuai dengan derajat komponen nilai RGB.

```
def predict_result(img):
    result = model.predict(img)

    dict_result = {}
    for i in range(8):
        dict_result[result[0][i]] = classes[i]
    res = result[0]
    res.sort()
    res = res[::-1]
    prob = res[:3]

    prob_result = []
    class_result = []
    for i in range(3):
        prob_result.append((prob[i]*100).round(2))
        class_result.append(dict_result[prob[i]])
    return class_result , prob_result
```

**Gambar 46.** Fungsi *predict_result*()

Fungsi *predict_result*() berfungsi untuk memprediksi gambar yang telah dipersiapkan sebelumnya pada fungsi *prepare_image*(). Pada fungsi *predict_result*, gambar diprediksi menggunakan model CNN yang telah didefinisikan sebelumnya. Hasil prediksi tersebut kemudian diurutkan mulai dari probabilitas tertinggi hingga terendah sekaligus label kelas pada setiap nilai probabilitas. Hasil nilai-nilai probabilias beserta labelnya disimpan ke dalam variabel 'prob_result' dan 'class_result' bertipe *array*.

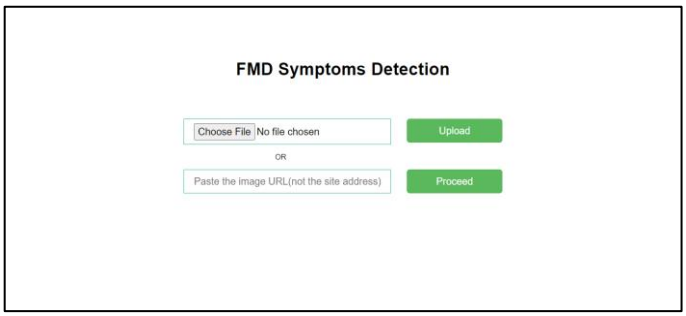
```
@app.route('/predict', methods=['POST'])
def infer_image():
    if 'file' not in request.files:
        return "Please try again. The Image doesn't exist"

    file = request.files.get('file')
    if not file:
        return

    img_bytes = file.read()
    img = prepare_image(img_bytes)
    return jsonify(prediction=predict_result(img))
```

**Gambar 47.** Fungsi *infer_image*()

Pada Gambar 47, terdapat fungsi *infer_image*() yang akan menerima *file* berupa gambar dari *client* yang melakukan *request* HTTP ke *server* dengan *method* POST. Setelah *file* berhasil dibaca oleh *server*, maka fungsi *prepare_image* akan dipanggil dan selanjutnya dilakukan prediksi pada *file* tersebut. Fungsi ini akan memberikan *output* berupa hasil prediksi tiga kelas dengan probabilitas tertinggi yang disimpan dalam format JSON.

## 4.9 Deteksi dengan *Website*

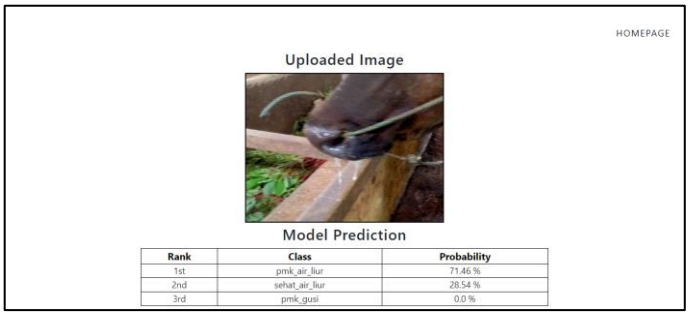
Pada tahap ini dilakukan pengujian dalam pendeteksian gambar dengan aplikasi berbasis *website* yang telah terintegrasi dengan REST API Model Deteksi PMK. *Website* ini dikembangkan dengan menggunakan sebuah *microframework* Python yang bernama Flask. Dalam penelitian ini, aplikasi dijalankan pada sebuah url, yaitu http://127.0.0.1:5000. Berikut merupakan halaman utama aplikasi:




**Gambar 48.** Halaman Utama Aplikasi

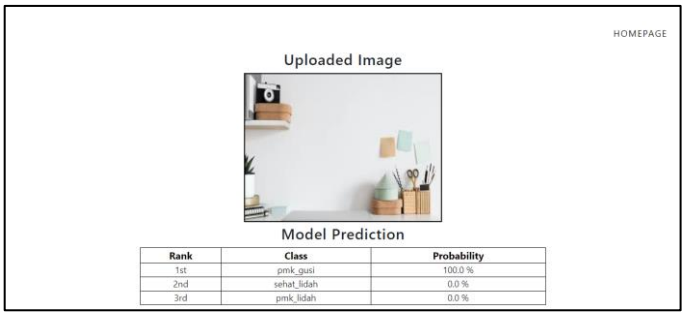
Pada halaman utama, terdapat dua jenis metode dalam *upload* gambar. *User* dapat mengirimkan gambar dengan *upload* dari penyimpanan lokal dan dapat mengisi *form input link* gambar dari internet. Setelah *file* gambar ataupun *link* dimasukkan, *user* dapat menekan tombol '*upload*' atau '*proceed*'. Setelah itu, sistem akan mengirimkan *file* gambar tersebut ke server REST API untuk diproses dalam pendeteksian PMK.

Setelah gambar berhasil diproses oleh REST API, maka akan dihasilkan *output* berupa JSON yang akan dibaca oleh aplikasi sebagai *client*. *Output* yang dihasilkan berupa tiga kelas dengan probabilitas tertinggi dan akan ditampilkan pada halaman yang baru seperti berikut:




**Gambar 49.** Halaman Hasil Prediksi Gambar

Ketika dilakukan pengujian pada gambar objek yang tidak termasuk ke dalam kelas manapun, akan menghasilkan akurasi yang sempurna dan label yang memiliki akurasi sempurna tersebut bukan merupakan kelas objek yang benar seperti berikut:




**Gambar 50.** Hasil Prediksi pada *Random Object*

Hal ini dapat terjadi karena model jaringan saraf modern merupakan model prediktif yang sangat kuat namun tidak mampu mengetahui kapan prediksi dari model tersebut melakukan kesalahan (DeVries & Taylor, 2018). Untuk mengatasi permasalahan ini, dapat digunakan teknik *out-of-distribution detection*. Beberapa penelitian telah dilakukan dalam pengembangan teknik *out-of-distribution* ini (DeVries & Taylor, 2018; Lee et al., 2017). Teknik ini akan membantu model untuk mengatasi permasalahan dalam mengenali objek yang bukan merupakan bagian dari kelas manapun yang telah ditetapkan sebelumnya.

# V. KESIMPULAN DAN SARAN

## 5.1 Kesimpulan

Berdasarkan penelitian yang telah dikerjakan, maka dapat ditarik beberapa kesimpulan sebagai berikut:

1. Dari hasil proses konfigurasi *hyperparameter*, didapatkan akurasi terbaik dengan *epoch* berjumlah 20 dan *batch size* 32. Dengan skenario perbandingan *dataset train* sebesar 70%, *dataset validation* 20%, dan *dataset test* sebesar 10%, digunakan *Input* citra yang berdimensi 150x150 piksel dan ukuran *learning rate* 0,001.
2. Hasil dari akurasi *testing* yang didapatkan dengan arsitektur yang telah dibuat adalah sebesar 91% yang mana dari 22 data *test* yang diberikan, 20 di antaranya terprediksi dengan benar dan prediksi tersebut sesuai dengan kelas yang sebenarnya. Dari hasil *recall, precision,* dan *f1-score* pada kelas 'pmk_kaki' dan 'sehat_kaki' menghasilkan akurasi yang lebih rendah dibandingkan kelas lainnya dan ini disebabkan karena beberapa data gambar kaki sapi dalam keadaan tidak bersih (terdapat kotoran seperti tanah/debu yang menutupi sebagian luka) sehingga sistem kesulitan mempelajari setiap piksel pada gambar yang terdapat luka lepuh.
3. Hasil dari penelitian ini menunjukkan bahwa gejala awal penyakit mulut dan kuku pada sapi dapat dideteksi menggunakan model yang dibangun dengan metode CNN dan model tersebut berhasil dijalankan pada aplikasi berbasis web yang terintegrasi REST API sehingga dapat mendeteksi gejala awal PMK pada sapi dengan menggunakan *framework* FLASK.
4. Model CNN yang telah dibuat untuk deteksi gejala awal PMK pada sapi ini masih belum menghasilkan akurasi yang sangat baik pada *validation accuracy* sehingga diperlukan *dataset* yang lebih besar, baik data gejala PMK maupun sehat untuk meningkatkan evaluasi performa model dan meningkatkan keakuratan dalam pendeteksian.

## 5.2 Saran

Berdasarkan penelitian yang telah dilakukan, beberapa saran yang penulis berikan sebagai upaya perbaikan untuk penelitian selanjutnya adalah sebagai berikut:

1. Menambahkan beberapa *hyperparameter* sebagai pembanding untuk memperoleh arsitektur CNN yang menghasilkan akurasi lebih baik lagi.
2. Sektor peternakan dan kesehatan hewan di Indonesia perlu didorong untuk selalu melakukan kegiatan dokumentasi terhadap gejala awal pada

hewan yang terkena PMK. Hal ini diperlukan untuk mempermudah penulis dalam memperoleh dan memperbanyak jumlah *dataset* yang dapat dipelajari oleh model sehingga pendeteksian dapat semakin akurat.

3. Pada penelitian selanjutnya perlu perubahan data gambar pada kelas ‘pmk_kaki’ dan ‘sehat_kaki’. Data gambar kaki pada kedua kelas tersebut sebaiknya bersih dan luka lepuh pada kaki sapi terlihat dengan jelas.

**DAFTAR PUSTAKA**


Andrej Karpathy, Justin Johnson, & Fei-Fei Li. (2015). *Convolutional Neural Networks for Visual Recognition*. Lecture Notes to CS231n.

Atteia, G., Abdel Samee, N., El-Kenawy, E.-S. M., & Ibrahim, A. (2022). CNN-Hyperparameter Optimization for Diabetic Maculopathy Diagnosis in Optical Coherence Tomography and Fundus Retinography. *Mathematics*, *10*(18), 3274.

Balamurugan, M. (2020). *How to Deploy Machine Learning Models as a REST API*. https://medium.com/@balamurugan.glakes/how-to-deploy-machine-learning-models-as-a-rest-api-1d59a8c496a3

Bera, S., & Shrivastava, V. K. (2020). Analysis of various optimizers on deep convolutional neural network model in the application of hyperspectral remote sensing image classification. *International Journal of Remote Sensing*, *41*(7), 2664–2683. https://doi.org/10.1080/01431161.2019.1694725

Burkov, A. (2019). *The hundred-page machine learning book* (Vol. 1). Andriy Burkov Quebec City, QC, Canada.

DeVries, T., & Taylor, G. W. (2018). Learning confidence for out-of-distribution detection in neural networks. *ArXiv Preprint ArXiv:1802.04865*.

Direktorat Kesehatan Hewan. (2022). *Kesiagaan Darurat Veteriner Indonesia Seri Penyakit Mulut dan Kuku (Kiat Vetindo) PMK* (3.1). Direktorat Kesehatan Hewan.

*Flask's Documentation*. (2022). http://flask.pocoo.org/

Géron, A. (2019). *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. " O'Reilly Media, Inc."

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press.

Grubman, M. J., & Baxt, B. (2004). Foot and Mouth Disease. *Clinical Microbiology Reviews*, *17*(2), 465–493. https://doi.org/10.1128/CMR.17.2.465-493.2004

Haenlein, M., & Kaplan, A. (2019). A Brief History of Artificial Intelligence: On the Past, Present, and Future of Artificial Intelligence. *California Management Review*, *61*(4), 5–14. https://doi.org/10.1177/0008125619864925

Hamjaya Putra Hamdu. (2019). *Laporan Surveilans Eksotik Penyakit Mulut dan Kuku (PMK) dan Bovine Spongiform Encephalopathy (BSE)*. Balai Besar Veteriner Maros.

Heaton, J. (2015). *Artificial Intelligence for Humans: Deep learning and neural networks*. Heaton Research, Incorporated. https://books.google.co.id/books?id=q9mijgEACAAJ

Hidayat, A., Darusalam, U., & Irmawati, I. (2019). Detection of Disease on Corn Plants Using Convolutional Neural Network Methods. *Jurnal Ilmu Komputer Dan Informasi*, *12*(1), 51–56.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *ArXiv Preprint ArXiv:1207.0580*.

Ilahiyah, S., & Nilogiri, A. (2018). Implementasi Deep Learning Pada Identifikasi Jenis Tumbuhan Berdasarkan Citra Daun Menggunakan Convolutional Neural Network. *JUSTINDO (Jurnal Sistem & Teknologi Informasi Indonesia)*, *3*(2), 50.

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *International Conference on Machine Learning*, 448–456.

Irfan, D., Rosnelly, R., Wahyuni, M., Samudra, J. T., & Rangga, A. (2022). PERBANDINGAN OPTIMASI SGD, ADADELTA, DAN ADAM DALAM KLASIFIKASI HYDRANGEA MENGGUNAKAN CNN. *JOURNAL OF SCIENCE AND SOCIAL RESEARCH*, *5*(2), 244–253.

Irsyad, R. (2018). *Penggunaan Python Web Framework Flask Untuk Pemula*.

Jin, J., Dundar, A., & Culurciello, E. (2014). Flattened Convolutional Neural Networks for Feedforward Acceleration. *ArXiv Preprint ArXiv:1412.5474*, 3.

Kandel, I., & Castelli, M. (2020). The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset. *ICT Express*, *6*(4), 312–315. https://doi.org/https://doi.org/10.1016/j.icte.2020.04.010

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *ArXiv Preprint ArXiv:1412.6980*.

Kitching, R. P. (2002). Clinical variation in foot and mouth disease: cattle. *Revue Scientifique et Technique-Office International Des Epizooties*, *21*(3), 499–502.

Kuhamba, T. K. (2020). *A Deep Learning Based Approach For Foot And Mouth Disease Detection*. https://hdl.handle.net/10646/4211

Lee, K., Lee, H., Lee, K., & Shin, J. (2017). Training confidence-calibrated classifiers for detecting out-of-distribution samples. *ArXiv Preprint ArXiv:1711.09325*.

Liu, L., Ouyang, W., Wang, X., Fieguth, P., Chen, J., Liu, X., & Pietikäinen, M. (2020). Deep Learning for Generic Object Detection: A Survey. *International Journal of Computer Vision*, *128*(2), 261–318. https://doi.org/10.1007/s11263-019-01247-4

Masse, M. (2011). *REST API design rulebook: designing consistent RESTful web service interfaces*. " O'Reilly Media, Inc."

Mueller, J. P., & Massaron, L. (2021a). *Artificial intelligence for dummies*. John Wiley & Sons.

Mueller, J. P., & Massaron, L. (2021b). *Machine learning for dummies*. John Wiley & Sons.

Nwankpa, C., Ijomah, W., Gachagan, A., & Marshall, S. (2018). Activation functions: Comparison of trends in practice and research for deep learning. *ArXiv Preprint ArXiv:1811.03378*.

Petrou, M. M. P., & Petrou, C. (2010). *Image processing: the fundamentals*. John Wiley & Sons.

Prastika, I. W., & Zuliarso, E. (2021). Deteksi Penyakit Kulit Wajah Menggunakan Tensorflow dengan Metode Convolutional Neural Network. *Jurnal Manajemen Informatika & Sistem Informasi*, *4*(2). http://e-journal.stmiklombok.ac.id/index.php/misi

Purnomo, A., & Tjandrasa, H. (2021). IMPROVED DEEP LEARNING ARCHITECTURE WITH BATCH NORMALIZATION FOR EEG SIGNAL PROCESSING. *Jurnal Ilmiah Teknologi Informasi*, *19*(1).

Pusat Data dan Analisa Tempo. (2020). *Indonesia dan Penanganan Penyakit Kuku dan Mulut*. Tempo Publishing.

Putri, T. A., Suratno, T., & Khaira, U. (2022). Identification of Incung Characters (Kerinci) to Latin Characters Using Convolutional Neural Network. *IJCCS (Indonesian Journal of Computing and Cybernetics Systems)*, *16*(2).

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 779–788.

Richardson, L., Amundsen, M., & Ruby, S. (2013). *RESTful Web APIs: Services for a Changing World*. " O'Reilly Media, Inc."

Santurkar, S., Tsipras, D., Ilyas, A., & Madry, A. (2018). How Does Batch Normalization Help Optimization? In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems* (Vol. 31). Curran Associates, Inc. https://proceedings.neurips.cc/paper/2018/file/905056c1ac1dad141560467e0a99e1cf-Paper.pdf

Saputra, R. A., Wasyianti, S., Supriyatna, A., & Saefudin, D. F. (2021). Penerapan Algoritma Convolutional Neural Network Dan Arsitektur MobileNet Pada Aplikasi Deteksi Penyakit Daun Padi. *JURNAL SWABUMI*, *9*(2).

Schapire, R. E., & Freund, Y. (2012). *Foundations of Machine Learning*. 1.

Shalev-Shwartz, S., & Ben-David, S. (2014). *Understanding machine learning: From theory to algorithms*. Cambridge university press.

Shanmugamani, R., & Moore, S. M. (2018). *Deep Learning for Computer Vision: Expert Techniques to Train Advanced Neural Networks Using TensorFlow and Keras*. Packt Publishing. https://books.google.co.id/books?id=dgdOswEACAAJ

Siaga PMK. (2022). *Sebaran Kasus PMK*. siagapmk.id

Sonka, M., Hlavac, V., & Boyle, R. (2014). *Image processing, analysis, and machine vision*. Cengage Learning.

Stathakis, D. (2009). How many hidden layers and nodes? *International Journal of Remote Sensing*, *30*(8), 2133–2147. https://doi.org/10.1080/01431160802549278

Subramanian, H., & Raj, P. (2019). *Hands-On RESTful API Design Patterns and Best Practices: Design, develop, and deploy highly adaptable, scalable, and secure RESTful web APIs*. Packt Publishing Ltd.

Sutawi. (2022, May 24). *Bioterorisme Penyakit Mulut dan Kuku (PMK)*. Https://Www.Agropustaka.Id/Pemikiran/Bioterorisme-Penyakit-Mulut-Dan-Kuku-Pmk/.

Visa, S., Ramsay, B., Ralescu, A. L., & van der Knaap, E. (2011). Confusion matrix-based feature selection. *MAICS*, *710*(1), 120–127.

Weni, I., Utomo, P. E. P., Hutabarat, B. F., & Alfalah, M. (2021). Detection of Cataract Based on Image Features Using Convolutional Neural Networks. *Indonesian Journal of Computing and Cybernetics Systems)*, *15*(1), 75–86.

Wibawa, M. S. (2017). Pengaruh Fungsi Aktivasi, Optimisasi dan Jumlah Epoch Terhadap Performa Jaringan Saraf Tiruan. *Jurnal Sistem Dan Informatika (JSI)*, *11*(2), 167–174.

Wikarta, A., Pramono, A. S., & Ariatedja, J. B. (2020). Analisa Bermacam Optimizer Pada Convolutional Neural Network Untuk Deteksi Pemakaian Masker Pengemudi Kendaraan. *Seminar Nasional Informatika (SEMNASIF)*, *1*(1), 69–72.

Yu, L., Li, B., & Jiao, B. (2019). Research and Implementation of CNN Based on TensorFlow. *IOP Conference Series: Materials Science and Engineering*, *490*, 042022. https://doi.org/10.1088/1757-899x/490/4/042022

Zhang, X., Lee, V. C. S., Rong, J., Liu, F., & Kong, H. (2022). Multi-channel Convolutional Neural Network Architectures for Thyroid Cancer Detection. *PLoS ONE*, *17*(1 January). https://doi.org/10.1371/journal.pone.0262128

Zou, Z., Shi, Z., Guo, Y., & Ye, J. (2019). Object detection in 20 years: A survey. *ArXiv Preprint ArXiv:1905.05055*.

**LAMPIRAN**

**Lampiran 1.** Pengambilan Data Sapi Terjangkit PMK di Dinas Pertanian dan Ketahanan Pangan Kota Jambi

**Lampiran 2.** Pengambilan Data Sapi Sehat di Fakultas Peternakan Universitas Jambi

**Lampiran 3.** *Script Scrapping* Data Gambar Sapi Terjangkit PMK dan Sehat di Google

```
# Import packages
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.chrome.service import Service
import requests
import hashlib
import os
import io
import time
from PIL import Image

# Define the path to chrome driver
s = Service('{chrome driver path}')
wd = webdriver.Chrome(service=s)
# searches images from google.com
wd.get('https://google.com')

class GoogleScraper():
    def __init__(self, webdriver:webdriver, max_num_of_images:int):
        self.wd = webdriver
        self.max_num_of_images = max_num_of_images
    def _scroll_to_the_end(self):
        wd.execute_script("window.scrollTo(0, document.body.scrollHeight);")
        time.sleep(1)
    def _build_query(self, query:str):
        return
f"https://www.google.com/search?safe=off&site=&tbm=isch&source=hp&q={query}&oq={
query}&gs_l=img"

    def _get_info(self, query: str):
        image_urls = set()
        wd.get(self._build_query(query))
        self._scroll_to_the_end()

        # img.Q4LuWd is the google tumbnail selector
        thumbnails = self.wd.find_elements(By.CSS_SELECTOR, "img.Q4LuWd")
        print(f"Found {len(thumbnails)} images...")
        print(f"Getting the links...")

        for img in thumbnails[0:self.max_num_of_images]:
            # click every thumbnail so we can get the full image.
            try:
                img.click()
            except Exception:
                print('ERROR: Cannot click on the image.')
                continue

            images = wd.find_elements(By.CSS_SELECTOR, "img.n3VNCb")
            time.sleep(0.3)
            for image in images:
                if image.get_attribute('src') and 'http' in
image.get_attribute('src'):
                    image_urls.add(image.get_attribute('src'))

        return image_urls
```

```
def download_image(self, folder_path:str, url:str):
        try:
            image_content = requests.get(url).content

        except Exception as e:
            print(f"ERROR: Could not download {url} - {e}")

        try:
            image_file = io.BytesIO(image_content)
            image = Image.open(image_file).convert('RGB')
            file =
os.path.join(folder_path,hashlib.sha1(image_content).hexdigest()[:10] + '.jpg')

            with open(file, 'wb') as f:
                image.save(f, "JPEG", quality=85)
            print(f"SUCCESS: saved {url} - as {file}")

        except Exception as e:
            print(f"ERROR: Could not save {url} - {e}")

    def scrape_images(self, query:str, folder_path= 'D:/FILE HELVIANI/FILE
KULIAH/SEMINAR/Code/Web-Image-Scraper-main/pmk'):
        folder = os.path.join(folder_path,'_'.join(query.lower().split(' ')))

        if not os.path.exists(folder):
            os.makedirs(folder)

        image_info = self._get_info(query)
        print(f"Downloading images...")

        for image in image_info:
            self.download_image(folder, image)

# Run scrape command
gs = GoogleScraper(wd, 200)
# image's keyword
gs.scrape_images('foot and mouth disease cow lesion')
```

**Lampiran 4.** *Script* Pembuatan Model CNN

```
# Import Libraries
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import os
import warnings
warnings.filterwarnings('ignore')
from sklearn.metrics import confusion_matrix, classification_report
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Activation, BatchNormalization, Conv2D,
Dense, Dropout, Flatten, MaxPooling2D
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.losses import CategoricalCrossentropy
from tensorflow.keras.regularizers import l2
from tensorflow.keras.callbacks import ReduceLROnPlateau
from keras.preprocessing.image import ImageDataGenerator
from skimage import io
from PIL import Image
import random
from shutil import copyfile

# Get Dataset
os.environ['KAGGLE_USERNAME'] = "helvianizebua"
os.environ['KAGGLE_KEY'] = "426c331c302aae93f07f4afe8abc0220"
!kaggle datasets download -d helvianizebua/roiselected

# Unzip
from zipfile import ZipFile
file_name = "/content/roiselected.zip"
with ZipFile(file_name, 'r') as zp:
  zp.extractall(path='/content/')
  print('done')

# directories
dir_list = ['/content/Dataset/pmk/', '/content/Dataset/sehat/']
cat = ['air_liur', 'gusi', 'kaki', 'lidah']

base_dir = '/content/dataset'
train_dir = '/content/dataset/train/'
val_dir = '/content/dataset/validation/'
test_dir = '/content/dataset/test/'

train_aug = '/content/augmented/train/'
val_aug = '/content/augmented/validation/'
test_aug = '/content/augmented/test/'

classes =  [
    'pmk_air_liur', 'pmk_gusi', 'pmk_kaki', 'pmk_lidah',
    'sehat_air_liur', 'sehat_gusi', 'sehat_kaki', 'sehat_lidah']

# Create new folders
os.mkdir(base_dir)
os.mkdir(train_dir)
os.mkdir(val_dir)
os.mkdir(test_dir)

for i in classes:
  os.mkdir(os.path.join(train_dir,i))
for j in classes:
  os.mkdir(os.path.join(val_dir,j))
for k in classes:
  os.mkdir(os.path.join(test_dir,k))
```

```
# Split Dataset
def split_data(source, training, validation, test, split_size) :
  files = []
  for filename in os.listdir(source):
    file = source + filename
    if os.path.getsize(file) > 0:
      files.append(filename)
    else:
      print(filename + " is zero length, so ignoring.")


  training_length = int(len(files) * split_size) # data training 70%
  val_length = int(len(files)*((1-split_size)*(2/3))) # data validasi 20%

  shuffled_set = random.sample(files, len(files))

  training_set = shuffled_set[0: training_length]
  val_set = shuffled_set[training_length: training_length+val_length]
  test_set = shuffled_set[training_length+val_length:]


  for filename in training_set:
    this_file = source + filename
    destination = training + filename
    copyfile(this_file, destination)

  for filename in val_set:
    this_file = source + filename
    destination = validation + filename
    copyfile(this_file, destination)

  for filename in test_set:
    this_file = source + filename
    destination = test + filename
    copyfile(this_file, destination)

split_size = .7
x = 0
y = 0
for i in classes:
  if x == len(cat):
    x = 0
    y += 1
  split_data(dir_list[y]+cat[x]+'/', train_dir+i+'/', val_dir+i+'/',
test_dir+i+'/', split_size)
  x += 1

def count_data(input_dir, title):
  print('Data '+ title)
  for j in classes:
    print(' {} : '.format(j)+str(len(os.listdir(input_dir+'/'+j+'/'))))
  print()
```

```
# Image Augmentation
IMG_WIDTH = 150
IMG_HEIGHT = 150
datagen = ImageDataGenerator(
        rotation_range=30,
        width_shift_range=0.2,
        height_shift_range=0.2,
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=True,
        fill_mode='constant')

def aug_img(input_dir, output_dir, nums):
  itr_num = nums
  itr = 0
  for each_data in classes:
    dataset = []

    image_directory = input_dir+each_data+'/'
    dataset = []

    my_images = os.listdir(image_directory)
    for i, image_name in enumerate(my_images):
        if (image_name.split('.')[1] == 'jpg' or 'jpeg'):
            image = io.imread(image_directory + image_name)
            image = Image.fromarray(image, 'RGB')
            image = image.resize((IMG_WIDTH,IMG_HEIGHT))
            dataset.append(np.array(image))

    x = np.array(dataset)
    aug_dir = output_dir+each_data
    os.makedirs(aug_dir)
    i = 1
    for batch in datagen.flow(x, batch_size=16,
                              save_to_dir=aug_dir,
                              save_prefix='aug'+image_name,
                              save_format='png'):
        i += 1
        if i > itr_num[itr]:
            break

    itr +=1
    print('images are: '.format(i)+str(len(os.listdir(aug_dir))))
  print()

num_train = [45, 35, 37, 43, 40, 64, 47, 40]
num_val = [15, 18, 18, 27, 15, 27, 15, 27, 18, 18]
num_test = [5, 10, 10, 10, 5, 10, 5, 10]

aug_img(train_dir, train_aug, num_train)
aug_img(val_dir, val_aug, num_val)
aug_img(test_dir, test_aug, num_test)

#  Visualizes Training & Validation
def bar_vis(input_dir, title) :
  nimgs = {}
  for i in classes:
    nimages = len(os.listdir(input_dir+i+'/'))
    nimgs[i] = nimages

  plt.figure(figsize=(15, 6))
  plt.bar(range(len(nimgs)), list(nimgs.values()), align='center')
  plt.xticks(range(len(nimgs)), list(nimgs.keys()))
  plt.title('Data '+title)
  plt.show()
bar_vis(train_aug,'Training')
bar_vis(val_aug, 'Validation')

def count_data(input_dir, title):
  print('Data '+ title)
  for j in classes:
    print(' {} : '.format(j)+str(len(os.listdir(input_dir+j+'/'))))
  print()
```

```
# Processing Data
BATCH_SIZE = 32
train_datagen = ImageDataGenerator(rescale=1. /255)
train_generator = train_datagen.flow_from_directory(train_aug,
                                                    target_size=(IMG_WIDTH,
IMG_HEIGHT),
                                                    batch_size=BATCH_SIZE,
                                                    class_mode='categorical',
                                                    shuffle=True)

validation_datagen = ImageDataGenerator(rescale=1. /255)
validation_generator = validation_datagen.flow_from_directory(val_aug,
                                                              target_size=(IMG_WIDTH,
IMG_HEIGHT),
                                                              batch_size=BATCH_SIZE,
                                                              class_mode='categorical',
                                                              shuffle=True)

# Labelling
labels = {value: key for key, value in train_generator.class_indices.items()}

print(" Pemetaan label pada tiap kelas : \n")
for key, value in labels.items():
    print(f"{key} : {value}")

# Plotting Training Data Sampel
fig, ax = plt.subplots(nrows=2, ncols=5, figsize=(15, 12))
idx = 0

for i in range(2):
    for j in range(5):
        label = labels[np.argmax(train_generator[0][1][idx])]
        ax[i, j].set_title(f"{label}")
        ax[i, j].imshow(train_generator[0][0][idx][:, :, :])
        ax[i, j].axis("off")
        idx += 1

plt.tight_layout()
plt.suptitle("Data Training", fontsize=21)
plt.show()

# Create CNN Model
def create_model():
    model = Sequential([
        Conv2D(filters=128, kernel_size=(5, 5), padding='valid',
input_shape=(IMG_WIDTH, IMG_HEIGHT, 3)),
        Activation('relu'),
        MaxPooling2D(pool_size=(2, 2)),
        BatchNormalization(),

        Conv2D(filters=64, kernel_size=(3, 3), padding='valid',
kernel_regularizer=l2(0.00005)),
        Activation('relu'),
        MaxPooling2D(pool_size=(2, 2)),
        BatchNormalization(),

        Conv2D(filters=32, kernel_size=(3, 3), padding='valid',
kernel_regularizer=l2(0.00005)),
        Activation('relu'),
        MaxPooling2D(pool_size=(2, 2)),
        BatchNormalization(),

        Flatten(),

        Dense(units=256, activation='relu'),
        Dropout(0.5),
        Dense(units=8, activation='softmax')
    ])

    return model

cnn_model = create_model()
print(cnn_model.summary())
```

```
# Callback ReduceLROnPlateau
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=np.sqrt(0.1), patience=5)
# Add Adam Optimizer
optimizer = Adam(learning_rate=0.001)

cnn_model.compile(optimizer=optimizer, loss=CategoricalCrossentropy(),
metrics=['accuracy'])

# Training
history = cnn_model.fit(train_generator, epochs=20,
validation_data=validation_generator,
                        verbose=2,
                        callbacks=[reduce_lr])
cnn_model.save('modelPMK.h5')
cnn_model.save('modelPMK.hdf5')

# Convert the model.
converter = tf.lite.TFLiteConverter.from_keras_model(cnn_model)
tflite_model = converter.convert()

# Save the model.
with open('model.tflite', 'wb') as f:
  f.write(tflite_model)

# Plot Accuracy
train_accuracy = history.history['accuracy']
val_accuracy = history.history['val_accuracy']

train_loss = history.history['loss']
val_loss = history.history['val_loss']

learning_rate = history.history['lr']

fig, ax = plt.subplots(nrows=3, ncols=1, figsize=(12, 10))
ax[0].set_title('Training Accuracy vs. Epochs')
ax[0].plot(train_accuracy, 'o-', label='Train Accuracy')
ax[0].plot(val_accuracy, 'o-', label='Validation Accuracy')
ax[0].set_xlabel('Epochs')
ax[0].set_ylabel('Accuracy')
ax[0].legend(loc='best')

ax[1].set_title('Training/Validation Loss vs. Epochs')
ax[1].plot(train_loss, 'o-', label='Train Loss')
ax[1].plot(val_loss, 'o-', label='Validation Loss')
ax[1].set_xlabel('Epochs')
ax[1].set_ylabel('Loss')
ax[1].legend(loc='best')

ax[2].set_title('Learning Rate vs. Epochs')
ax[2].plot(learning_rate, 'o-', label='Learning Rate')
ax[2].set_xlabel('Epochs')
ax[2].set_ylabel('Loss')
ax[2].legend(loc='best')

plt.tight_layout()
plt.show()
```

```
# Confusion Matrix
y_pred = np.argmax(predictions, axis=1)
y_true = test_generator.classes

cf_mtx = confusion_matrix(y_true, y_pred)

group_counts = ["{0:0.0f}".format(value) for value in cf_mtx.flatten()]
group_percentages = ["{0:.2%}".format(value) for value in
cf_mtx.flatten()/np.sum(cf_mtx)]
box_labels = [f"{v1}\n({v2})" for v1, v2 in zip(group_counts, group_percentages)]
box_labels = np.asarray(box_labels).reshape(8, 8)

plt.figure(figsize = (12, 10))
sns.heatmap(cf_mtx, xticklabels=labels.values(), yticklabels=labels.values(),
            cmap="YlGnBu", fmt="", annot=box_labels)
plt.xlabel('Predicted Classes')
plt.ylabel('True Classes')
plt.show()

print(classification_report(y_true, y_pred, target_names=labels.values()))
```

**Lampiran 5.** *Script REST API*

```
import io
import os
import tensorflow as tf
from flask import Flask, jsonify, request
from tensorflow.keras.preprocessing.image import load_img , img_to_array

model = tf.keras.models.load_model('PMK_Detection\90%modelPMK.h5')
classes = ['pmk_air_liur', 'pmk_gusi', 'pmk_kaki', 'pmk_lidah', 'sehat_air_liur',
'sehat_gusi', 'sehat_kaki', 'sehat_lidah']

def prepare_image(img):
    img = load_img(io.BytesIO(img), target_size = (150 , 150))
    img = img_to_array(img)
    img = img.reshape(1 , 150 ,150 ,3)

    img = img.astype('float32')
    img = img/255.0
    return img

def predict_result(img):
    result = model.predict(img)

    dict_result = {}
    for i in range(8):
        dict_result[result[0][i]] = classes[i]

    res = result[0]
    res.sort()
    res = res[::-1]
    prob = res[:3]

    prob_result = []
    class_result = []
    for i in range(3):
        prob_result.append((prob[i]*100).round(2))
        class_result.append(dict_result[prob[i]])

    return class_result , prob_result

app = Flask(__name__)

@app.route('/predict', methods=['POST'])
def infer_image():
    if 'file' not in request.files:
        return "Please try again. The Image doesn't exist"

    file = request.files.get('file')

    if not file:
        return
    i = 0
    img_bytes = file.read()
    img = prepare_image(img_bytes)

    # save predicted file
    img_result = predict_result(img)
    best_class = img_result[0][0]
    target_img = 'PMK_Detection\\images\\'+ best_class
    file.save(os.path.join(target_img , str(i)+file.filename))

    return jsonify(prediction=predict_result(img))

@app.route('/', methods=['GET'])
def index():
    return 'Machine Learning Inference'

if __name__ == '__main__':
    app.run(debug=True, host='0.0.0.0', port=3000)
```

**Lampiran 6.** *Script* Aplikasi

```
import os
import uuid
import flask
import urllib
import tensorflow
from PIL import Image
from tensorflow.keras.models import load_model
from flask import Flask , render_template  , request , send_file
from tensorflow.keras.preprocessing.image import load_img , img_to_array
import requests

app = Flask(__name__)
BASE_DIR = os.path.dirname(os.path.abspath(__file__))
model = load_model(os.path.join(BASE_DIR , 'modelPMK.h5'))
apiurl = 'http://127.0.0.1:3000/predict'

ALLOWED_EXT = set(['jpg' , 'jpeg' , 'png' , 'jfif'])
def allowed_file(filename):
    return '.' in filename and \
           filename.rsplit('.', 1)[1] in ALLOWED_EXT

@app.route('/')
def home():
        return render_template("index.html")

@app.route('/success' , methods = ['GET' , 'POST'])
def success():
    error = ''
    target_img = os.path.join(os.getcwd() , 'PMK_Detection\\static\\images')
    if request.method == 'POST':
        if(request.form):
            link = request.form.get('link')
            try :
                resource = urllib.request.urlopen(link)
                unique_filename = str(uuid.uuid4())
                filename = unique_filename+".jpg"
                img_path = os.path.join(target_img , filename)
                output = open(img_path , "wb")
                output.write(resource.read())
                output.close()
                img = filename

                files = {'file': open(img_path, 'rb')}
                response = requests.post(apiurl, files=files)
                prediction = response.json()

            except Exception as e :
                print(str(e))
                error = 'This image from this site is not accesible or
inappropriate input'

            if(len(error) == 0):
                return  render_template('success.html' , img  = img , predictions
= prediction)
            else:
                return render_template('index.html' , error = error)
```

```
        elif (request.files):
            file = request.files['file']
            if file and allowed_file(file.filename):
                file.save(os.path.join(target_img , file.filename))
                img_path = os.path.join(target_img , file.filename)
                img = file.filename
                files = {'file': open(img_path, 'rb')}
                response = requests.post(apiurl, files=files)
                prediction = response.json()
            else:
                error = "Please upload images of jpg , jpeg and png extension
only"

            if(len(error) == 0):
                return  render_template('success.html' , img  = img, predictions
= prediction)
            else:
                return render_template('index.html' , error = error)

    else:
        return render_template('index.html')

if __name__ == "__main__":
    app.run(debug=True, host='0.0.0.0')
```