

UNIVERSITAS GUNADARMA
FAKULTAS ILMU KOMPUTER & TEKNOLOGI INFORMASI



**IDENTIFIKASI NILAI NUTRISI PADA MAKANAN POPULER
DI INDONESIA BERBASIS CITRA MENGGUNAKAN CNN
DENGAN ARSITEKTUR EFFICIENTNETV2**

Disusun Oleh :

Nama : Lingga Rohadyan
NPM : 10120594
Program Studi : Sistem Informasi
Pembimbing : Prof. Dr. Eri Prasetyo Wibowo, SSi.

**Diajukan Guna Melengkapi Sebagian Syarat
Dalam Mencapai Gelar Sarjana Strata Satu (S1)**

**JAKARTA
2024**

PERNYATAAN ORIGINALITAS DAN PUBLIKASI

Saya yang bertanda tangan di bawah ini,

Nama : Lingga Rohadyan
NPM : 10120594
Judul Skripsi : Identifikasi Nilai Nutrisi pada Makanan Populer di Indonesia Berbasis Citra menggunakan CNN dengan arsitektur EfficientNetV2
Tanggal Sidang : 28 Agustus 2024
Tanggal Lulus : 28 Agustus 2024

Menyatakan bahwa tulisan ini merupakan hasil karya saya sendiri dan dapat dipublikasikan sepenuhnya oleh Universitas Gunadarma. Segala kutipan dalam bentuk apa pun telah mengikuti kaidah dan etika yang berlaku. Mengenai isi dan tulisan merupakan tanggung jawab penulis, bukan Universitas Gunadarma.

Demikian pernyataan ini dibuat dengan sebenarnya dan dengan penuh kesadaran.

Depok, 28 Agustus 2024



(Lingga Rohadyan)

LEMBAR PENGESAHAN

KOMISI PEMBIMBING

No	Nama	Kedudukan
1	Prof. Dr. Eri Prasetyo Wibowo, SSi.	Ketua
2	Dr. Muhammad Subali, SSi., MT.	Anggota
3	Dr. Joko Purnomo, ST., MT.	Anggota

Tanggal Sidang : 28 Agustus 2024

PANITIA UJIAN

No	Nama	Kedudukan
1	Dr. Ravi Ahmad Salim	Ketua
2	Prof. Dr. Wahyudi Priyono	Sekretaris
3	Prof. Dr. Eri Prasetyo Wibowo, SSi.	Anggota
4	Dr. Muhammad Subali, SSi., MT.	Anggota
5	Dr. Joko Purnomo, ST., MT.	Anggota

Tanggal Lulus : 28 Agustus 2024

Mengetahui,

Pembimbing

Bagian Sidang Ujian

(Prof. Dr. Eri Prasetyo Wibowo, SSi.) (Dr. Edi Sukirman, SSi., MM., M.I.Kom)

ABSTRAK

Lingga Rohadyan, 10120594

IDENTIFIKASI NILAI NUTRISI PADA MAKANAN POPULER DI INDONESIA BERBASIS CITRA MENGGUNAKAN CNN DENGAN ARSITEKTUR EFFICIENTNETV2

Skripsi, Jurusan Sistem Informasi, Fakultas Ilmu Komputer dan Teknologi Informasi, Universitas Gunadarma, 2024.

Kata Kunci : Convolutional Neural Network, Transfer Learning, Klasifikasi.
(xiv + 98 + Lampiran)

Teknologi berkembang sangat pesat dan memberikan dampak yang besar bagi kehidupan manusia. Salah satu aspek paling menarik dari perkembangan teknologi saat ini adalah *Artificial Intelligence* (AI). Dalam perkembangan AI, salah satu cabang yang paling berpengaruh adalah *Machine Learning* (ML). Salah satu penerapan ML pada bidang kesehatan yaitu dengan menggunakan teknologi pengenalan gambar untuk mengklasifikasikan makanan dan mengestimasi kandungan nutrisinya. Salah satu teknik ML yang untuk analisis gambar adalah *Convolutional Neural Network* (CNN). Penelitian ini bertujuan untuk menghasilkan model deep learning berbasis arsitektur EfficientNetV2 yang dapat mengklasifikasikan 10 jenis citra makanan dan mengetahui nilai nutrisinya. Dari penelitian ini dapat disimpulkan bahwa model yang dibuat telah berhasil mengklasifikasikan citra makanan dan menampilkan estimasi nutrisinya. HPC DGX A100 terbukti dapat melakukan pelatihan model 2x lebih cepat dari GPU T4. Model yang dilatih menggunakan GPU T4 dengan model yang dilatih menggunakan HPC DGX A100 mempunyai performa dan akurasi yang hampir sama dan tidak berbeda secara signifikan. Pengujian model menggunakan GPU T4 berhasil mendapatkan accuracy 87,50% pada data validasi, dengan rata-rata *precision*, *recall*, dan *F1-Score* 88%. Kemudian pengujian model menggunakan HPC DGX A100 berhasil mendapatkan accuracy 87,25% pada data validasi, dengan *precision* 87%, *recall* 88%, dan *F1-Score* 87%.

Daftar Pustaka (2017 - 2024)

ABSTRACT

Lingga Rohadyan, 10120594

IDENTIFICATION OF NUTRITIONAL VALUE IN POPULAR FOOD IN INDONESIA BASED ON IMAGE USING CNN WITH EFFICIENTNETV2 ARCHITECTURE

*Thesis, Department of Information Systems, Faculty of Computer Science and
Information Technology, Gunadarma University, 2024.*

*Keyword : Convolutional Neural Network, Transfer Learning, Classification.
(xiv + 98 + Attachment)*

Technology is developing very rapidly and has a great impact on human life and society. One of the most interesting aspects of current technological developments is Artificial Intelligence (AI). In development of AI, one of the most influential branches is Machine Learning (ML). One of the applications of ML in the health sector is by using image recognition technology to classify food and estimate its nutritional content. One of the ML techniques for image analysis is Convolutional Neural Network (CNN). This research aims to produce a deep learning model based on the EfficientNetV2 architecture that can classify 10 types of food images and determine their nutritional value. From this research, it can be concluded that the model has successfully classified food images and displayed nutritional estimation. The DGX A100 HPC is proven to be able to perform model training 2x faster than the T4 GPU. The model trained using GPU T4 with the model trained using HPC DGX A100 have similar performance and accuracy which are not significantly different. Evaluation of GPU T4 model managed to get an accuracy of 87.50% on validation data with average precision, recall, and F1-Score of 88%. While evaluation of HPC DGX A100 model managed to get an accuracy of 87.25% on the validation data with precision of 87%, recall 88%, and F1-Score 87%.

Bibliography (2017 - 2024)

KATA PENGANTAR

Puji syukur kehadirat Allah SWT atas rahmat dan karunia nya, penulis dapat menyelesaikan tugas akhir yang berjudul “Identifikasi Nilai Nutrisi pada Makanan Populer di Indonesia Berbasis Citra menggunakan CNN dengan arsitektur EfficientNetV2” dengan baik serta sesuai dengan harapan penulis.

Tugas akhir ini disusun sebagai bagian dari syarat dalam mencapai gelar Sarjana Strata Satu pada Jurusan Sistem Informasi, Fakultas Ilmu Komputer dan Teknologi Informasi, Universitas Gunadarma. Dalam proses menyusun tugas akhir ini tentunya tidak mudah bagi penulis, namun berkat bantuan serta kerjasama dari berbagai pihak yang penulis hormati membuat penulisan tugas akhir ini dapat terselesaikan. Untuk itu penulis mengucapkan terima kasih kepada pihak-pihak terkait, diantaranya adalah:

1. Prof. Dr. E.S. Margianti, SE., MM., selaku Rektor Universitas Gunadarma.
2. Prof. Dr. Rer. Nat, Achmad Benny Mutiara, SSi., SKom., selaku Dekan Fakultas Ilmu Komputer dan Teknologi Informasi Universitas Gunadarma.
3. Dr. Setia Wirawan, SKom., MMSI., selaku Kepala Program Studi Sistem Informasi Universitas Gunadarma.
4. Dr. Edi Sukirman, SSi., MM., M.I.Kom. selaku Kepala Bagian Sidang Ujian Universitas Gunadarma.
5. Prof. Dr. Eri Prasetyo Wibowo, SSi. selaku Dosen Pembimbing yang telah memberikan arahan dan waktunya kepada penulis selama penulisan ini berlangsung hingga selesai.
6. Bapak dan Ibu Dosen Universitas Gunadarma yang telah memberikan ilmu pengetahuan kepada penulis.
7. Kedua orang tua tercinta Irwan Al Gunawan dan Heni Lidyawati yang telah memberikan kasih sayang, semangat, dan doa sehingga penulisan ini berjalan dengan baik.

8. Fanny Limin yang telah memberikan dukungan selama penulisan ini.
9. Mentor Nurhadi yang telah memberikan banyak masukan dan ilmu baru ketika magang di Vedio.
10. Teman-teman seperjuangan saat bimbingan, teman-teman kelas 1KA08, dan kelas 4KA03 yang telah berjuang bersama dalam menyelesaikan studi di Universitas Gunadarma.
11. Semua pihak yang tidak dapat disebutkan satu persatu, terimakasih atas kerja samanya sehingga penulisan ini dapat diselesaikan dengan tepat waktu.

Tentunya penulisan ini jauh dari kata sempurna, namun penulis berharap penulisan ini dapat memberikan manfaat dan menambah wawasan bagi pihak-pihak yang membutuhkan. Dengan segala rasa hormat penulis berharap mendapatkan kritik dan saran yang sifatnya membangun untuk menjadi perbaikan di masa yang akan datang.

Depok, 22 Agustus 2024



Lingga Rohadyan

DAFTAR ISI

COVER	i
PERNYATAAN ORIGINALITAS DAN PUBLIKASI	ii
LEMBAR PENGESAHAN	iii
ABSTRAK	iv
ABSTRACT	v
KATA PENGANTAR	vi
DAFTAR ISI	viii
DAFTAR TABEL	xii
DAFTAR GAMBAR	xiii
BAB I PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	3
1.3 Batasan Masalah	3
1.4 Tujuan Penelitian	4
1.5 Metode Penelitian	4
1.6 Sistematika Penulisan	5
BAB II TINJAUAN PUSTAKA	7
2.1 Artificial Intelligence	7
2.2 Machine Learning	7
2.3 <i>Deep Learning</i>	9
2.3.1 <i>Neural Network</i>	9
2.3.2 Fungsi Aktivasi	10

2.3.3 <i>Dropout</i>	13
2.3.4 <i>Loss Function</i>	14
2.4 Dataset	14
2.5 <i>Epoch</i>	15
2.6 Citra	15
2.7 Augmentasi Citra	16
2.8 Google	17
2.8.1 Google Colaboratory	17
2.9 Python	18
2.10 Tensorflow	18
2.11 Keras	19
2.12 Scikit-learn	20
2.13 Adam <i>Optimizer</i>	22
2.14 <i>Transfer Learning</i>	23
2.15 <i>Convolutional Neural Network</i>	25
2.16 Lapisan <i>Convolutional Neural Network</i> (CNN)	26
2.16.1 <i>Convolutional Layer</i>	26
2.16.2 <i>Activation Layer</i>	26
2.16.3 <i>Pooling Layer</i>	27
2.16.4 <i>Fully Connected Layer</i>	28
2.16.5 <i>Normalization Layer</i>	28
2.16.6 <i>Output Layer</i>	29
2.17 EfficientNet	29
2.17.1 EfficientNetV2	30
2.18 Grad-CAM	31
2.19 <i>Flowchart</i>	32
2.20 <i>Confusion Matrix</i>	33
2.21 Jupyter Notebook	34
2.22 Git	35
2.23 <i>Image Normalization</i>	36
2.24 <i>Web Scraping</i>	37

2.25 <i>Good fit, Overfit, Underfit</i>	37
2.25.1 <i>Good fit</i>	37
2.25.2 <i>Overfit</i>	37
2.25.3 <i>Underfit</i>	38
2.26 Mesin DGX	38
BAB III METODE PENELITIAN	40
3.1 Pengumpulan Data	41
3.2 <i>Data Preprocessing</i>	43
3.2.1 <i>Data Cleaning</i>	44
3.2.2 <i>Undersampling</i>	45
3.2.3 <i>Split Data</i>	46
3.2.4 <i>Data Normalization</i>	47
3.2.5 <i>Data Augmentation</i>	48
3.3 Pembuatan dan Pelatihan Model	49
3.3.1 <i>Transfer Learning EfficientNetV2</i>	50
3.3.2 <i>Training Model</i>	52
3.4 Pengujian Model	56
3.4.1 Evaluasi Model	57
3.4.2 Visualisasi Grad-CAM	59
3.4.3 Pengujian Model Menggunakan Data Nutrisi Makanan	60
BAB IV HASIL DAN PEMBAHASAN	63
4.1 Hasil Pengumpulan Data	63
4.2 Hasil <i>Data Preprocessing</i>	64
4.2.1 Hasil <i>Data Cleaning</i>	64
4.2.2 Hasil <i>Undersampling</i>	65
4.2.3 Hasil <i>Split Data</i>	65
4.2.4 Hasil <i>Data Normalization</i> dan <i>Data Augmentation</i>	66
4.3 Hasil Pembuatan dan Pelatihan Model	67
4.3.1 Hasil <i>Transfer Learning EfficientNetV2</i>	68
4.3.2 Hasil <i>Training Model</i>	69

4.4 Hasil Pengujian Model	73
4.4.1 Hasil Pengujian Model Google Colab GPU T4	74
4.4.2 Hasil Pengujian Model mesin DGX A100 Universitas Gunadarma	83
BAB V PENUTUP	94
5.1 Kesimpulan	94
5.2 Saran	94
DAFTAR PUSTAKA	95
LAMPIRAN	L-1

DAFTAR TABEL

4.1	Hasil Cleaning Data	64
4.2	Hasil Undersampling	65
4.3	Hasil Training Model menggunakan Google Colab GPU T4	69
4.4	Hasil Training Model menggunakan mesin DGX A100 Universitas Gunadarma	71

DAFTAR GAMBAR

2.1	Struktur Neural Network	10
2.2	Grafik Fungsi Aktivasi ReLu	11
2.3	Neural Network Sebelum dan Sesudah Melakukan Dropout	13
2.4	Hasil Teknik Augmentasi Citra	16
2.5	Google Colaboratory	17
2.6	Arsitektur Convolutional Neural Network (CNN)	25
2.7	Proses Max Pooling	27
2.8	Proses Average Pooling	28
2.9	Struktur MBConv dan Fused-MBConv	30
2.10	Arsitektur EfficientNetV2-S (Tan et al. 2021)	31
2.11	Contoh Visualisasi Grad-CAM	31
2.12	Simbol-Simbol Flowchart	32
2.13	Confusion Matrix	33
3.1	Flowchart Tahapan Metode Penelitian	40
3.2	Tampilan kode main.py	42
3.3	Hasil scraping menggunakan Google Image Scraper	43
3.4	Contoh Citra Duplicate	44
3.5	Contoh Citra yang tidak sesuai	45
3.6	Hasil pengumpulan data nutrisi makanan dari situs nilaigizi.com . .	61
4.1	Hasil pengumpulan data citra makanan	63
4.2	Hasil Split Data	65
4.3	Data Sebelum Proses <i>Data Normalization</i> dan <i>Data Augmentation</i> .	66
4.4	Data Sesudah Proses <i>Data Normalization</i> dan <i>Data Augmentation</i> .	67
4.5	Ringkasan Model Hasil Transfer Learning EfficientNetV2	68
4.6	Visualisasi Pelatihan Model Google Colab GPU T4	75
4.7	Hasil Classification Report Model Google Colab GPU T4	76
4.8	Visualisasi Confusion Matrix Model Google Colab GPU T4	77
4.9	Hasil Visualisasi Grad-CAM Model Google Colab GPU T4	81

4.10 Hasil Pengujian Model Google Colab GPU T4 menggunakan data nutrisi makanan	83
4.11 Visualisasi Pelatihan Model mesin DGX A100 Universitas Gunadarma	85
4.12 Hasil Classification Report Model mesin DGX A100 Universitas Gunadarma	86
4.13 Visualisasi Confusion Matrix Model mesin DGX A100 Universitas Gunadarma	87
4.14 Hasil Visualisasi Grad-CAM Model mesin DGX A100 Universitas Gunadarma	91
4.15 Hasil Pengujian Model mesin DGX A100 Universitas Gunadarma menggunakan data nutrisi makanan	93

BAB I

PENDAHULUAN

1.1 Latar Belakang

Teknologi berkembang sangat pesat dan memberikan dampak yang besar bagi kehidupan manusia dan masyarakat. Teknologi telah membuat kehidupan manusia dan masyarakat lebih nyaman dan efisien di berbagai bidang seperti ekonomi, pendidikan, komunikasi, transportasi, dan hiburan. Di era digital ini, perkembangan teknologi terus berlanjut dan berkembang sesuai dengan kebutuhan masyarakat dan permasalahan sosial yang semakin kompleks. Perkembangan dan inovasi teknologi yang berkelanjutan dapat membawa manfaat besar bagi manusia dan masyarakat jika dikelola dengan baik.

Salah satu aspek paling menarik dari perkembangan teknologi ini adalah kemajuan pesat dalam bidang *Artificial Intelligence* (AI). Menurut (Abbass, 2021) Kecerdasan Buatan adalah fenomena sosial dan kognitif fenomena yang memungkinkan mesin untuk berintegrasi secara sosial dengan masyarakat untuk melakukan tugas-tugas kompetitif yang membutuhkan proses kognitif dan berkomunikasi dengan entitas lain dalam masyarakat dengan mengubah pesan dengan konten informasi yang tinggi dan lebih pendek.

Dalam perkembangan AI, salah satu cabang yang paling berpengaruh adalah *Machine Learning* (ML). Menurut (Rebala et al., 2019) ML adalah bidang ilmu komputer yang mempelajari algoritma dan teknik untuk mengotomatisasi solusi untuk masalah kompleks yang sulit diprogram menggunakan metode pemrograman konvensional. ML telah menjadi komponen penting dalam berbagai aplikasi modern, termasuk pengenalan gambar, analisis data, dan sistem rekomendasi.

(Sarker, 2021) menjelaskan bahwa ML memiliki berbagai jenis algoritma, termasuk analisis klasifikasi, analisis regresi, pengelompokan data, dan banyak lagi. Algoritma-algoritma ini dapat diterapkan untuk meningkatkan kecerdasan dan kemampuan berbagai aplikasi di dunia nyata. Penerapan ML telah memberikan dampak signifikan dalam berbagai bidang, seperti sistem keamanan siber, kota pintar, perawatan kesehatan, *e-commerce*, pertanian, dan banyak lagi.

Berhubungan dengan kesehatan, nutrisi menjadi semakin diperhatikan oleh masyarakat Indonesia, terutama sejak presiden terpilih Indonesia Prabowo Subianto meluncurkan program untuk menyediakan makanan bergizi dan gratis bagi anak-anak. Inisiatif ini menyoroti pentingnya nutrisi bagi masyarakat Indonesia. Berdasarkan penelitian (Fitria et al., 2022) yang dilakukan pada 127 siswa di SMA Muhammadiyah 13 Jakarta, sebagian besar siswa Sekolah Menengah Atas (SMA) Muhammadiyah 13 Jakarta masih memiliki pengetahuan gizi seimbang yang kurang yaitu sebesar 53,5%. Salah satu pendekatan untuk mengatasi masalah ini yaitu dengan menggunakan teknologi machine learning.

Teknologi machine learning dapat membantu mengatasi masalah ini dengan menyediakan alat untuk mengidentifikasi dan menganalisis kandungan nutrisi dalam makanan, sehingga membantu individu membuat keputusan yang lebih sehat. Salah satu pendekatan yang menjanjikan adalah penggunaan teknologi pengenalan gambar untuk mengklasifikasikan makanan dan mengestimasi kandungan nutrisinya.

Dalam konteks ini, salah satu teknik ML yang sangat efektif untuk analisis gambar adalah *Convolutional Neural Network* (CNN). CNN adalah jaringan saraf tiruan yang terdiri dari beberapa lapisan neuron yang dihubungkan dalam pola tertentu dan dioptimalkan untuk memproses *array* data terstruktur seperti gambar. CNN dirancang untuk secara otomatis dan adaptif mempelajari hierarki spasial dari fitur melalui algoritma *backpropagation* dengan menggunakan beberapa blok bangunan, seperti lapisan konvolusi, lapisan *pooling*, dan lapisan *fully connected* (Yamashita et al., 2018)

Convolutional Neural Network (CNN) telah terbukti efektif dalam klasifikasi dan pengenalan objek pada citra, termasuk makanan. (Rajayogi et al., 2019) mengklasifikasikan citra makanan untuk program diet dan ekstraksi kalori berbasis citra. Penelitian ini menggunakan dataset makanan India yang terdiri dari 20 kelas dan mempunyai 500 citra untuk setiap kelas. Penelitian ini menggunakan teknik *transfer learning* serta menggunakan beberapa model yakni InceptionV3, VGG16, VGG19, dan ResNet untuk mengawasi kebiasaanmakan agar pola hidup lebih sehat. Nilai akurasi tertinggi yang didapatkan yakni 87,9% dan loss rate 0,5893 dari model InceptionV3 dibandingkan model yang lain seperti VGG19 yang mendapatkan nilai

akurasi 78,9%, VGG16 78,2%, dan ResNet 69,91%.

Arsitektur EfficientNetV2 merupakan pengembangan terbaru dari CNN yang menawarkan performa tinggi dengan efisiensi komputasi yang lebih baik. Arsitektur EfficientNetV2 merupakan salah satu model pengolahan citra keluaran baru yakni pada tahun 2021 dari keluarga EfficientNet. EfficientNetV2 memiliki 11x lebih cepat dalam pelatihan dan model yang 6.8x lebih kecil (Tan et al., 2021). (Karthik et al., 2022) melakukan klasifikasi citra untuk mengidentifikasi penyakit kulit yang dilakukan pada empat kelas dengan menggunakan model EfficientNetV2 dan model ini mendapatkan nilai akurasi pengujian keseluruhan sebesar 84,70%. Dataset terdiri dari 10.399 data latih dan 3.465 data uji.

Dengan latar belakang ini, penelitian ini bertujuan untuk mengembangkan sistem klasifikasi citra makanan populer di Indonesia menggunakan CNN dengan arsitektur EfficientNetV2 untuk mengetahui kandungan nutrisinya, terutama kalori.

1.2 Rumusan Masalah

Berdasarkan latar belakang diatas maka dapat dirumuskan masalah pada tulisan ini sebagai berikut:

1. Bagaimana melakukan indentifikasi nilai nutrisi menggunakan klasifikasi citra pada citra makanan populer Indonesia menggunakan metode *Convolutional Neural Network* (CNN) dengan arsitektur EfficientNetV2?
2. Bagaimana tingkat akurasi yang dihasilkan dari klasifikasi citra makanan populer di Indonesia menggunakan metode *Convolutional Neural Network* (CNN) dengan arsitektur EfficientNetV2?

1.3 Batasan Masalah

Berdasarkan latar belakang dan rumusan masalah yang telah dijelaskan dalam penulisan ini, terdapat 7 batasan masalah, yaitu:

1. Data citra yang digunakan adalah citra berbagai jenis makanan yang dikumpulkan dari *google images*.

2. Data citra yang dikumpulkan terdiri dari 10 jenis makanan dengan 200 citra tiap jenis makanan dengan total 2000 data citra.
3. Data citra terbagi menjadi data latih sebesar 80% dan data validasi sebesar 20%.
4. Bahasa pemrograman yang digunakan adalah Python.
5. Klasifikasi Citra dilakukan menggunakan metode *Convolutional Neural Network* dengan arsitektur EfficientNetV2.
6. Data nutrisi makanan diambil dari situs nilaigizi.com.
7. Data nutrisi makanan merupakan data per porsi atau per 100 gram.

1.4 Tujuan Penelitian

Berdasarkan permasalahan yang telah dijabarkan, tujuan dari penulisan tugas ini antara lain:

1. Menghasilkan model *deep learning* berbasis arsitektur EfficientNetV2 yang dapat mengklasifikasikan citra makanan dan mengetahui nilai nutrisinya.
2. Menghasilkan performa model *deep learning* berbasis arsitektur EfficientNetV2 dengan minimum akurasi 80%.
3. Mengetahui perbedaan akurasi model menggunakan Google Colab *Graphics Processing Unit* (GPU) T4 dengan *High Performance Computing* (HPC) DGX A100.

1.5 Metode Penelitian

Terdapat 4 tahapan yang dilakukan pada penelitian ini agar penelitian dapat berjalan dengan baik dan sesuai tujuan, yaitu sebagai berikut:

1. Pengumpulan Dataset

Tahap pertama yang dilakukan pada penelitian ini adalah mengumpulkan data citra dari *google images*. Data citra yang digunakan terdiri dari 10 jenis

makanan Indonesia yang populer, yaitu ayam bakar, bakso, gado-gado, gudeg, nasi goreng, pempek, rawon, rendang, sate, dan soto. Penulis membagi dataset dengan rasio 80% data latih dan 20% data validasi.

2. Data Preprocessing

Tahap ini melakukan serangkaian proses berupa *image normalization* dan *image augmentation*. Citra yang diperoleh kemudian dinormalisasi dan diolah menggunakan *image augmentation* yang bertujuan untuk memperbanyak jumlah citra pada data latih.

3. Pembuatan dan Pelatihan Model

Tahap ini melalui serangkaian proses berupa pembuatan model dan pelatihan model. Proses pembuatan model menggunakan metode *transfer learning* guna mempersingkat waktu perancangan model dan meningkatkan akurasi. Tahapan proses pelatihan model mengimplementasikan perancangan model EfficientNetV2 yang telah dibuat dengan melakukan pelatihan terhadap dataset hasil *preprocessing* yang berupa citra makanan.

4. Pengujian Model

Penulis menguji model dengan data validasi yang sebanyak 400 data citra. Pengujian model memiliki tujuan untuk mengetahui keberhasilan proses pelatihan model dilakukan pada data citra baru. Hasil pengujian model menjadi standar penulis untuk mengukur kinerja model. Proses pengukuran performa model berdasarkan perhitungan 4 jenis metrik, yaitu *accuracy*, *precision*, *recall*, dan *F-1 score*.

1.6 Sistematika Penulisan

Sistematika pada penulisan ini dibagi ke dalam lima bab yang akan dituliskan berurut, dimana setiap bab memiliki prosedur-prosedur yang akan dibahas, yaitu:

BAB I PENDAHULUAN

Pada bab pendahuluan terdapat informasi mengenai, latar belakang masalah, rumusan masalah, batasan masalah, tujuan penelitian, metode penelitian dan sistematika penulisan.

BAB II TINJAUAN PUSTAKA

Pada bab ini berisi informasi mengenai teori-teori pendukung yang menjadi sumber informasi pada penelitian ini, selain itu terdapat penjelasan-penjelasan singkat mengenai *software* pendukung penelitian.

BAB III METODE PENELITIAN

Bab ini berisi penjelasan langkah-langkah dalam penelitian dan gambaran umum dalam membangun model *deep learning* dari penelitian serta penjelasan kode yang digunakan.

BAB IV HASIL DAN PEMBAHASAN

Bab ini merupakan bagian yang membahas tentang hasil pengolahan data dan analisa pada model yang telah dirancang serta hasil uji coba dari program.

BAB V PENUTUP

Pada bab penutup meliputi kesimpulan dan saran, kesimpulan akan berkaitan mengenai keberhasilan penelitian terhadap tujuan penelitian yang dilakukan serta saran-saran berisi informasi langkah yang mungkin dapat menjadi penyempurnaan pada penelitian yang dilakukan.

BAB II

TINJAUAN PUSTAKA

2.1 Artificial Intelligence

Artificial Intelligence (AI) atau kecerdasan buatan merupakan bidang ilmu komputer yang bertujuan untuk mengembangkan mesin atau program komputer yang dalam melakukan tugas yang biasanya memerlukan kecerdasan manusia, seperti pengenalan wajah, pengenalan suara, bahasa alami, analisis data, dan pengambilan keputusan. Kecerdasan buatan mencakup berbagai teknologi seperti *machine learning, deep learning, natural language processing, image processing*, dan *robotics*.

Menurut (Abbass, 2021) Kecerdasan Buatan adalah fenomena sosial dan kognitif fenomena yang memungkinkan mesin untuk berintegrasi secara sosial dengan masyarakat untuk melakukan tugas-tugas kompetitif yang membutuhkan proses kognitif dan berkomunikasi dengan entitas lain dalam masyarakat dengan mengubah pesan dengan konten informasi yang tinggi dan lebih pendek.

2.2 Machine Learning

Machine learning adalah sub-bidang dari *Artificial Intelligence* (AI) yang fokus pada pengembangan algoritma dan teknik untuk membuat sistem komputer yang dapat belajar dan meningkatkan performanya secara otomatis dari pengalaman. *Machine learning* bertujuan untuk memungkinkan komputer untuk mengenali pola dalam data dan membuat keputusan berdasarkan pola tersebut.

Menurut (Rebala et al., 2019) *Machine learning* adalah bidang ilmu komputer yang mempelajari algoritme dan teknik untuk mengotomatisasi solusi untuk masalah kompleks yang sulit diprogram menggunakan metode pemrograman konvensional. *Machine learning* secara umum dibagi menjadi 2 tipe, yaitu *supervised learning* dan *unsupervised learning*.

Supervised learning adalah salah satu jenis pembelajaran mesin (*machine learning*) di mana model atau algoritma belajar dari data yang telah diberi label. Dalam *supervised learning*, data yang digunakan untuk melatih model atau algoritma

terdiri dari pasangan input dan output yang terkait. Input disebut fitur (*features*) dan output disebut label atau target. Tujuan dari *supervised learning* adalah untuk mempelajari hubungan antara fitur dan label, dan menggunakan hubungan tersebut untuk membuat prediksi atau klasifikasi pada data baru yang belum pernah dilihat sebelumnya. Dalam *supervised learning*, model atau algoritma belajar dari data latih dengan mengoptimalkan fungsi objektif yang telah ditentukan, dengan cara menyesuaikan parameter model atau algoritma sehingga dapat meminimalkan kesalahan prediksi pada data latih. Contoh aplikasi *supervised learning* meliputi klasifikasi email sebagai spam atau bukan spam, prediksi harga rumah berdasarkan fitur seperti lokasi, jumlah kamar, dan luas tanah, serta pengenalan gambar berdasarkan label atau kategori tertentu. Beberapa algoritma *supervised learning* yang populer antara lain regresi linear, regresi logistik, *decision tree*, dan *random forest*.

Unsupervised learning adalah jenis pembelajaran mesin (*machine learning*) di mana model atau algoritma belajar dari data yang tidak memiliki label atau informasi target. Dalam *unsupervised learning*, data yang digunakan untuk melatih model atau algoritma hanya terdiri dari fitur atau atribut dari data tersebut. Tujuan dari *unsupervised learning* adalah untuk menemukan pola atau struktur yang tersembunyi dalam data, tanpa bantuan label atau informasi target yang telah diketahui sebelumnya. Dalam *unsupervised learning*, model atau algoritma harus mampu mengklasifikasikan data ke dalam kelompok atau kategori yang berbeda berdasarkan kemiripan atau kesamaan antara fitur atau atribut yang dimiliki oleh data tersebut. Contoh aplikasi *unsupervised learning* meliputi pengelompokan data ke dalam kelompok-kelompok yang terkait, deteksi anomali dalam data, dan reduksi dimensi data untuk mempermudah analisis. Beberapa algoritma *unsupervised learning* yang populer antara lain *k-means clustering*, *hierarchical clustering*, *principal component analysis* (PCA), dan *autoencoder*.

Perbedaan mendasar antara *supervised learning* dan *unsupervised learning* adalah pada jenis data yang digunakan untuk melatih model atau algoritma. *Supervised learning* menggunakan data yang telah diberi label atau informasi target, sedangkan *unsupervised learning* menggunakan data tanpa label atau informasi

target.

2.3 Deep Learning

Deep Learning sub-bidang kecerdasan buatan yang berfokus pada pembuatan model jaringan saraf yang besar yang mampu membuat keputusan berbasis data yang akurat. *Deep learning* sangat cocok untuk konteks di mana datanya kompleks dan di mana tersedia kumpulan data yang besar (Kelleher, 2019).

Deep learning telah mencapai kesuksesan besar dalam berbagai bidang, termasuk pengenalan suara, pengenalan wajah, pengenalan tulisan tangan, pengenalan objek dalam citra, dan bahasa alami. Keunggulan *deep learning* terutama terlihat dalam kemampuannya untuk belajar dari data yang sangat besar dan kompleks, serta kemampuannya untuk mengatasi masalah yang sulit dipecahkan dengan pendekatan tradisional.

Beberapa teknik atau arsitektur deep learning yang populer antara lain *Convolutional Neural Network* (CNN) untuk pengenalan citra, *Recurrent Neural Network* (RNN) untuk pengenalan suara dan bahasa alami, dan *Generative Adversarial Network* (GAN) untuk sintesis data.

Dalam *deep learning*, model atau algoritma diatur dalam beberapa lapisan tersembunyi yang membentuk suatu hierarki untuk memproses data secara bertahap. Setiap lapisan memiliki sejumlah neuron atau unit yang terhubung dengan lapisan sebelumnya dan sesudahnya. Proses pembelajaran pada *deep learning* melibatkan optimisasi parameter model atau algoritma dengan mengoptimalkan fungsi objektif berdasarkan data latih.

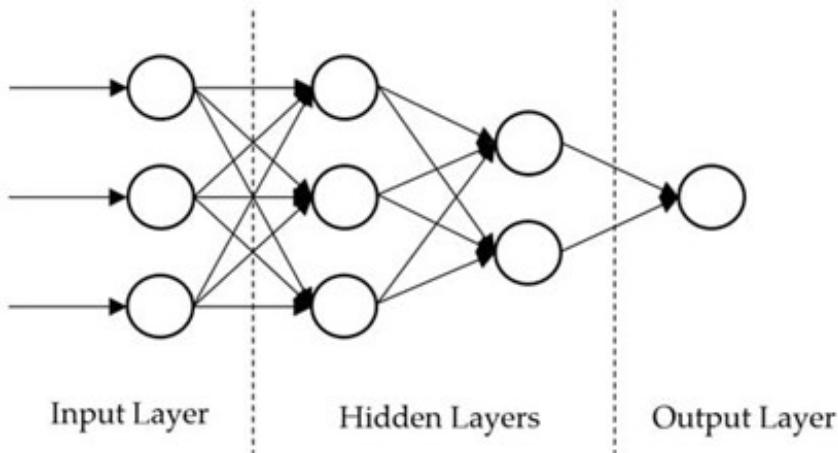
Deep learning memerlukan data yang besar dan terdiversifikasi, serta komputasi yang sangat cepat dan efisien untuk melatih model atau algoritma yang kompleks. Oleh karena itu, *deep learning* umumnya digunakan pada masalah yang sangat kompleks dan memerlukan tingkat akurasi yang sangat tinggi.

2.3.1 Neural Network

Neural network atau jaringan saraf merupakan istilah yang pertama kali digunakan oleh McCulloch & Pitts (1990) pada percobaan dalam menemukan

representasi matematis dari pemrosesan informasi dalam sistem biologis. Jaringan saraf merupakan jaringan dari node (simpul), yang meniru struktur neuron otak dari manusia. Node menghitung jumlah nilai bobot dari masukan dan memprosesnya pada lapisan tersembunyi, kemudian mengeluarkan hasil dari fungsi aktivasi dengan nilai bobot.

Jaringan saraf telah dikembangkan dari arsitektur sederhana menjadi struktur yang semakin kompleks. Awalnya, pelopor jaringan saraf memiliki arsitektur yang sangat sederhana dengan hanya lapisan input dan output, yang disebut jaringan saraf *single layer*. Ketika lapisan tersembunyi atau *hidden layer* ditambahkan ke jaringan saraf tersebut, maka akan menghasilkan jaringan saraf *multi-layer*. Oleh karena itu, jaringan saraf *multi-layer* terdiri atas lapisan input, lapisan tersembunyi, dan lapisan output seperti pada Gambar 2.1.



Gambar 2.1: Struktur Neural Network

2.3.2 Fungsi Aktivasi

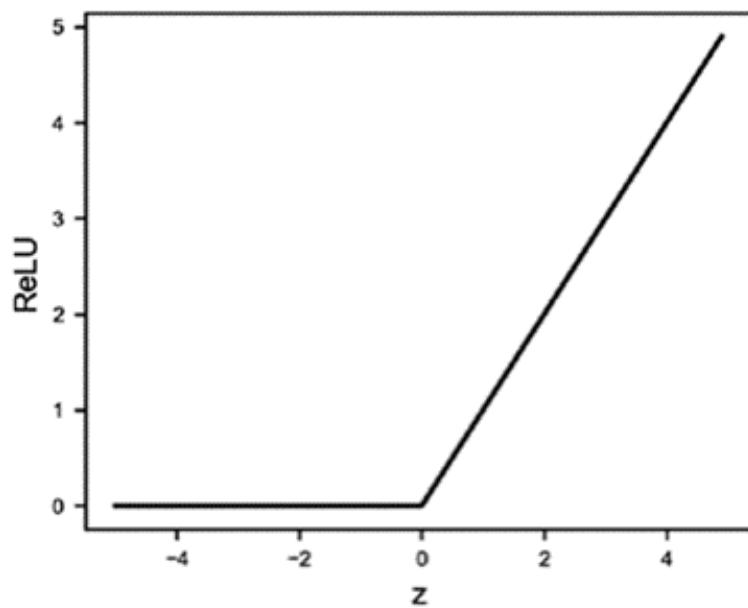
Fungsi aktivasi adalah sebuah persamaan matematika yang tujuannya untuk mengaktifkan node yang ada di jaringan saraf. Node yang aktif akan menghasilkan keluaran sesuai dengan jenis fungsi aktivasi. Terdapat beberapa jenis fungsi aktivasi yang dapat digunakan sesuai dengan kebutuhan. Fungsi aktivasi yang umum digunakan dalam proses klasifikasi adalah fungsi aktivasi *Rectified Linear Unit* (ReLU) dan juga fungsi aktivasi softmax. Berikut merupakan penjelasan lebih lanjut

dari fungsi aktivasi tersebut:

1. ***Rectified Linear Unit***

Rectified Linear Unit (ReLU) merupakan sebuah fungsi aktivasi yang sering digunakan pada pembuatan model *machine learning*. Fungsi ReLU sering digunakan pada lapisan-lapisan tersembunyi (*hidden layers*) karena kesederhanaan komputasi yang dilakukan oleh fungsi ini. Hal ini dikarenakan tidak adanya komputasi yang terlalu rumit sehingga membuat proses pelatihan suatu model dapat dijalankan dalam waktu yang relatif singkat.

ReLU menggunakan fungsi $f(z) = \max(0, z)$, yang artinya jika output positif maka akan menghasilkan nilai yang sama, jika tidak maka akan menghasilkan nilai 0. ReLU tidak hanya meningkatkan kinerja secara signifikan tetapi juga membantu mengurangi jumlah perhitungan selama fase pelatihan. Hal ini terjadi akibat dari nilai 0 dalam output ketika nilai z negatif, sehingga menonaktifkan neuron (Moolayil, 2019).



Gambar 2.2: Grafik Fungsi Aktivasi ReLU

2. ***Softmax***

Softmax merupakan fungsi aktivasi yang menginterpretasikan vektor nilai real X menjadi vektor nilai real sebagai probabilitas. Nilai probabilitas *softmax* bergantung pada nilai recall yang dimasukkan ke dalam fungsi ini. Jika salah satu nilai input kecil atau negatif, nilai output dari fungsi *softmax* adalah nilai probabilitas rendah. Nilai probabilitas tinggi diperoleh ketika fungsi ini menerima nilai masukan yang besar atau positif. Semua nilai output dari fungsi *softmax* terletak antara 0 dan 1.

Fungsi *softmax* sering digunakan dalam masalah klasifikasi dengan syarat kelas-kelas dalam data saling eksklusif. Hasil perhitungan MLP biasanya berupa titik-titik nyata, yang tidak mudah diskalakan dan sulit dimanipulasi. Fungsi *softmax* mengubah titik-titik ini menjadi nilai probabilitas yang dinormalisasi, dan nilai probabilitas ini dapat digunakan sebagai input ke sistem lain. Ini adalah dasar untuk menggunakan fungsi aktivasi *Softmax* di lapisan keluaran model pembelajaran mendalam seperti CNN. Definisi matematis dari *softmax* sebagai persamaan 2.1 berikut:

$$S(x)_i = \frac{e^{x_i}}{\sum_{j=1}^n e^{x_j}} \quad (2.1)$$

Dengan keterangan:

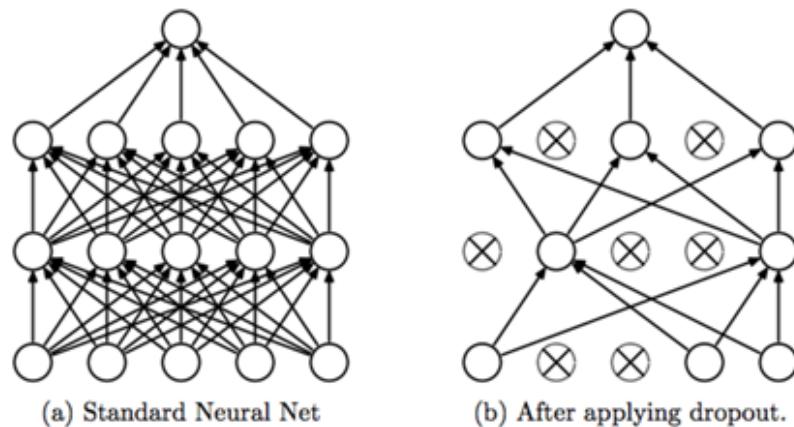
- x = Vektor input ke fungsi softmax, terdiri dari (x_0, \dots, x_K)
- x_i = Elemen dari vektor input ke fungsi softmax, dan mereka dapat mengambil nilai real apa pun, positif, nol, atau negatif.
- e^{x_j} = Fungsi eksponensial standar diterapkan pada setiap elemen dari vektor input. Ini memberikan nilai positif di atas 0, yang akan sangat kecil jika inputnya negatif, dan sangat besar jika inputnya besar. Namun, itu masih belum ditetapkan dalam kisaran $(0, 1)$ yang diperlukan dari suatu probabilitas.
- $\sum_{j=1}^n e^{x_j}$ Normalisasi. Normalisasi memastikan bahwa semua nilai keluaran fungsi akan berjumlah 1 dan masing-masing berada dalam kisaran $(0, 1)$, sehingga merupakan distribusi probabilitas yang valid.

- n = Jumlah kelas dalam multi-class classifier.
- Nilai x_i adalah elemen dari vektor input dan dapat mengambil nilai ril.

Fungsi *softmax* dinormalisasi seperti pada persamaan 2.1. Tujuannya adalah untuk memastikan bahwa semua nilai keluaran bersama-sama memiliki nilai 1, yang menunjukkan bahwa nilai ini adalah nilai probabilitas yang valid. *Softmax* memperluas ide ini ke klasifikasi multi-kelas atau biasa disebut dengan klasifikasi multi-kelas. *Softmax* memberikan nilai probabilitas untuk setiap kelas tugas klasifikasi. Nilai probabilitas total yang ditentukan harus 1, yang menunjukkan bahwa nilai tersebut merupakan nilai probabilitas yang valid.

2.3.3 *Dropout*

Dropout merupakan teknik yang digunakan untuk menghindari *overfitting* model. Dalam metode ini, aktivasi beberapa neuron yang dipilih secara acak dalam jaringan diasumsikan nol selama pelatihan. Neuron yang dipilih diubah di setiap iterasi pelatihan. Proses pembelajaran menjadi lebih andal dengan metode ini dan *overfitting* berkurang.



Gambar 2.3: Neural Network Sebelum dan Sesudah Melakukan Dropout

Istilah *dropout* mengacu pada pemutusan neuron (tersembunyi dan terlihat) dalam *neural network*. Dengan mengeluarkan unit (neural) untuk sementara menghapusnya dari jaringan (*network*), bersama dengan semua koneksi masuk dan keluarannya, seperti yang ditunjukkan pada Gambar 2.3. Pemilihan unit yang dijatuhan secara acak (Yadav, 2022).

2.3.4 Loss Function

Loss function, atau disebut juga *cost function*, adalah suatu fungsi matematis yang digunakan dalam *machine learning* untuk mengukur seberapa baik model memetakan input ke output yang diharapkan. *Loss function* menghitung selisih antara prediksi model dengan nilai *ground truth*, atau nilai yang seharusnya dihasilkan oleh model.

Tujuan dari *loss function* adalah untuk mengoptimalkan model agar menghasilkan prediksi yang semakin mendekati nilai *ground truth*. Untuk itu, *loss function* sering digunakan sebagai acuan dalam proses optimasi, di mana model akan mencoba meminimalkan nilai *loss function* tersebut dengan menyesuaikan parameter model. Contoh umum dari *loss function* adalah *Mean Squared Error* (MSE), *cross-entropy loss*, dan *hinge loss*. Pemilihan *loss function* yang tepat sangat penting dalam pengembangan model, tergantung pada jenis masalah dan tipe data yang digunakan.

2.4 Dataset

Dataset merupakan istilah yang merujuk pada kumpulan data. Dataset berisi lebih dari satu variabel dan menyangkut suatu topik tertentu. Dataset adalah representasi di memori dari satu tabel atau lebih dan digunakan untuk menyimpan baris yang didapatkan saat permintaam dikirim ke basis data. dataset dapat ditambahkan, dihapus, atau diperbarui.

Dataset tetap ada di memori dan data didalamnya bisa dimanipulasi dan diperbarui tanpa bergantung pada asalnya. Jika diperlukan, dataset bisa bertindak sebagai template untuk memperbarui data pusat. Dataset dapat dibagi menjadi data training dan data testing. Data training digunakan untuk melatih algoritma dalam mencari model yang sesuai sedangkan data testing akan dipakai untuk menguji dan mengetahui performa model yang didapatkan pada tahap testing.

2.5 *Epoch*

Epoch adalah ketika seluruh dataset sudah melalui proses training pada Neural Network sampai dikembalikan ke awal untuk sekali putaran, karena satu *epoch* terlalu besar untuk dimasukkan kedalam komputer maka dari itu perlu membaginya kedalam satuan kecil. *Epoch* digunakan untuk mengoptimalkan pembelajaran dan grafik yang digunakan.

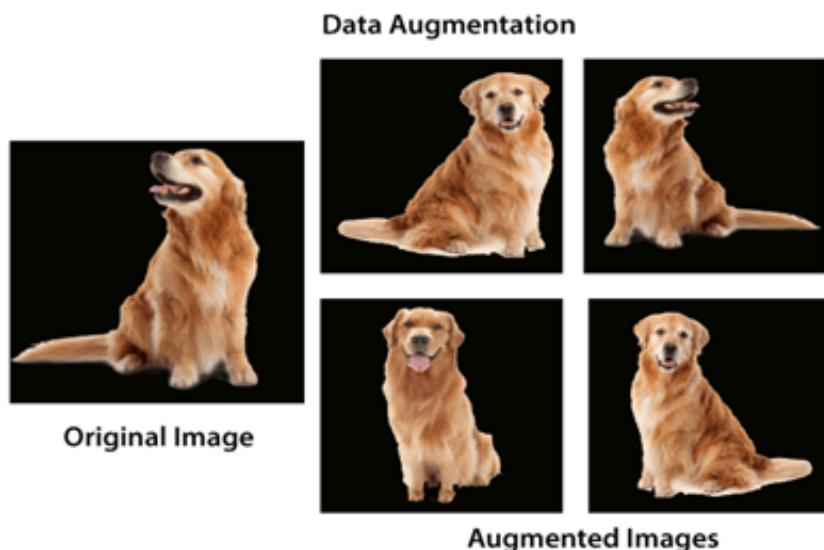
Seiring bertambahnya jumlah *epoch*, semakin banyak pula bobot yang berubah dalam neural network. Jumlah *epoch* yang digunakan tidak ditentukan, jumlah epoch yang digunakan terkait dengan beragamnya data yang digunakan atau tergantung dengan dataset yang dimiliki.

2.6 Citra

Citra merupakan suatu gambaran atau kemiripan dari suatu objek. Citra analog tidak dapat dipresentasikan dalam komputer, sehingga tidak bisa diproses oleh komputer secara langsung. Citra analog harus dikonversi menjadi citra yang dapat diolah oleh komputer sedangkan citra yang dihasilkan dari peralatan digital (Citra Digital) langsung bisa diolah oleh komputer. Citra di dalam peralatan digital terdapat sistem sampling dan kuantisasi sedangkan peralatan analog tidak dilengkapi kedua sistem tersebut. Sistem sampling adalah sistem yang mengubah citra kontinu menjadi Citra Digital dengan cara membagi Citra Analog menjadi M baris dan N kolom, sehingga menjadi citra diskrit. Semakin besar nilai M dan N, semakin halus citra digital yang dihasilkan. Pertemuan antara baris dan kolom disebut pixel. Sistem kuantisasi adalah Sistem yang melakukan pengubahan intensitas analog ke intensitas diskrit, sehingga dengan proses ini dimungkinkan untuk membuat gradasi warna sesuai dengan kebutuhan. Kedua sistem inilah yang bertugas untuk memotong-motong Citra menjadi M baris dan N kolom (proses sampling) sekaligus menentukan besar intensitas yang terdapat di titik tersebut (proses kuantisasi), sehingga menghasilkan resolusi Citra yang diinginkan (Andono et al., 2017).

2.7 Augmentasi Citra

Augmentasi citra adalah sebuah teknik yang dapat menambahkan jumlah data latih dengan menghasilkan banyak varian yang realistik dari setiap contoh data latih (Géron, 2019). Teknik ini dapat mengurangi kemungkinan permasalahan *overfit* yang menjadikan teknik ini merupakan salah satu jenis teknik regularisasi. Data citra hasil augmentasi harus serealistik mungkin sehingga manusia tidak dapat membedakan data hasil augmentasi dan data sebenarnya. Oleh karena itu, data artifisial dibuat dengan cara menggeser, memutar, dan mengubah ukuran dari setiap citra yang terdapat dalam data latih. Hasil dari proses pengubahan citra tersebut akan dimasukan ke dalam data latih.



Gambar 2.4: Hasil Teknik Augmentasi Citra

Gambar 2.4 adalah contoh dari penggunaan teknik augmentasi citra. Hasil dari teknik ini menyerupai citra yang sesungguhnya. Teknik augmentasi citra biasa digunakan ketika dihadapkan dengan kondisi data yang sedikit dan sekiranya data tersebut kurang variasi. Data yang sedikit akan membuat model kurang mengenal variasi dari suatu citra sehingga model tersebut cenderung *overfit*. Teknik augmentasi citra dapat membuat suatu model menambah pengetahuan mengenai fitur dari suatu citra dari hasil augmentasi citra. Semakin banyak fitur yang dipelajari dari sebuah citra, hal ini dapat meminimalisir kemungkinan model menjadi *overfit*.

2.8 Google

Google adalah perusaham mesin pencari yang didirikan pada tahun 1998 oleh Sergey Brin dan Larry Page. Saat ini kantor utama Google ada di Mountain View, California. Lebih dari 70% permintaan pencari online di seluruh dunia telah ditangani oleh Google. Mesin pencari Google merupakan situs yang paling sukses dan popular. Banyaknya layanan dan produk online yang dapat digunakan seperti akun email, browser web, software produktivitas, ponsel dan aplikasi, alat pemetaan, *e-book*, serta berbagai layanan lainnya membuat para penggunanya senang untuk menggunakan Google.

2.8.1 Google Colaboratory

Google Colaboratory lebih sering disebut sebagai "Google Colab" atau sekadar "Colab" adalah proyek penelitian untuk membuat prototipe model pembelajaran mesin pada opsi perangkat keras canggih seperti GPU dan TPU. Ini menyediakan lingkungan notebook Jupyter tanpa server untuk pengembangan interaktif. Google Colab gratis untuk digunakan seperti produk Google lainnya (Bisong, 2019). Google



Gambar 2.5: Google Colaboratory

Colab atau Colaboratory adalah layanan cloud gratis yang dihosting oleh Google untuk mendorong penelitian *Machine Learning* dan *Artificial Intelligence*, yang sering kali menjadi penghalang pembelajaran dan kesuksesan adalah persyaratan kekuatan komputasi yang sangat besar (Naik, 2023).

2.9 Python

Code Python adalah bahasa pemrograman interpretatif yang bisa dipasang pada berbagai platform, khususnya platform yang berfokus pada keterbacaan kode. Kode Python bisa di-embed ke bahasa lain seperti C dan Java, atau sebaliknya, dari bahasa C atau Java ke Python. Pemrograman Python itu merupakan salah satu bahasa pemrograman yang dapat melakukan eksekusi sejumlah instruksi multu guna secara langsung dengan metode Object Oriented Programming dan menggunakan semantik dinamis untuk memberikan tingkat keterbacaan sintak.

Pemrograman Python memiliki bahasa yang kemampuan, menggabungkan kapabilitas dan sintaksis kode yang jelas dan dilengkapi dengan fungsionalitas pustaka standar yang besar serta komprehensif. Pemrograman Bahasa Python difokuskan untuk digunakan dalam menganalisis data, visualisasi data, membuat dan mengembangkan AI. Pemrograman Python adalah pemrograman yang paling mudah di pelajari dengan code yang pendek dan tidak susah. Python memiliki pustaka (*library*) yang luas dan dapat dikembangkan ke bidang-bidang lainnya. Beberapa *library* python yang popular dalam Data Science dan AI adalah Scikit-Learn, TensorFlow,PyTorch.

Pemrograman bahasa Python merupakan bahasa pemrograman yang tidak menggunakan compiler. Dengan sifat open-source yang dimilikinya, pengguna dapat mempelajari Python dengan mudah karena bahasa ini dapat digunakan dalam membuat situs,mengembangkan situs, mengembangkan video game, membangun GUI Desktop, dan mengembangkan perangkat lunak (Setiawan et al., 2022).

2.10 Tensorflow

TensorFlow adalah salah satu pustaka deep learning yang paling populer, yang awalnya dikembangkan oleh peneliti di Google. TensorFlow sangat memudahkan dan mempercepat penelitian dan penerapan model jaringan saraf. Pustaka ini mencakup konsep-konsep inti seperti fungsi konstruksi graf, alat eksekusi graf, dan alat visualisasi TensorBoard (Pang et al., 2020).

TensorFlow adalah sebuah platform open source untuk *machine learning*

yang dibuat oleh Google. TensorFlow memungkinkan pengembang dan peneliti untuk membuat dan melatih model *machine learning* dari berbagai jenis data, seperti gambar, teks, dan suara. TensorFlow menggunakan konsep tensor, yaitu struktur data multidimensi, sebagai representasi dari data masukan dan keluaran dalam model *machine learning*. TensorFlow menyediakan banyak pustaka dan algoritma *machine learning* yang dapat digunakan untuk membangun berbagai jenis model, seperti *neural network*, *decision tree*, dan *regression model*. TensorFlow memiliki fitur yang kuat untuk distribusi dan skalabilitas, sehingga dapat digunakan untuk menangani data yang sangat besar dan kompleks. TensorFlow juga mendukung berbagai platform, termasuk *desktop*, *server*, dan perangkat *mobile*.

Salah satu kelebihan TensorFlow adalah mudahnya penggunaannya, terutama untuk pengguna Python, karena TensorFlow menyediakan API yang mudah digunakan dan dokumentasi yang lengkap. Selain itu, TensorFlow juga memiliki komunitas yang besar dan aktif, sehingga pengguna dapat dengan mudah menemukan sumber daya dan dukungan dalam pengembangan model *machine learning* mereka.

2.11 Keras

Keras adalah antarmuka pemrograman aplikasi (API) berbasis Python yang dirancang untuk mempermudah pembuatan dan pelatihan model jaringan saraf. Keras dikembangkan untuk berjalan di atas kerangka pembelajaran mesin seperti TensorFlow, Theano, atau CNTK. Keras menekankan kemudahan penggunaan dan modularitas, memungkinkan pengguna untuk membangun dan menguji model *deep learning* dengan cepat dan efisien (Chicho et al., 2021).

Keras merupakan sebuah pustaka open source untuk *machine learning* yang dibangun di atas bahasa pemrograman Python. Keras menyediakan antarmuka tingkat tinggi untuk membangun dan melatih model *machine learning* dengan mudah dan cepat. Keras dirancang untuk memudahkan pembuatan model *neural network* dengan menyediakan API yang sederhana dan intuitif. Keras mendukung berbagai jenis model *neural network*, termasuk model *feedforward*, model *recurrent*, dan model *convolutional*.

Salah satu kelebihan Keras adalah fleksibilitasnya dalam menggunakan

backend untuk komputasi numerik. Keras dapat digunakan dengan beberapa *backend*, termasuk TensorFlow, Theano, dan Microsoft Cognitive Toolkit. Keras juga menyediakan berbagai algoritma *machine learning* yang siap digunakan, termasuk optimasi, regularisasi, dan fungsi aktivasi. Selain itu, Keras juga menyediakan berbagai utilitas untuk memproses data dan mengelola model, seperti operasi matriks dan fungsi pembagian data.

Keras telah menjadi salah satu pustaka *machine learning* paling populer di dunia, karena mudah digunakan, cepat, dan fleksibel. Keras juga memiliki komunitas yang besar dan aktif, sehingga pengguna dapat dengan mudah menemukan sumber daya dan dukungan dalam pengembangan model *machine learning* mereka. Sekarang, Keras telah diintegrasikan ke dalam TensorFlow sebagai bagian dari proyek TensorFlow.

2.12 Scikit-learn

Scikit-learn merupakan sebuah *library open-source* yang digunakan untuk *machine learning* pada bahasa pemrograman Python. *Library* ini menyediakan berbagai algoritma *machine learning*, seperti klasifikasi, regresi, pengelompokan, dan pengurutan data. Selain itu, scikit-learn juga dilengkapi dengan berbagai fitur untuk memproses dan memanipulasi data, seperti *preprocessing*, *feature selection*, dan *data transformation*.

Scikit-learn atau sklearn merupakan sebuah module dari bahasa pemrograman Python yang dibangun berdasarkan NumPy, SciPy, dan Matplotlib. Fungsi dari module ini adalah untuk membantu melakukan processing data ataupun melakukan training data untuk kebutuhan machine learning atau data science. Banyak sekali fitur yang ada pada module ini seperti model – model klasifikasi, *clustering*, regresi berbasis model machine learning dan proses – proses yang dapat dimanfaatkan pada tahap *Feature Engineering* seperti reduksi dimensi menggunakan PCA. Module ini sangatlah popular dan sering digunakan dikalangan *data scientist* karena banyak sekali model – model *machine learning* yang dapat kita panggil menggunakan module ini (BISA AI, 2024).

Scikit-learn dirancang untuk menjadi mudah digunakan dan dapat diakses

oleh pengguna dengan berbagai tingkat keahlian dalam machine learning. Library ini memiliki dokumentasi yang lengkap dan tutorial yang tersedia di website resminya sehingga memudahkan pengguna untuk mempelajari dan mengimplementasikan algoritma machine learning. Beberapa fitur dari scikit-learn antara lain:

1. **Algoritma machine learning yang lengkap**

Scikit-learn menyediakan beragam algoritma machine learning, mulai dari algoritma yang sederhana hingga yang kompleks, seperti *Decision Tree*, *Random Forest*, *Naive Bayes*, *Support Vector Machine*, dan *Neural Network*.

2. **Preprocessing Data**

Scikit-learn dilengkapi dengan berbagai fitur untuk memproses dan memanipulasi data, seperti *handling missing values*, *scaling data*, *encoding* kategorikal data, dan lain-lain.

3. **Feature Selection**

Scikit-learn menyediakan berbagai teknik untuk memilih fitur yang paling relevan dalam dataset, seperti *SelectKBest*, *Principal Component Analysis* (PCA), dan *Recursive Feature Elimination* (RFE).

4. **Evaluasi Model**

Scikit-learn memiliki berbagai metrik evaluasi untuk mengukur performa model, seperti *accuracy*, *precision*, *recall*, *F-1 Score*, dan lain-lain.

Scikit-learn telah digunakan dalam berbagai aplikasi dalam bidang *machine learning*, seperti klasifikasi spam email, prediksi harga saham, deteksi penyakit, dan lain-lain. Library ini juga dapat digunakan bersamaan dengan library Python lainnya, seperti NumPy, Pandas, dan Matplotlib untuk memproses dan memvisualisasikan data secara efektif.

2.13 Adam *Optimizer*

Optimizer adalah metode atau algoritma yang digunakan untuk memecahkan masalah optimisasi dalam *machine learning*. Tujuan utamanya adalah untuk menemukan parameter model yang meminimalkan atau memaksimalkan fungsi objektif. *Optimizer* berperan penting dalam proses pembelajaran mesin, di mana esensi dari sebagian besar algoritma *machine learning* adalah membangun model optimisasi dan mempelajari parameter dalam fungsi objektif dari data yang diberikan (Sun et al., 2019).

Optimizer adalah sebuah algoritma yang berfungsi untuk meminimalkan fungsi kesalahan atau untuk memaksimalkan efisiensi produksi. *Optimizer* biasanya berupa fungsi matematika yang bergantung pada parameter model yang dapat dipelajari seerti bobot dan bias. *Optimizer* dapat membantu proses pelatihan suatu model *deep learning* dengan cara mengubah bobot dan bias untuk mengurangi kerugian.

Adam *optimizer* adalah algoritma optimasi yang sering digunakan dalam *machine learning* untuk mengoptimalkan model *neural network*. Adam merupakan singkatan dari *“adaptive moment estimation”*, yang merujuk pada kemampuan algoritma ini untuk menyesuaikan laju pembelajaran (*learning rate*) untuk setiap parameter dalam model secara adaptif.

Adam *optimizer* adalah algoritma optimasi yang banyak digunakan dalam pelatihan jaringan saraf dalam. Algoritma ini menggabungkan keuntungan dari dua metode optimasi sebelumnya, yaitu AdaGrad dan RMSProp, dengan melacak rata-rata momentum pertama (mean) dan kedua (variance) dari gradien. Adam mengadaptasi laju pembelajaran untuk setiap parameter, membuatnya sangat efektif dalam mengatasi masalah gradien yang jarang (Bock et al., 2019).

Secara umum, Adam *optimizer* memperhitungkan gradien dari seluruh data latih saat memperbarui parameter, dan menghitung laju pembelajaran adaptif untuk setiap parameter dalam model. Hal ini memungkinkan Adam *optimizer* untuk menyesuaikan laju pembelajaran secara adaptif untuk setiap parameter, tergantung pada kondisi lokal dari gradien.

Adam *optimizer* telah terbukti efektif dalam mempercepat konvergensi dalam

pelatihan model *neural network*, terutama pada masalah yang kompleks dan data yang besar. Selain itu, Adam *optimizer* juga mudah diimplementasikan dan digunakan dalam berbagai jenis model *neural network*.

2.14 Transfer Learning

Transfer learning merupakan sebuah teknik yang memanfaatkan representasi fitur dari model yang telah dilatih sebelumnya sehingga saat melatih model baru, tidak perlu melatih model tersebut dari awal. Model pra-pelatihan umumnya dilatih pada dataset yang besar yang menjadi patokan standar dalam bidang visi komputer. Bobot yang dihasilkan dari model yang telah dilatih dapat diterapkan pada model baru untuk menyelesaikan masalah visi komputer dan lainnya. Model pra-pelatihan dapat langsung digunakan dalam membuat prediksi tugas baru atau diintegrasikan ke dalam pelatihan model baru. Penggunaan model pra-pelatihan dalam model baru dapat menghasilkan waktu pelatihan yang lebih cepat dan kesalahan generalisasi yang lebih rendah. *Transfer learning* sangat bermanfaat ketika data pelatihan terbatas. Bobot yang terdapat pada model pra-pelatihan dapat digunakan untuk memprediksi fitur-fitur pada data latih sehingga pelatihan dapat dilakukan dalam waktu yang lebih singkat (Zhuang et al., 2020). Terdapat 5 tahap dalam implementasi transfer learning, yaitu:

1. Menentukan model pra-pelatihan

Langkah pertama adalah menentukan model pra-pelatihan yang ingin digunakan. Pemilihan model pra-pelatihan dapat dilakukan dengan cara melihat performa model tersebut pada dataset yang besar.

2. Membuat model dasar

Proses ini dilakukan dengan cara mengimplementasikan arsitektur dari model pra-pelatihan seperti EfficientNetV2. Proses ini dilakukan dengan cara mengunduh bobot pra-pelatihan secara opsional, hal ini dilakukan agar model dasar yang dibuat memiliki dasar pengetahuan untuk kasus tertentu. Pada umumnya model dasar akan memiliki lebih banyak unit di lapisan keluaran

akhir daripada yang dibutuhkan. Saat membuat model dasar, diharuskan untuk menghapus lapisan keluaran akhir. Selanjutnya adalah menambahkan lapisan keluaran akhir yang kompatibel dengan masalah tertentu.

3. *Freezing Layer*

Freezing layer dilakukan pada model dasar. Hal ini bertujuan agar bobot pada lapisan model dasar tidak diinisialisasi ulang pada saat proses pelatihan. Jika tidak melakukan *freezing layer*, maka beban yang sebelumnya telah diunduh akan hilang secara keseluruhan dan pelatihan model dilakukan dari awal yang dapat memakan waktu lebih lama.

4. Melatih layer baru untuk dataset baru

Layer baru adalah layer yang dibuat untuk menyesuaikan layer output dengan dataset baru tanpa mengubah layer yang telah dibekukan (*freeze*) sebelumnya. Hal ini dilakukan karena layer output dari model pra-pelatihan umumnya telah disesuaikan dengan kebutuhan dataset besar yang dilatih sebelumnya, seperti ImageNet yang memiliki 1000 kelas. Tentu ini tidak selalu relevan dengan dataset baru yang digunakan, sebagai contoh apabila dataset yang digunakan untuk melatih model baru hanya memiliki 2 kelas, maka layer output ini tidak sesuai. Oleh karena itu layer output harus disesuaikan dengan kebutuhan dataset baru.

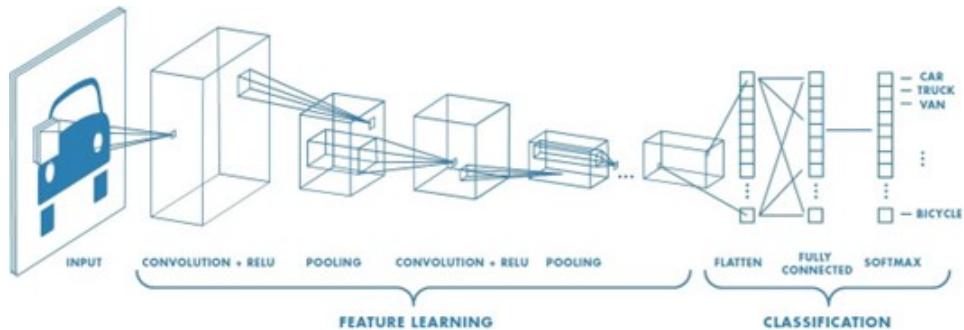
5. Meningkatkan performa model dengan *fine-tuning*

Performa suatu model *machine learning* tentu tidaklah selalu baik untuk semua tipe dataset. Hal ini juga yang dapat menyebabkan model pra-pelatihan kurang cocok digunakan untuk suatu tipe dataset tertentu. *Fine-tuning* adalah sebuah metode untuk mengubah parameter yang ada pada model tertentu pada saat proses pelatihan dan melakukan proses pelatihan ulang. Salah satu parameter yang berpengaruh pada saat proses pelatihan adalah *learning rate*. Pemilihan nilai *learning rate* yang tepat akan membuat performa model meningkat dan dapat memprediksi tugas sesuai yang diinginkan.

2.15 Convolutional Neural Network

Convolutional Neural Network atau dapat disingkat dengan CNN merupakan salah satu jenis deep learning. *Machine learning* memiliki cabang yaitu *deep learning* yang berguna untuk membuat komputer dapat melakukan pekerjaan seperti pekerjaan manusia. *Convolutional Neural Network* bekerja dengan meniru berkomunikasi dengan neuron yang cara kerja dari saraf-saraf manusia berhubungan.

Sebagai algoritma dengan kinerja yang sangat baik, CNN telah banyak digunakan di bidang pemrosesan citra dan mencapai hasil yang baik dengan mengandalkan bidang reseptif lokalnya sendiri, pembagian bobot, penggabungan, dan koneksi yang terpisah (Tian, 2020).



Gambar 2.6: Arsitektur Convolutional Neural Network (CNN)

Pada Gambar 2.6, tahap pertama pada arsitektur *Convolutional Neural Network* yaitu tahap Konvolusi, kemudian dilanjutkan menuju fungsi aktivasi yang biasanya menggunakan fungsi aktivitas *Rectifier Linear Unit* atau ReLU. Proses *Pooling* merupakan proses setelah fungsi aktivasi yang dimana proses ini diulang beberapa kali sampai mendapatkan peta fitur agar dapat melanjutkan ke *fully connected neural network*.

Setelah *fully connected neural network* akan dilanjutkan ke *output class*. *Convolutional layer* merupakan bagian dari tahap pada arsitektur CNN. Tahap ini melakukan operasi konvolusi pada output dari layer sebelumnya. Layer tersebut merupakan proses utama yang mendasari jaringan arsitektur CNN. Konvolusi merupakan istilah matematis dimana pengaplikasian sebuah fungsi pada output fungsi lain secara berulang. Operasi konvolusi merupakan operasi pada 2 fungsi argumen bernilai nyata. Operasi ini menerapkan fungsi output sebagai feature map

dari input citra. Input dan output ini dapat dilihat sebagai 2 argumen bermilai real.

2.16 Lapisan *Convolutional Neural Network* (CNN)

Convolutional Neural Network (CNN) terdiri atas tiga lapis (layer) yaitu *input layer*, *output layer*, dan beberapa *hidden layers*. *Hidden layer* umumnya berisi *convolutional layers*, *pooling layers*, *normalization layers*, ReLu layer, *fully connected layers*, dan serta *loss layer* (Alom et al., 2018).

2.16.1 *Convolutional Layer*

Lapisan konvolusi, atau *Convolutional Layer*, merupakan komponen utama dalam jaringan saraf konvolusional (CNN) yang berfungsi untuk mengekstraksi fitur-fitur dari input citra. Lapisan ini menggunakan filter atau kernel yang bergerak melintasi citra untuk melakukan operasi konvolusi, menghasilkan peta fitur (*feature map*) sebagai output. Setiap filter dalam lapisan ini belajar mendeteksi pola-pola spesifik, seperti tepi, tekstur, atau warna, dengan menerapkan bobot tertentu pada piksel dalam citra input. Hasil dari operasi konvolusi ini membantu jaringan dalam mengenali dan memahami elemen-elemen penting dari citra yang diolah. Dengan demikian, lapisan konvolusi adalah kunci dalam mengubah data citra mentah menjadi representasi fitur yang dapat digunakan oleh lapisan-lapisan berikutnya dalam jaringan untuk tugas-tugas seperti klasifikasi dan deteksi objek (Goodfellow et al., 2016).

2.16.2 *Activation Layer*

Lapisan aktivasi, atau *Activation Layer*, adalah bagian penting dari jaringan saraf konvolusional (CNN) yang memperkenalkan non-linearitas ke dalam model. Setelah operasi konvolusi menghasilkan peta fitur, lapisan ini menerapkan fungsi aktivasi seperti ReLU (Rectified Linear Unit) yang mengubah semua nilai negatif menjadi nol dan mempertahankan nilai positif. Fungsi aktivasi lainnya yang sering digunakan termasuk Sigmoid dan Tanh, meskipun ReLU lebih umum dalam CNN karena sifatnya yang mempercepat konvergensi. Dengan menambahkan non-linearitas, lapisan aktivasi memungkinkan model untuk belajar dari data yang lebih kompleks

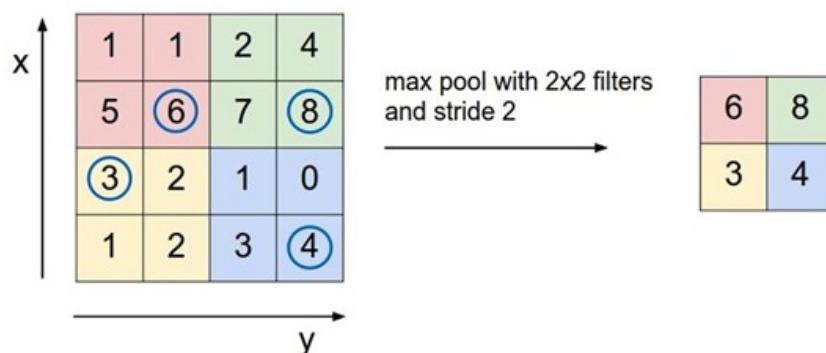
dan meningkatkan kemampuannya dalam menyelesaikan masalah klasifikasi dan deteksi objek (Goodfellow et al., 2016).

2.16.3 Pooling Layer

Lapisan *pooling*, atau *Pooling Layer*, digunakan untuk mengurangi dimensi spasial dari peta fitur yang dihasilkan oleh lapisan konvolusi, yang membantu mengurangi jumlah parameter dan komputasi dalam jaringan. Dua jenis pooling yang umum digunakan adalah *Max Pooling* dan *Average Pooling*. *Max Pooling* mengambil nilai maksimum dari setiap patch kecil peta fitur, sementara *Average Pooling* mengambil rata-rata nilai. Proses *pooling* membantu membuat representasi fitur lebih tahan terhadap pergeseran dan distorsi kecil dalam gambar, sehingga meningkatkan ketahanan model terhadap variasi pada data input (Goodfellow et al., 2016).

1. Max Pooling

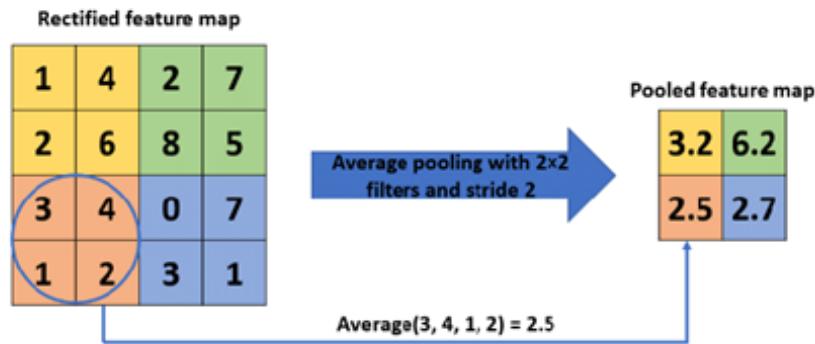
Max pooling merupakan metode *pooling* yang paling populer. Operasi ini dilakukan dengan cara mengambil neuron dengan nilai aktivasi tertinggi di setiap bidang reseptif lokal (*grid cell*), dan hanya mengambil nilai tertinggi dan mengirim nilai itu ke proses selanjutnya. Pada Gambar 2.7 merupakan contoh max pooling, dimana bidang yang berjumlah 4x4, diperkecil menjadi bidang reseptif 2x2 dengan cara mengambil nilai yang terbesar (Vasilev et al., 2019).



Gambar 2.7: Proses Max Pooling

2. Average Pooling

Average pooling merupakan metode *pooling* lainnya, di mana output dari masing-masing bidang reseptif adalah nilai rata-rata dari semua aktivasi dalam bidang tersebut. Pada Gambar 2.8 merupakan contoh *average pooling* dengan cara mengambil nilai rata-rata untuk memperkecil ukuran (Vasilev et al., 2019).



Gambar 2.8: Proses Average Pooling

2.16.4 Fully Connected Layer

Lapisan fully connected, atau *Fully Connected Layer*, biasanya ditempatkan di bagian akhir CNN, di mana setiap neuron terhubung dengan semua neuron di lapisan sebelumnya. Lapisan ini berfungsi untuk menggabungkan fitur-fitur yang diekstraksi oleh lapisan konvolusi dan *pooling* untuk melakukan klasifikasi akhir. Data yang diterima oleh lapisan ini diubah menjadi satu dimensi dan kemudian diproses oleh sejumlah neuron yang dapat memprediksi kelas dari input. Lapisan *fully connected* ini sering kali diikuti oleh fungsi aktivasi seperti Softmax (untuk masalah klasifikasi multi-kelas) atau Sigmoid (untuk masalah klasifikasi biner) untuk menghasilkan probabilitas dari kelas-kelas yang diprediksi (Goodfellow et al., 2016).

2.16.5 Normalization Layer

Lapisan normalisasi, atau *Normalization Layer*, seperti Batch Normalization, digunakan untuk menormalkan input dari satu lapisan sebelum memasuki lapisan berikutnya. Normalisasi dilakukan dengan cara menstandarkan aktivasi dari layer

sebelumnya untuk setiap *mini-batch*, dengan mempertahankan rata-rata dan variansi tertentu. Proses ini membantu dalam mempercepat proses pelatihan dan membuat model lebih stabil dengan mengurangi masalah *vanishing gradients* dan *exploding gradients*, yang umum terjadi dalam jaringan saraf yang dalam (Ioffe et al., 2015).

2.16.6 *Output Layer*

Lapisan output, atau *Output Layer*, adalah lapisan terakhir dalam CNN yang memberikan hasil akhir dari jaringan. Untuk masalah klasifikasi, lapisan ini biasanya terdiri dari neuron sebanyak jumlah kelas yang ada dalam data dan menggunakan fungsi aktivasi seperti Softmax untuk mengubah skor menjadi probabilitas. Probabilitas ini kemudian digunakan untuk menentukan kelas dari input citra. Fungsi aktivasi Softmax membantu dalam menghasilkan distribusi probabilitas dari berbagai kelas, sehingga memudahkan dalam penentuan keputusan akhir (Goodfellow et al., 2016).

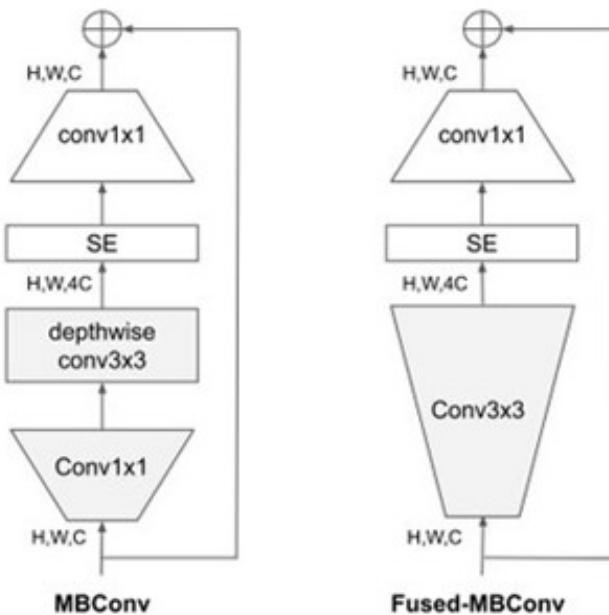
2.17 EfficientNet

EfficientNet merupakan arsitektur *Convolutional Neural Network* atau CNN dan metode penskalaan yang secara seragam menskala semua dimensi kedalaman (*depth*), lebar (*width*), dan resolusi menggunakan koefisien gabungan (*compound coefficient*). Metode penskalaan EfficientNet didasarkan pada intuisi bahwa jika gambar input lebih besar, maka jaringan membutuhkan lebih banyak lapisan untuk meningkatkan bidang reseptif (*receptive field*) dan lebih banyak saluran untuk menangkap pola yang lebih detail pada gambar yang lebih besar.

EfficientNet telah menunjukkan performa yang sangat baik dalam berbagai tugas *transfer learning*, mencapai akurasi *state-of-the-art* pada dataset seperti CIFAR-100 (91,7%), Flowers (98,8%), dan tiga dataset *transfer learning* lainnya, dengan jumlah parameter yang jauh lebih sedikit dibandingkan model-model sebelumnya (Tan et al., 2020).

2.17.1 EfficientNetV2

Model EfficientNetV2 merupakan keluarga baru dari *Convolutional Neural Network* (CNN) yang memiliki kemampuan unggul dalam pengklasifikasian gambar, karena mampu 11x lebih cepat dalam pelatihan dan model yang 6.8x lebih kecil. EfficientNetV2 menggunakan blok *Mobile Inverted Bottleneck Convolution* (MBConv) dengan rasio ekspansi yang lebih kecil dan Fused-MBConv yang ditambahkan pada lapisan awal serta menggunakan kernel berukuran 3x3 yang lebih kecil dengan beberapa layer (Tan et al., 2021).



Gambar 2.9: Struktur MBConv dan Fused-MBConv

Gambar 2.9 menunjukkan struktur dari MBConv dan Fused- MBConv. Blok Mobile Inverted Bottleneck Convolution (MBConv) dan Fused-MBConv pada EfficientNetV2 memiliki sedikit perbedaan layer yang digunakan. MBConv menggunakan depthwise conv3x3 dan Conv1x1, sedangkan Fused-MBConv menggunakan Conv3x3.

Gambar 2.10 menunjukkan blok-blok arsitektur EfficientNetV2-S. Blok-blok arsitektur EfficientNetV2 pada Gambar 2.10 terdiri dari layer Conv3x3 (layer konvolusi dengan kernel berukuran 3x3), Fused-MBConv k3x3 (kernel berukuran 3x3) dengan rasio ekspansi 1, dan 4, MBConv k3x3 (kernel berukuran 3x3) dengan

Stage	Operator	Stride	#Channels	#Layers
0	Conv3x3	2	24	1
1	Fused-MBConv1, k3x3	1	24	2
2	Fused-MBConv4, k3x3	2	48	4
3	Fused-MBConv4, k3x3	2	64	4
4	MBConv4, k3x3, SE0.25	2	128	6
5	MBConv6, k3x3, SE0.25	1	160	9
6	MBConv6, k3x3, SE0.25	2	272	15
7	Conv1x1 & Pooling & FC	-	1792	1

Gambar 2.10: Arsitektur EfficientNetV2-S (Tan et al. 2021)

rasio ekspansi 4, dan 6, dan Conv1x1 (layer konvolusi dengan kernel berukuran 1x1) & Pooling & FC layer. Total layer yang digunakan pada Gambar 2.10 berjumlah 42 layer.

2.18 Grad-CAM

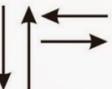
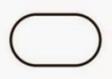
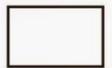
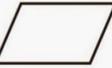
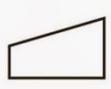
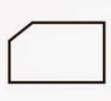
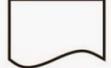


Gambar 2.11: Contoh Visualisasi Grad-CAM

Grad-CAM atau *Gradient-weighted Class Activation Mapping* merupakan teknik visualisasi yang menghasilkan penjelasan visual untuk keputusan yang dibuat oleh model *Convolutional Neural Network*. Metode ini menggunakan gradien dari skor kelas target yang mengalir ke lapisan konvolusional terakhir untuk menghasilkan *heatmap* yang menyoroti area penting dalam gambar yang mempengaruhi prediksi model. Grad-CAM merupakan generalisasi dari teknik *Class Activation Mapping* (CAM), namun dapat diterapkan pada berbagai arsitektur CNN tanpa perlu memodifikasi atau melatih ulang model. Grad-CAM dapat digunakan untuk berbagai tugas seperti *weakly-supervised localization*, segmentasi,

dan memahami kegagalan serta bias model. Teknik ini membantu meningkatkan transparansi dan interpretabilitas model CNN dengan menunjukkan area mana dalam gambar yang paling berpengaruh terhadap keputusan klasifikasi (Selvaraju et al., 2017).

2.19 Flowchart

	Flow Direction symbol Yaitu simbol yang digunakan untuk menghubungkan antara simbol yang satu dengan simbol yang lain. Simbol ini disebut juga connecting line.
	Terminator Symbol Yaitu simbol untuk permulaan (start) atau akhir (stop) dari suatu kegiatan
	Connector Symbol Yaitu simbol untuk keluar - masuk atau penyambungan proses dalam lembar / halaman yang sama.
	Connector Symbol Yaitu simbol untuk keluar - masuk atau penyambungan proses pada lembar / halaman yang berbeda.
	Processing Symbol Simbol yang menunjukkan pengolahan yang dilakukan oleh komputer
	Simbol Manual Operation Simbol yang menunjukkan pengolahan yang tidak dilakukan oleh computer
	Simbol Decision Simbol pemilihan proses berdasarkan kondisi yang ada.
	Simbol Input-Output Simbol yang menyatakan proses input dan output tanpa tergantung dengan jenis peralatannya
	Simbol Manual Input Simbol untuk pemasukan data secara manual on-line keyboard
	Simbol Preparation Simbol untuk mempersiapkan penyimpanan yang akan digunakan sebagai tempat pengolahan di dalam storage.
	Simbol Predefine Proses Simbol untuk pelaksanaan suatu bagian (sub-program)/prosedure
	Simbol Display Simbol yang menyatakan peralatan output yang digunakan yaitu layar, plotter, printer dan sebagainya.
	Simbol disk and On-line Storage Simbol yang menyatakan input yang berasal dari disk atau disimpan ke disk.
	Simbol magnetik tape Unit Simbol yang menyatakan input berasal dari pita magnetik atau output disimpan ke pita magnetik.
	Simbol Punch Card Simbol yang menyatakan bahwa input berasal dari kartu atau output ditulis ke kartu
	Simbol Dokumen Simbol yang menyatakan input berasal dari dokumen dalam bentuk kertas atau output dicetak ke kertas.

Gambar 2.12: Simbol-Simbol Flowchart

Menurut (cmlabs, 2023), *Flowchart* adalah suatu bagan dengan simbol-simbol tertentu yang menggambarkan urutan proses secara mendetail dan hubungan antara suatu proses (instruksi) dengan proses lainnya dalam suatu program. *Flowchart* adalah suatu bagan dengan simbol-simbol tertentu yang menggambarkan urutan proses secara mendetail dan hubungan antara suatu proses (instruksi) dengan proses lainnya dalam suatu program. Pada perancangan *flowchart* sebenarnya tidak ada

rumus atau patokan yang bersifat mutlak (pasti). Hal ini didasari oleh *flowchart* (bagan alir) adalah sebuah gambaran dari hasil pemikiran dalam menganalisa suatu permasalahan dalam komputer, karena setiap analisa akan menghasilkan hasil yang bervariasi antara satu dan lainnya. Secara garis besar setiap perancangan *flowchart* selalu terdiri dari tiga bagian, yaitu input, proses, dan output. *Flowchart* memiliki simbol-simbol tersendiri dari setiap anotasi-anotasi geometri yang digunakan.

2.20 Confusion Matrix

		Predicted Class		
		Positive	Negative	
Actual Class	Positive	True Positive (TP)	False Negative (FN) Type II Error	Sensitivity $\frac{TP}{(TP + FN)}$
	Negative	False Positive (FP) Type I Error	True Negative (TN)	Specificity $\frac{TN}{(TN + FP)}$
		Precision $\frac{TP}{(TP + FP)}$	Negative Predictive Value $\frac{TN}{(TN + FN)}$	Accuracy $\frac{TP + TN}{(TP + TN + FP + FN)}$

Gambar 2.13: Confusion Matrix

Confusion matrix adalah alat evaluasi yang digunakan dalam klasifikasi statistik untuk menilai kinerja suatu model klasifikasi. Matriks ini memperlihatkan perbandingan antara hasil prediksi model dan hasil aktual dari data yang diuji. *Confusion matrix* berfungsi untuk memberikan gambaran detail tentang bagaimana model melakukan klasifikasi, dengan mencakup empat komponen utama: True Positive (TP), True Negative (TN), False Positive (FP), dan False Negative (FN).

- **True Positive (TP):** Jumlah kasus di mana model memprediksi kelas positif dengan benar.
- **True Negative (TN):** Jumlah kasus di mana model memprediksi kelas negatif dengan benar.

- **False Positive (FP):** Jumlah kasus di mana model salah memprediksi kelas positif (Type I error).
- **False Negative (FN):** Jumlah kasus di mana model salah memprediksi kelas negatif (Type II error).

2.21 Jupyter Notebook

Jupyter Notebook adalah aplikasi web open-source yang memungkinkan pengguna untuk membuat dan berbagi dokumen yang mengandung kode langsung, visualisasi, dan teks penjelasan. Jupyter Notebook sangat populer di kalangan ilmuwan data, peneliti, dan pengembang karena kemampuannya untuk mendukung analisis data interaktif dan komputasi ilmiah dalam berbagai bahasa pemrograman seperti Python, R, dan Julia. (Jupyter, 2021) Fitur Utama Jupyter Notebook:

1. **Interaktif dan Real-Time:** Pengguna dapat menulis dan mengeksekusi kode langsung di dalam notebook, melihat hasilnya secara langsung. Hal ini memungkinkan eksperimen yang cepat dan interaktif.
2. **Rich Media Integration:** Selain kode dan teks, Jupyter Notebook dapat menyertakan visualisasi data, gambar, video, dan bahkan widget interaktif, yang membantu dalam presentasi data yang lebih komprehensif dan mudah dipahami.
3. **Reproducible Research:** Notebook dapat dibagikan dengan orang lain dan dieksekusi ulang untuk mendapatkan hasil yang sama, mendukung praktik penelitian yang dapat direproduksi.
4. **Markdown Support:** Pengguna dapat menggunakan Markdown untuk menambahkan teks format, header, daftar, link, dan banyak lagi, membuat dokumen lebih rapi dan mudah dibaca.
5. **Versi dan Kolaborasi:** Notebook disimpan dalam format file .ipynb yang dapat dengan mudah dipantau versi dan kolaborasi menggunakan sistem kontrol versi seperti Git.

6. **Ekosistem Luas:** Jupyter mendukung berbagai jenis kernel untuk bahasa pemrograman yang berbeda, memperluas kemampuannya di luar Python saja.

Contoh Penggunaan Jupyter Notebook:

1. **Data Analysis:** Analisis dataset besar menggunakan pustaka seperti Pandas, NumPy, dan SciPy.
2. **Machine Learning:** Pembangunan dan pelatihan model machine learning menggunakan pustaka seperti scikit-learn, TensorFlow, dan PyTorch.
3. **Visualization:** Membuat visualisasi data yang kompleks menggunakan Matplotlib, Seaborn, dan Plotly.
4. **Education:** Digunakan sebagai alat pembelajaran interaktif untuk mengajarkan pemrograman dan konsep ilmu data.

2.22 Git

Git adalah sistem kontrol versi terdistribusi yang dirancang untuk melacak perubahan pada file dan mengoordinasikan pekerjaan di antara beberapa orang. Git awalnya dikembangkan oleh Linus Torvalds pada tahun 2005 untuk mendukung pengembangan kernel Linux. Sistem ini sangat populer di kalangan pengembang perangkat lunak karena kemampuannya untuk menangani proyek besar dengan kecepatan dan efisiensi yang tinggi (Git, 2024). Fitur Utama Git:

1. **Distribusi:** Setiap klon dari repositori Git adalah repositori lengkap dengan riwayat lengkap dan kemampuan pelacakan versi penuh. Hal ini berarti setiap pengembang dapat bekerja secara offline dan melakukan sinkronisasi perubahan saat kembali online.
2. **Branching dan Merging:** Git memungkinkan pembuatan cabang (branch) yang mudah dan efisien. Branching memungkinkan pengembang untuk bekerja pada fitur baru atau perbaikan bug secara terpisah dari cabang utama (main branch). Setelah selesai, cabang dapat digabungkan (merge) kembali ke cabang utama.

3. **Kecepatan dan Kinerja:** Git dirancang untuk efisiensi dalam hal kecepatan dan performa. Operasi lokal seperti commit, diff, dan log sangat cepat karena tidak memerlukan komunikasi dengan server.
4. **Keamanan:** Git menggunakan hashing SHA-1 untuk memastikan integritas konten dan riwayat perubahan, membuatnya sangat sulit untuk memanipulasi data tanpa terdeteksi.
5. **Kolaborasi:** Git mendukung berbagai alur kerja kolaboratif, termasuk pengembangan terpusat, alur kerja GitHub, dan pengembangan berbasis cabang.

2.23 *Image Normalization*

Image normalization adalah teknik yang digunakan dalam pemrosesan gambar dan pembelajaran mesin untuk menstandardisasi skala nilai piksel dalam gambar. Normalisasi gambar bertujuan untuk meningkatkan konvergensi dan performa model dengan memastikan bahwa data input berada dalam rentang yang seragam dan memiliki distribusi yang lebih stabil. Ini sangat penting dalam jaringan saraf konvolusional (CNN) dan metode pembelajaran mesin lainnya, di mana skala nilai yang tidak konsisten dapat mempengaruhi pembelajaran model (Goodfellow et al., 2016).

Keuntungan *Image Normalization*:

1. **Konsistensi:** Membuat data lebih konsisten dan seragam, yang membantu model belajar dengan lebih efisien.
2. **Konvergensi Lebih Cepat:** Membantu model neural network untuk berkonvergensi lebih cepat selama pelatihan.
3. **Mengurangi Variabilitas:** Mengurangi efek variabilitas pencahayaan dan kontras dalam gambar.

2.24 *Web Scraping*

Web scraping adalah teknik untuk mengekstrak data dari situs web secara otomatis menggunakan program atau bot. Proses ini melibatkan pengiriman permintaan ke server web, mengambil halaman web, dan kemudian menganalisis konten halaman tersebut untuk mengekstrak informasi yang diinginkan. *Web scraping* sering digunakan untuk mengumpulkan data dari internet dalam jumlah besar, yang kemudian dapat dianalisis atau digunakan untuk berbagai aplikasi (Mitchell, 2018). Google Image Scraper adalah contoh khusus dari web scraping yang digunakan untuk mengekstrak gambar dari hasil pencarian Google Images. *Scraper* ini mengirimkan permintaan pencarian ke Google Images dan kemudian mengunduh gambar yang muncul dalam hasil pencarian.

2.25 *Good fit, Overfit, Underfit*

2.25.1 *Good fit*

Good fit terjadi ketika model statistik atau machine learning berhasil menangkap pola yang ada dalam data pelatihan dan mampu membuat prediksi yang akurat pada data baru (tidak terlihat). Model yang memiliki good fit menunjukkan keseimbangan yang baik antara bias dan variansi, artinya model tersebut cukup kompleks untuk menangkap pola data tetapi tidak terlalu kompleks sehingga mempelajari noise dalam data (Goodfellow et al., 2016).

2.25.2 *Overfit*

Overfitting terjadi ketika model terlalu kompleks dan mulai belajar *noise* atau variasi acak dalam data pelatihan sebagai informasi yang sebenarnya. Hal ini menyebabkan model memiliki performa yang sangat baik pada data pelatihan tetapi buruk pada data baru atau tidak terlihat. Model *overfit* memiliki variansi yang tinggi dan bias yang rendah. *Overfitting* dapat diidentifikasi melalui kinerja yang jauh lebih baik pada data pelatihan dibandingkan data validasi atau pengujian (Goodfellow et al., 2016).

Menurut (Desai, 2019), model dikatakan mengalami *overfit* jika dilatih secara berlebihan pada data sehingga model tersebut bahkan mempelajari *noise* dari data tersebut. Model *overfit* mempelajari setiap contoh dengan sangat sempurna sehingga salah mengklasifikasikan contoh yang tidak terlihat atau baru. Untuk model yang mengalami *overfit*, memiliki skor set pelatihan yang sempurna atau mendekati sempurna, sementara skor pengujian atau validasi buruk.

2.25.3 Underfit

Underfitting terjadi ketika model terlalu sederhana dan gagal menangkap pola dasar dalam data pelatihan. Ini berarti model tidak memiliki cukup kapasitas untuk memahami hubungan yang mendasari data, yang menyebabkan performa buruk baik pada data pelatihan maupun data baru. Model *underfit* memiliki bias yang tinggi dan variansi yang rendah. (Goodfellow et al., 2016)

Menurut (Desai, 2019), Model dikatakan mengalami underfit jika tidak mampu mempelajari pola dalam data dengan baik. Model underfit tidak sepenuhnya mempelajari setiap contoh dalam dataset. Dalam kasus seperti ini, dapat dilihat skor yang rendah pada set pelatihan maupun set pengujian atau validasi.

2.26 Mesin DGX

Mesin DGX adalah platform komputasi yang dirancang oleh NVIDIA untuk mendukung aplikasi kecerdasan buatan (AI) dan pembelajaran mendalam (*deep learning*). Mesin DGX menyediakan infrastruktur yang kuat dan efisien untuk melakukan pelatihan model AI yang kompleks, memungkinkan peneliti dan praktisi untuk mempercepat pengembangan dan implementasi solusi AI.

DGX dilengkapi dengan GPU NVIDIA yang kuat, memori besar, dan perangkat lunak yang dioptimalkan untuk AI, seperti NVIDIA CUDA dan cuDNN. Ini menjadikan DGX sebagai pilihan utama untuk berbagai aplikasi AI, mulai dari pengenalan gambar hingga pemrosesan bahasa alami (Corporation, 2021).

Universitas Gunadarma menghadirkan mesin super komputer Nvidia DGX-1 dan DGX A100 di Indonesia. Mesin super komputer Nvidia DGX-1 dan DGX A100 merupakan mesin dengan kemampuan tinggi dalam melakukan proses dengan jumlah

data besar, mendukung dalam perkembangan *Big Data*, *Data Analytic*, dan *Artificial Intelligence*.

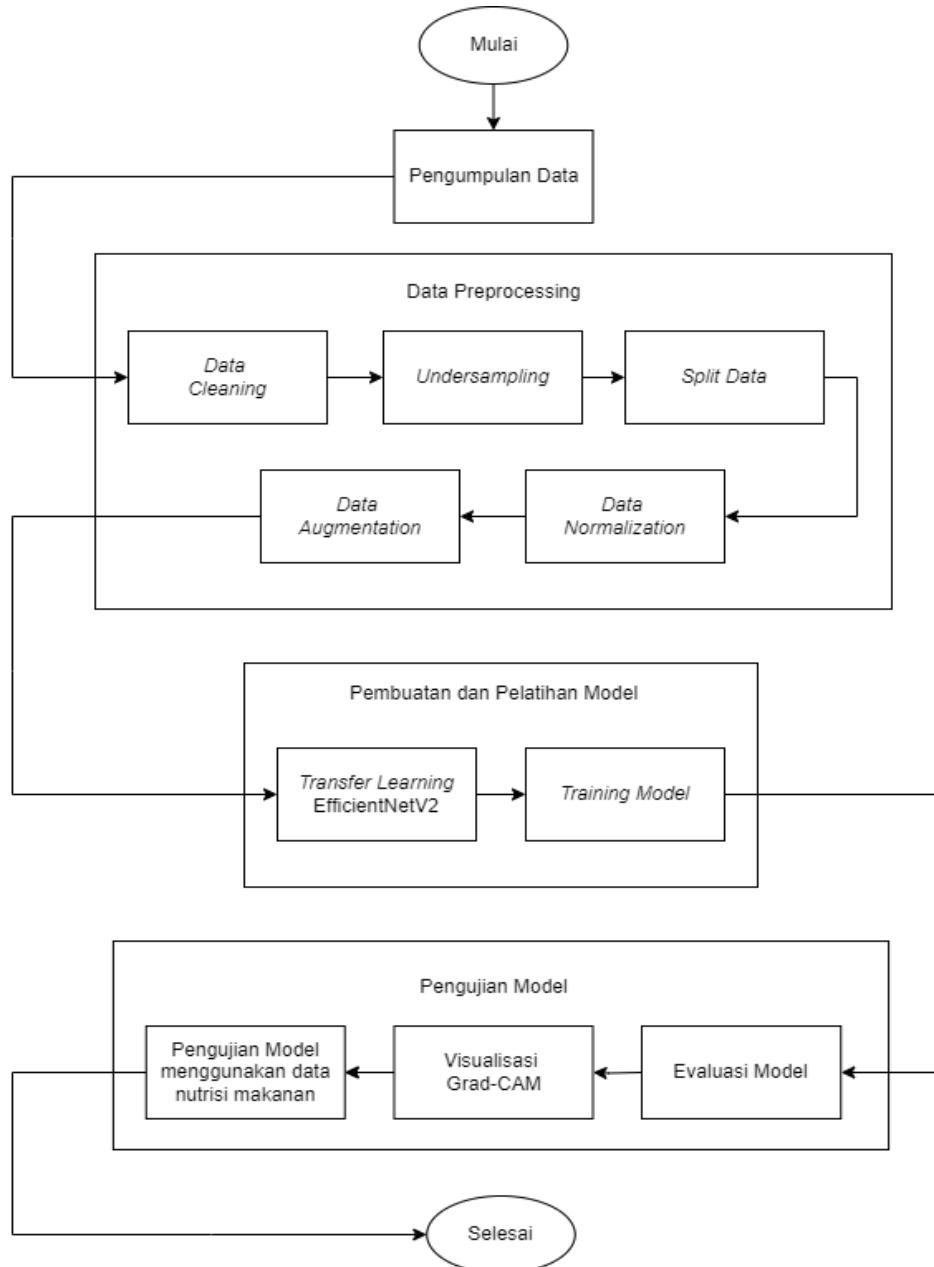
Mesin Nvidia DGX-1 merupakan mesin dengan 8 (delapan) GPU dan masing-masing memiliki 16 GB memory. Mesin DGX A100 merupakan mesin dengan 8 (delapan) GPU dan total 320 GB memory, serta memiliki 6 (enam) Nvidia NVSwitches.

Mesin Nvidia DGX-1 memiliki kemampuan 56 kali lebih baik daripada CPU, dan 75 kali kecepatan dalam melakukan pelatihan. Mesin Nvidia DGX A100 memiliki kemampuan 172 kali lebih baik daripada CPU server, dan juga kemampuan training, inference, serta data analytic (Gunadarma, 2024).

BAB III

METODE PENELITIAN

Metode pada penelitian ini terdiri dari beberapa tahapan proses seperti yang terlihat pada Gambar 3.1.



Gambar 3.1: Flowchart Tahapan Metode Penelitian

Tahapan awal pada penelitian ini adalah pengumpulan data citra makanan

dari internet. Pada penelitian ini, data citra makanan diambil dari *google images* menggunakan teknik *scraping*. Tools yang digunakan dalam penelitian ini yaitu Google Image Scraper. Total dataset citra yang berhasil dikumpulkan sebanyak 5000 citra yang terbagi dalam 10 kelas (ayam bakar, bakso, gado-gado, gudeg, nasi goreng, pempek, rawon, rendang, sate, dan soto) dimana setiap kelas memiliki total 500 citra makanan. Setelah data terkumpul, langkah selanjutnya adalah *preprocessing data* yang melibatkan beberapa tahapan, yaitu pembersihan data (*data cleaning*) untuk memastikan data bebas dari kesalahan dan inkonsistensi, undersampling untuk menangani ketidakseimbangan data, pembagian data (*split data*) menjadi set pelatihan dan set pengujian, augmentasi data (*data augmentation*) untuk memperluas jumlah data pelatihan dengan membuat variasi dari data yang sudah ada, dan normalisasi data (*data normalization*) untuk membuat data menjadi lebih banyak variasi.

Setelah tahap *preprocessing* selesai, proses dilanjutkan dengan pembuatan dan pelatihan model. Pada penelitian ini, pelatihan model akan menggunakan 2 GPU yang berbeda, yang pertama yaitu menggunakan Google Colab dengan GPU T4, kemudian yang kedua menggunakan mesin DGX A100 dari Universitas Gunadarma. Tahapan ini melibatkan *transfer learning* menggunakan EfficientNetV2, sebuah arsitektur jaringan saraf yang telah dilatih sebelumnya pada sejumlah besar data. Model yang dihasilkan dari *transfer learning* kemudian dilatih lebih lanjut menggunakan data yang telah melalui tahap *preprocessing*.

Tahap berikutnya adalah pengujian model. Langkah ini mencakup evaluasi model untuk mengukur kinerjanya, visualisasi Grad-CAM untuk memahami bagaimana model mengambil keputusan dengan memvisualisasikan bagian dari gambar yang diperhatikan oleh model, dan pengujian model akhir menggunakan data nutrisi makanan untuk memastikan model berfungsi dengan baik dalam konteks yang diinginkan.

3.1 Pengumpulan Data

Pada tahapan ini, dataset dikumpulkan melalui internet. Pada penelitian ini, data citra makanan diambil dari *google images* menggunakan teknik *scraping*. Tools yang digunakan dalam penelitian ini yaitu *google image scraper*. Total dataset citra

yang berhasil dikumpulkan sebanyak 2000 citra yang terbagi dalam 10 kelas (ayam bakar, bakso, gado-gado, gudeg, nasi goreng, pempek, rawon, rendang, sate, dan soto) dimana setiap kelas memiliki total 200 citra makanan.

Berikut ini merupakan langkah-langkah yang dilakukan untuk mengumpulkan data citra makanan menggunakan teknik *scraping* menggunakan *Google Image Scraper*:

1. Pastikan komputer sudah terinstall git. lalu lakukan clone menggunakan *command prompt* 'git clone <https://github.com/ohyicong/Google-Image-Scraper>'
2. Lalu lakukan instalasi *dependency* menggunakan *command prompt* 'pip install -r requirements.txt'
3. Buka file 'main.py', lalu ganti kode pada variabel 'search_keys' sesuai kata kunci yang diinginkan. Pada penelitian ini, kata kunci yang digunakan yaitu ayam bakar, bakso, gado gado, gudeg, nasi goreng, pempek, rawon, rendang, sate, dan soto.
4. Kemudian ganti juga kode nilai pada variabel 'number_of_images' sesuai dengan jumlah citra yang diinginkan. Pada penelitian ini, citra yang diinginkan berjumlah 500 citra. Pastikan simpan file main.py setelah dilakukan pengubahan kode.

```
if __name__ == "__main__":
    #Define file path
    webdriver_path = os.path.normpath(os.path.join(os.getcwd(), 'webdriver', webdriver_executable()))
    image_path = os.path.normpath(os.path.join(os.getcwd(), 'photos'))\n\n
    #Add new search key into array ["cat","t-shirt","apple","orange","pear","fish"]
    search_keys = list(set(['ayam bakar', 'bakso', 'gado gado', 'gudeg', 'nasi goreng', 'pempek', 'rawon', 'rendang', 'sate', 'soto']))\n\n
    #Parameters
    number_of_images = 500          # Desired number of images
    headless = True                 # True = No Chrome GUI
    min_resolution = (0, 0)         # Minimum desired image resolution
    max_resolution = (9999, 9999)   # Maximum desired image resolution
    max_missed = 10                 # Max number of failed images before exit
    number_of_workers = 1           # Number of "workers" used
    keep_filenames = False          # Keep original URL image filenames
```

Gambar 3.2: Tampilan kode main.py

5. Buka *command prompt*, lalu jalankan program menggunakan 'python main.py'. Proses *scraping* akan berjalan dalam waktu yang cukup lama.

6. Setelah proses *scraping* selesai, citra akan tersimpan pada folder 'photos'. Hasilnya dapat dilihat pada Gambar 3.3



Gambar 3.3: Hasil scraping menggunakan Google Image Scraper

3.2 Data Preprocessing

Setelah dilakukan pengumpulan dataset, selanjutnya yaitu dilakukan *Data Preprocessing* pada data citra. Ada beberapa tahapan dalam *Data Preprocessing*, yaitu *Data Cleaning*, *Undersampling*, *Split Data*, *Data Augmentation*, dan *Data Normalization*. Dalam proses ini dibutuhkan beberapa *package* Python yang harus diinstalasi dan dipanggil saat program dijalankan di Jupyter Notebooks, yaitu:

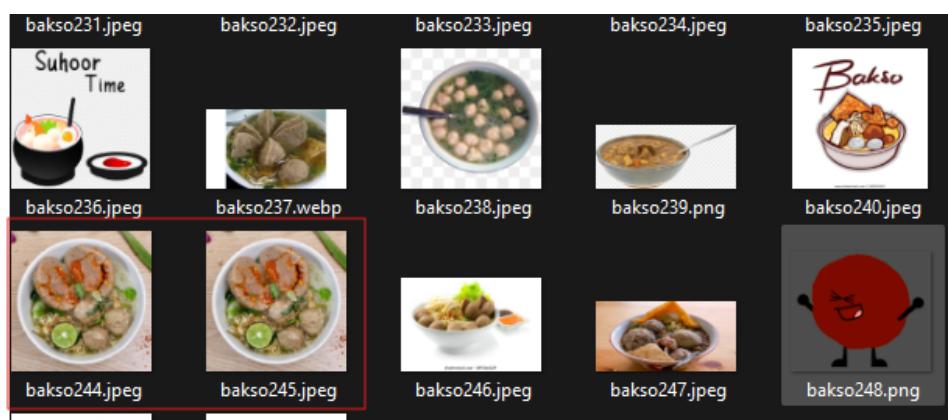
```
import splitfolders
import os
from keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt
import numpy as np
import cv2
```

- *Library* 'splitfolders' digunakan untuk membagi dataset menjadi data latih dan data pengujian.

- *Library 'os'* digunakan untuk membantu inisialisasi variabel data latih dan data pengujian berdasarkan nama folder.
- 'from keras.preprocessing.image import ImageDataGenerator' digunakan untuk mengambil fungsi *ImageDataGenerator* dari *library Keras*. *ImageDataGenerator* digunakan untuk melakukan *Image Normalization* dan *Image Augmentation*
- *Library 'matplotlib'* disingkat '*plt*' digunakan untuk visualisasi
- *Library 'numpy'* disingkat '*np*' digunakan untuk operasi matematika dan manipulasi array.
- *Library 'cv2'* digunakan untuk manipulasi data berbentuk citra.

3.2.1 Data Cleaning

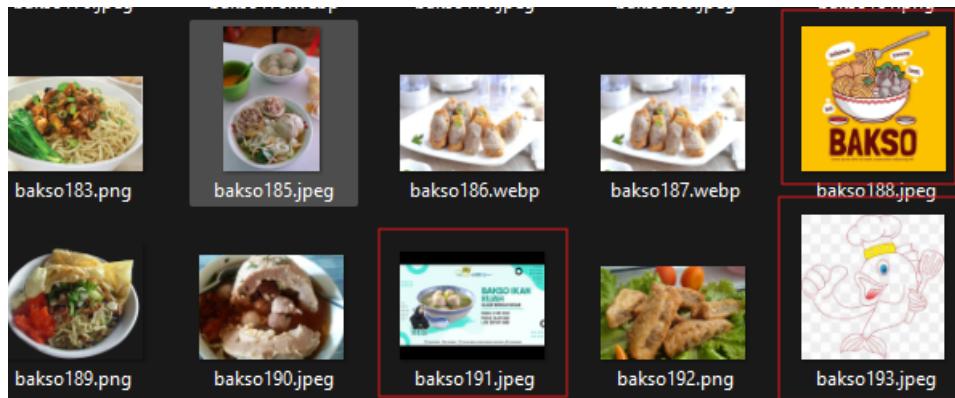
Pada tahap *Data Cleaning*, data citra yang berhasil dikumpulkan dilakukan pembersihan. Proses ini dilakukan untuk memastikan data bebas dari kesalahan dan inkonsistensi. Hal ini dilakukan karena data citra yang didapatkan dari *scraping* banyak yang *duplicate* dan banyak citra yang tidak sesuai dengan makanan yang dimaksud.



Gambar 3.4: Contoh Citra Duplicate

Pada Gambar 3.4 terlihat file citra dengan nama 'bakso244.jpeg' dengan 'bakso245.jpeg' merupakan citra yang sama atau disebut *duplicate*. Hal ini terjadi di

banyak kelas, tidak hanya di kelas 'bakso'. Pada Gambar 3.5 terihat beberapa file citra banyak yang tidak sesuai dengan kelasnya, seperti gambar kartun ataupun poster promosi.



Gambar 3.5: Contoh Citra yang tidak sesuai

Pada penelitian ini, tahap *Data Cleaning* dilakukan secara manual. Langkah pertama dalam proses *Data Cleaning* adalah pemeriksaan awal citra makanan. Setiap citra diperiksa untuk memastikan bahwa resolusi dan kualitasnya memadai untuk analisis lebih lanjut. Citra yang buram, terlalu gelap, atau terlalu terang dihapus dari dataset. Selain itu, citra yang tidak relevan atau tidak sesuai dengan kategori makanan yang diteliti juga dieliminasi.

Langkah kedua adalah deteksi anomali. Citra yang memiliki *noise* atau artefak yang tidak diinginkan diidentifikasi dan dieliminasi. Misalnya, citra yang memiliki bayangan berlebihan atau adanya objek asing yang mengganggu akan dieliminasi.

Tahap terakhir dalam proses *Data Cleaning* adalah validasi dan verifikasi. Setelah semua langkah di atas dilakukan, citra makanan yang telah dibersihkan diverifikasi kembali untuk memastikan bahwa tidak ada kesalahan yang terlewatkan. Citra yang lolos tahap ini kemudian siap digunakan untuk analisis lebih lanjut.

3.2.2 *Undersampling*

Pada tahapan ini, data citra yang telah dibersihkan melalui tahap *data cleaning* kemudian dilakukan proses *undersampling*. *Undersampling* dilakukan untuk menangani ketidakseimbangan data yang ada dalam dataset. Ketidakseimbangan data terjadi ketika jumlah citra dalam satu kelas makanan jauh lebih banyak dibandingkan

dengan kelas lainnya. Langkah-langkah dalam proses *undersampling* pada data citra ini meliputi:

1. Identifikasi Ketidakseimbangan Data

Langkah pertama adalah mengidentifikasi ketidakseimbangan dalam dataset. Hal ini dilakukan dengan menghitung jumlah citra dalam setiap kelas makanan dan menentukan kelas mana yang memiliki jumlah sampel yang jauh lebih banyak dibandingkan kelas lainnya.

2. Pemilihan Sampel untuk *Undersampling*

Setelah mengidentifikasi kelas mayoritas, langkah selanjutnya adalah memilih sejumlah sampel dari kelas tersebut untuk dihapus. Pemilihan sampel ini dapat dilakukan secara acak atau menggunakan metode tertentu yang mempertimbangkan distribusi data untuk menjaga keragaman sampel yang tersisa.

3. Penghapusan Sampel Kelas Mayoritas

Hapus sampel yang telah dipilih dari kelas mayoritas. Penghapusan ini dilakukan dengan hati-hati untuk memastikan bahwa sampel yang tersisa masih representatif dan mencakup berbagai variasi dalam kelas tersebut.

4. Validasi Dataset Baru

Setelah melakukan *undersampling*, penting untuk memvalidasi dataset baru. Proses validasi ini memastikan bahwa data yang tersisa masih representatif dan tidak mengandung bias yang dapat mempengaruhi hasil analisis. Validasi juga memastikan bahwa tidak ada informasi penting yang hilang dari kelas mayoritas.

3.2.3 Split Data

Setelah melewati tahapan *Undersampling*, kemudian dilakukan proses *Split Data*. *Split Data* merupakan proses untuk membagi data menjadi data latih dan data

uji. Pada penelitian ini, proses *Split Data* menggunakan *library* 'split-folders' dengan rasio 80% data latih dan 20% data uji. Untuk melakukan *Split Data*, digunakan perintah sebagai berikut:

```
!pip install split-folders
import splitfolders
splitfolders.ratio('/content/dataset',
output="split_images", seed=1337, ratio=(.8,.2))
```

Keterangan sintaks:

- '!pip install split-folders' digunakan untuk instalasi 'split-folders'
- 'import splitfolders' digunakan untuk memanggil fungsi splitfolders agar dapat digunakan.
- 'splitfolders.ratio' merupakan fungsi dari *library* 'splitfolders' untuk *Split Data*.
- '/content/dataset' merupakan folder dataset.
- 'output="split_images"' merupakan folder output.
- 'seed=1337' digunakan agar pembagian dataset tetap sama dan tidak random setiap menjalankan perintah ini.
- 'ratio=(.8,.2)' merupakan rasio pembagian dataset 80% untuk data latih dan 20% untuk data uji.

3.2.4 *Data Normalization*

Tahapan selanjutnya yaitu *Data Normalization*. Pada tahapan ini, data citra yang telah dibagi menjadi data latih dan data uji kemudian dilakukan proses normalisasi data. Proses *data normalization* digunakan untuk menyesuaikan skala nilai piksel pada sebuah gambar sehingga rentang nilai pikselnya menjadi lebih konsisten. Tujuannya adalah untuk meningkatkan performa model *machine learning* dalam mempelajari pola gambar tanpa adanya bias dari perbedaan skala nilai piksel. Pada proses *image normalization* setiap nilai piksel citra dibagi dengan 255. Kemudian data citra diubah menjadi ukuran 224 x 224 x 3.

3.2.5 Data Augmentation

Tahapan akhir pada *Data Preprocessing* yaitu tahapan *Data Augmentation*. *Data Augmentation* digunakan untuk memperluas dataset dengan membuat variasi pada gambar-gambar yang ada dengan cara melakukan transformasi pada gambar asli. Tujuannya dapat mencegah *overfitting*, mencegah adanya bias dan meningkatkan akurasi. Proses *Data Normalization* dan *Data Augmentation* diimplementasikan ke dalam bentuk kode seperti yang dapat dilihat pada blok program di bawah:

```
batch_size = 16

train_datagen = ImageDataGenerator(
    preprocessing_function=preprocess_input,
    rotation_range = 40,
    shear_range = 0.2,
    zoom_range = 0.2,
    horizontal_flip = True,
    vertical_flip = True)

val_datagen = ImageDataGenerator(
    preprocessing_function=preprocess_input)

train_dataset = train_datagen.flow_from_directory(
    train_dir,
    shuffle=False,
    target_size=(224, 224),
    batch_size=batch_size,
    class_mode='categorical')

val_dataset = val_datagen.flow_from_directory(
    val_dir,
    shuffle=False,
    target_size=(224, 224),
```

```
batch_size=batch_size,
class_mode='categorical')
```

Proses *Data Normalization* dan *Data Augmentation* dilakukan menggunakan fungsi `ImageDataGenerator` dari *library tensorflow*. Untuk proses *Data Normalization* dilakukan menggunakan sintaks '`preprocessing_function`', dan data citra diubah menjadi ukuran $224 \times 224 \times 3$ dilakukan menggunakan sintaks '`target_size(224, 224)`'. Kemudian proses *Data Augmentation* dilakukan hanya pada data latih dengan keterangan sebagai berikut:

1. '`rotation_range = 40`' digunakan untuk melakukan rotasi pada data citra sebesar 40 derajat.
2. '`shear_range = 0.2`' digunakan untuk menentukan intensitas shear (sudut shear dalam arah berlawanan jarum jam dalam derajat) sebagai fraksi dari lebar citra sebesar 20%.
3. '`zoom_range = 0.2`' digunakan untuk *zoom in* atau *zoom out* citra secara acak hingga 20%.
4. '`horizontal_flip = True`' digunakan untuk membalik citra secara acak secara horizontal.
5. '`vertical_flip = True`' digunakan untuk membalik citra secara acak secara vertical.

3.3 Pembuatan dan Pelatihan Model

Data citra yang sudah melewati proses *Data Preprocessing* kemudian dilakukan proses Pembuatan dan Pelatihan model. Pada penelitian ini, pelatihan model akan menggunakan 2 GPU yang berbeda, yang pertama yaitu menggunakan Google Colab dengan GPU T4, kemudian yang kedua menggunakan mesin DGX A100 dari Universitas Gunadarma. Pada proses ini, ada 2 tahapan yaitu tahapan *Transfer Learning* menggunakan *EfficientNetV2* dan *Training Model*. Dalam proses ini dibutuhkan beberapa *package Python* yang harus diinstalasi dan dipanggil saat program dijalankan di Jupyter Notebooks, yaitu

```
from tensorflow.keras.applications import EfficientNetV2S
from keras.layers import Input
from keras import layers
from tensorflow.keras.callbacks import ModelCheckpoint
import time
```

Keterangan sintaks:

- 'from tensorflow.keras.applications import EfficientNetV2S' digunakan untuk memanggil fungsi 'EfficientNetV2S' dari *library* tensorflow yang akan digunakan untuk proses *Transfer Learning*.
- 'from keras.layers import Input' digunakan untuk memanggil fungsi 'Input' dari *library* keras yang akan digunakan untuk menentukan input layer pada model.
- 'from keras import layers' digunakan untuk memanggil fungsi 'layers' dari *library* keras yang akan digunakan untuk membuat layer-layer pada model.
- 'from tensorflow.keras.callbacks import ModelCheckpoint' digunakan untuk memanggil fungsi 'ModelCheckPoint' dari *library* tensorflow yang akan digunakan untuk membuat checkpoint model terbaik pada saat *training model*.
- 'import time' digunakan untuk mencatat lama waktu training.

3.3.1 *Transfer Learning EfficientNetV2*

Tahapan pertama pada proses *Pembuatan dan Pelatihan Model* yaitu tahap *Transfer Learning* menggunakan EfficientNetV2. Pada penelitian ini model pra-terlatih yang digunakan merupakan EfficientNetV2S. EfficientNetV2S merupakan model pra-terlatih dari EfficientNetV2 yang terkecil. Pada penelitian ini, proses *transfer learning* dilakukan menggunakan *pre-trained weights* dari dataset ImageNet. ImageNet berisi lebih dari 14 juta gambar yang diberi label dalam 1000 kategori. Ini mencakup berbagai objek dan skenario, memberikan model dasar yang kuat dengan representasi visual yang luas dan beragam. Dengan menggunakan bobot yang telah

dilatih sebelumnya (*pre-trained weights*), kita dapat memanfaatkan pengetahuan yang telah diperoleh oleh model tersebut tanpa perlu melatihnya dari awal. Ini tidak hanya menghemat waktu tetapi juga biaya komputasi. Proses *Transfer Learning* menggunakan EfficientNetV2S diimplementasikan ke dalam bentuk kode seperti yang dapat dilihat pada blok program di bawah:

```
pre_trained_model = EfficientNetV2S(weights='imagenet',
include_top=False, input_tensor=Input(shape=(224, 224, 3)))

pre_trained_model.trainable = False

x = pre_trained_model.output
x = layers.GlobalAveragePooling2D()(x)
outputs = layers.Dense(n_classes, activation='softmax')(x)
model = tf.keras.models.Model(
pre_trained_model.input, outputs)
```

Keterangan sintaks:

- 'EfficientNetV2S': Ini adalah arsitektur model yang dipilih, yaitu EfficientNet versi 2 dengan ukuran S (small).
- 'weights='imagenet)': Model ini menggunakan bobot yang telah dilatih sebelumnya pada dataset ImageNet.
- 'include_top=False': Hanya menggunakan bagian dasar dari model (tanpa lapisan dense terakhir atau "top" layer yang biasanya digunakan untuk klasifikasi pada 1000 kelas ImageNet).
- 'input_tensor=Input(shape=(224, 224, 3))': Menentukan bentuk input dari gambar yang akan diproses oleh model, yaitu gambar dengan ukuran 224x224 piksel dan 3 saluran warna (RGB).
- 'trainable = False': Dengan mengatur trainable menjadi False, perintah ini akan mengunci semua bobot dalam model sehingga tidak akan diperbarui selama

proses pelatihan. Ini berarti bobot akan tetap sama seperti bobot awal yang dimuat dari ImageNet.

- 'pre_trained_model.output': Ini mengambil output dari lapisan terakhir model pra-terlatih (EfficientNetV2S). Output ini akan menjadi input untuk lapisan tambahan yang akan ditambahkan.
- 'layers.GlobalAveragePooling2D()': Ini adalah lapisan pooling yang mengurangi dimensi spasial dari fitur peta (*feature map*) yang dihasilkan oleh model pra-terlatih. Global Average Pooling menghitung rata-rata dari setiap peta fitur dan mengurangi data menjadi satu vektor per peta fitur.
- 'layers.Dense(n_classes, activation='softmax')': Ini adalah lapisan dense (*fully connected layer*) dengan jumlah neuron sebanyak n_classes (10 kelas). Aktivasi softmax digunakan untuk mengonversi output menjadi probabilitas kelas untuk tugas klasifikasi.
- 'tf.keras.models.Model(pre_trained_model.input, outputs)': Ini mendefinisikan model baru dengan input sama seperti input dari model pra-terlatih dan output sebagai output dari lapisan dense yang baru saja kita tambahkan. Model ini sekarang terdiri dari model pra-terlatih (yang bobotnya terkunci) dan lapisan baru yang akan dilatih.

3.3.2 *Training Model*

Setelah membuat model baru menggunakan metode *Transfer Learning* menggunakan EfficientNetV2S, dilakukan proses *training model* atau pelatihan model. Pada penelitian ini, pelatihan model akan menggunakan 2 GPU yang berbeda, yang pertama yaitu menggunakan Google Colab dengan GPU T4, kemudian yang kedua menggunakan mesin DGX A100 dari Universitas Gunadarma. Tahap pertama yang dilakukan yaitu melakukan kompilasi model. Proses kompilasi model diimplementasikan ke dalam bentuk kode seperti yang dapat dilihat pada blok program di bawah:

```
model.compile(
```

```

    optimizer=tf.keras.optimizers.Adam(learning_rate=0.001),
    loss='categorical_crossentropy',
    metrics=['accuracy']
)

```

Keterangan sintaks:

- 'tf.keras.optimizers.Adam': Adam (*Adaptive Moment Estimation*) adalah algoritma pengoptimalan yang sering digunakan dalam pembelajaran mendalam karena menggabungkan kelebihan dari dua algoritma optimasi lainnya, yaitu AdaGrad dan RMSProp. Adam bekerja dengan menyimpan perkiraan rata-rata momentum dari gradien serta kuadrat dari gradien.
- 'learning_rate=0.001': Ini adalah parameter laju pembelajaran yang digunakan oleh optimizer. Laju pembelajaran menentukan seberapa besar pembaruan bobot yang dilakukan pada setiap iterasi pelatihan. Nilai 0.001 adalah nilai umum yang sering digunakan karena memberikan keseimbangan yang baik antara kecepatan konvergensi dan stabilitas.
- 'categorical_crossentropy': Ini adalah fungsi kehilangan (*loss function*) yang digunakan untuk masalah klasifikasi dengan lebih dari dua kelas (klasifikasi multi-kelas). *Categorical crossentropy* mengukur perbedaan antara distribusi probabilitas yang diprediksi dan distribusi sebenarnya (*ground truth*).
- 'accuracy': Metrik yang digunakan untuk mengevaluasi kinerja model selama pelatihan dan pengujian. Akurasi adalah metrik yang umum digunakan yang menunjukkan persentase prediksi model yang benar. Dalam konteks klasifikasi multi-kelas, akurasi mengukur seberapa sering prediksi kelas dari model sesuai dengan label kelas sebenarnya.

Tahap kedua yaitu melakukan inisialisasi *checkpointer*. *Checkpointer* menggunakan fungsi callback 'ModelCheckpoint' dari *library tensorflow*. Pada penelitian ini, callback digunakan untuk menyimpan model selama pelatihan berdasarkan performa terbaik pada dataset validasi. Proses inisialisasi *checkpointer*

diimplementasikan ke dalam bentuk kode seperti yang dapat dilihat pada blok program di bawah:

```
checkpointer = ModelCheckpoint(
    filepath='best_model.h5',
    monitor='val_loss',
    verbose=1,
    save_best_only=True,
    mode='min'
)
```

Keterangan sintaks:

- 'best_model.h5': Ini adalah folder di mana model terbaik akan disimpan. File dengan ekstensi .h5 menunjukkan bahwa model akan disimpan dalam format HDF5, yang merupakan format populer untuk menyimpan model Keras.
- 'val_loss': Parameter ini menentukan metrik yang akan dipantau untuk menentukan apakah model yang sedang dilatih mengalami perbaikan. Dalam kasus ini, metrik yang dipantau adalah val_loss (loss pada data validasi).
- 'save_best_only=True': Jika diatur ke True, model hanya akan disimpan jika perbaikan pada metrik yang dipantau (dalam hal ini val_loss) ditemukan. Ini berarti model hanya akan disimpan ketika ia menunjukkan kinerja yang lebih baik dibandingkan semua iterasi sebelumnya.
- 'mode=min': Parameter ini menentukan apakah model yang lebih baik adalah model dengan nilai lebih rendah ('min') atau lebih tinggi ('max') dari metrik yang dipantau. Dalam kasus ini, mode 'min' berarti model dengan val_loss terendah akan disimpan.

Setelah kompilasi dan inisialisasi *checkpointer*, tahap terakhir yaitu melakukan *training model* atau pelatihan model menggunakan TensorFlow. Pada penelitian ini, jumlah epoch yang dipakai yaitu 50 epoch. Proses *training model* diimplementasikan ke dalam bentuk kode seperti yang dapat dilihat pada blok program di bawah:

```

num_epochs = 50
start_time = time.time()

history = model.fit(train_dataset,
                     epochs=num_epochs,
                     validation_data=val_dataset,
                     callbacks=[checkpointer]
                     )

elapsed_time = time.time() - start_time
time.strftime("%H:%M:%S", time.gmtime(elapsed_time))

```

Keterangan sintaks:

- 'num_epoch's = 50: Mengatur jumlah epoch untuk pelatihan model. Satu epoch adalah satu siklus penuh melalui seluruh dataset pelatihan. Dalam kasus ini, model akan dilatih selama 50 epoch.
- 'start_time = time.time()': Menyimpan waktu mulai pelatihan dalam satuan detik. Ini digunakan untuk menghitung berapa lama waktu yang dibutuhkan untuk melatih model.
- 'model.fit': Metode ini digunakan untuk melatih model dengan dataset pelatihan.
- 'train_dataset': Dataset yang digunakan untuk melatih model.
- 'epochs=num_epochs': Menentukan jumlah epoch untuk pelatihan, yaitu 50 epoch.
- 'validation_data=val_dataset': Dataset yang digunakan untuk validasi model selama pelatihan. Performansi model pada dataset ini digunakan untuk memantau *overfitting*.
- 'callbacks=[checkpointer]': Daftar callback yang digunakan selama pelatihan. Dalam hal ini, checkpointer adalah callback ModelCheckpoint yang telah

diinisialisasi sebelumnya, yang digunakan untuk menyimpan model terbaik selama pelatihan.

- 'elapsed_time = time.time() - start_time': Menghitung selisih waktu antara waktu selesai pelatihan dan waktu mulai pelatihan. Ini memberikan total waktu yang dibutuhkan untuk melatih model dalam satuan detik.
- 'time.strftime("%H:%M:%S", time.gmtime(elapsed_time))': Mengubah waktu yang dihitung (elapsed_time) menjadi format yang lebih mudah dibaca (jam:menit)

3.4 Pengujian Model

Setelah dilakukan proses *Data Preprocessing*, selanjutnya yaitu dilakukan Pengujian Model. Pengujian Model merupakan proses terakhir dari penelitian ini. Pada proses ini dibagi menjadi 3 tahapan, yaitu Evaluasi Model, Visualisasi Grad-CAM dan Pengujian Model menggunakan data nutrisi makanan. Dalam proses ini dibutuhkan beberapa *package* Python yang harus diinstalasi dan dipanggil saat program dijalankan di Jupyter Notebooks, yaitu:

```
from sklearn.metrics import accuracy_score,
classification_report, confusion_matrix,
ConfusionMatrixDisplay
```

Keterangan sintaks:

- 'sklearn.metrics': merupakan kode untuk menggunakan kelas metrics dari library sklearn atau scikit-learn.
- 'accuracy_score': Fungsi ini menghitung akurasi klasifikasi, yang merupakan rasio prediksi benar dibandingkan dengan total prediksi. Ini adalah metrik evaluasi yang paling dasar untuk masalah klasifikasi.
- 'classification_report': Fungsi ini menghasilkan laporan yang menunjukkan metrik kinerja utama untuk masalah klasifikasi, seperti *precision*, *recall*, *f1-score*, dan *support* untuk setiap kelas.

- 'confusion_matrix': Fungsi ini menghitung *confussion matrix* untuk mengukur kinerja algoritma klasifikasi. *Confussion matrix* adalah tabel yang digunakan untuk menggambarkan kinerja model klasifikasi pada serangkaian data uji yang nilai sebenarnya diketahui.
- 'ConfusionMatrixDisplay': Kelas ini digunakan untuk memvisualisasikan *confussion matrix* yang dihasilkan oleh 'confusion_matrix'. Ini membantu dalam memvisualisasikan kesalahan klasifikasi dengan lebih mudah.

3.4.1 Evaluasi Model

Pada tahapan evaluasi model, langkah pertama yang dilakukan yaitu melihat apakah model yang dilatih merupakan model yang *overfit*, *underfit*, atau *goodfit*. Untuk mengetahui hal tersebut, dapat dilakukan visualisasi terhadap pelatihan model setiap epoch nya. Proses visualisasi diimplementasikan ke dalam bentuk kode seperti yang dapat dilihat pada blok program di bawah:

```
def report_train(history):
    # loss
    plt.plot(history.history['loss'], label='train loss')
    plt.plot(history.history['val_loss'], label='val loss')
    plt.legend()
    plt.show()

    # accuracies
    plt.plot(history.history['accuracy'], label='train acc')
    plt.plot(history.history['val_accuracy'], label='val acc')
    plt.legend()
    plt.show()

report_train(history)
```

Kode tersebut memberikan visualisasi yang jelas tentang bagaimana performa model selama pelatihan dan validasi dengan menampilkan loss dan akurasi. Ini membantu dalam menganalisis apakah model mengalami overfitting atau underfitting. Dengan melihat grafik loss dan akurasi, proses pelatihan dapat dipantau dan

mengambil keputusan yang tepat, seperti menyesuaikan hiperparameter atau menghentikan pelatihan lebih awal jika diperlukan.

Setelah memastikan model yang dilatih merupakan model yang *good fit*, langkah selanjutnya yaitu mengevaluasi kinerja model pada dataset pengujian. Dalam penelitian ini, evaluasi model dilakukan menggunakan fungsi 'classification_report' dari scikit-learn, dan menggunakan visualisasi *Confussion Matrix*. Proses tersebut diimplementasikan ke dalam bentuk kode seperti yang dapat dilihat pada blok program di bawah:

```
def report_test(test_set, model=model):

    # evaluating test
    model.evaluate(test_set, batch_size=32)
    y_pred = model.predict(test_set)
    y_pred = np.argmax(y_pred, axis=1)
    accuracy_score(y_pred, test_set.classes)
    print(classification_report(y_pred, test_set.classes))

    # confusion_matrix
    labels = [i for i in train_dataset.class_indices]
    cm = confusion_matrix(y_pred, test_set.classes)
    disp = ConfusionMatrixDisplay(cm, display_labels=labels)
    disp.plot(cmap='Blues', xticks_rotation='vertical')
    plt.show()

report_test(val_dataset)
```

Kode tersebut memberikan evaluasi komprehensif dari kinerja model pada dataset uji dengan memberikan metrik *accuracy*, *loss*, *precision*, *recall*, *f1-score*, *support*, dan *confussion matrix*.

3.4.2 Visualisasi Grad-CAM

Setelah melalui tahap evaluasi model, visualisasi Grad-CAM dibutuhkan untuk memahami bagaimana model mengambil keputusan dengan memvisualisasikan bagian dari citra yang diperhatikan oleh model. Teknik ini membantu melihat area mana yang dianggap penting oleh model saat membuat prediksi. Dengan menggunakan Grad-CAM, fitur-fitur spesifik dalam gambar yang mempengaruhi keputusan model dapat disorot, sehingga memberikan wawasan yang lebih dalam mengenai mekanisme internal model. Proses visualisasi Grad-CAM diimplementasikan ke dalam bentuk kode seperti yang dapat dilihat pada blok program di bawah:

```
def get_gradcam_heatmap(model, img_array,
last_conv_layer_name, pred_index=None):
    grad_model = tf.keras.models.Model(
        [model.inputs], [model.get_layer(
            last_conv_layer_name).output, model.output]
    )

    with tf.GradientTape() as tape:
        conv_layer_output,
        predictions = grad_model(img_array)
        if pred_index is None:
            pred_index = tf.argmax(predictions[0])
        class_channel = predictions[:, pred_index]

        grads = tape.gradient(class_channel, conv_layer_output)
        pooled_grads = tf.reduce_mean(grads, axis=(0, 1, 2))

        conv_layer_output = conv_layer_output[0]
        heatmap = conv_layer_output @
        pooled_grads[..., tf.newaxis]
        heatmap = tf.squeeze(heatmap)
```

```

    heatmap = tf.maximum(heatmap, 0)
    / tf.math.reduce_max(heatmap)
    heatmap = heatmap.numpy()
    return heatmap

def display_gradcam_with_labels(img_path,
                                heatmap, real_label, predicted_label, alpha=0.4):
    img = cv2.imread(img_path)
    heatmap = cv2.resize(heatmap,
                         (img.shape[1], img.shape[0]))
    heatmap = np.uint8(255 * heatmap)
    heatmap = cv2.applyColorMap(heatmap, cv2.COLORMAP_JET)
    superimposed_img = cv2.addWeighted(img, alpha,
                                       heatmap, 1 - alpha, 0)

    plt.figure(figsize=(10, 10))
    plt.subplot(1, 2, 1)
    plt.title(f'Original Image\nReal: {real_label}')
    plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
    plt.axis('off')

    plt.subplot(1, 2, 2)
    plt.title(f'Grad-CAM\nPredicted: {predicted_label}')
    plt.imshow(cv2.cvtColor(
        superimposed_img, cv2.COLOR_BGR2RGB))
    plt.axis('off')
    plt.show()

```

3.4.3 Pengujian Model Menggunakan Data Nutrisi Makanan

Setelah dipastikan model layak untuk dipakai, langkah terakhir yaitu pengujian model menggunakan data nutrisi makanan. Data nutrisi makanan diambil secara manual melalui situs nilaigizi.com. Dalam penelitian ini, data berhasil dikumpulkan

dalam bentuk file excel.

Setelah mempunyai data nutrisi makanan, file tersebut akan digunakan untuk mencari nutrisi makanan berdasarkan nama makanan yang berhasil diklasifikasi oleh model yang telah dilatih. Pada Gambar 3.6 terdapat data nutrisi 10 makanan yaitu ayam bakar, bakso, gado gado, gudeg, nasi goreng, pempek kapal selam, rawon, rendang, sate ayam, dan soto betawi. Data yang didapat meliputi energi, protein, lemak, karbo, dan jumlah per sajian.

no	nama makanan	energi (kkal)	protein (g)	lemak (g)	karbo (g)	jumlah per sajian (g)
1	ayam bakar	226	23,00	15,00	0,10	100
2	bakso	202	12,41	13,16	7,58	100
3	gado gado	137	6,10	3,20	21,00	100
4	gudeg	160	3,30	9,20	16,00	100
5	nasi goreng	500	7,00	34,40	40,20	200
6	pempek kapal selam	152	4,50	2,30	28,20	100
7	rawon	60	5,40	2,50	4,00	100
8	rendang	193	22,60	7,90	7,80	100
9	sate ayam	225	19,54	14,82	4,87	100
10	soto betawi	135	2,50	8,80	11,50	100

Gambar 3.6: Hasil pengumpulan data nutrisi makanan dari situs nilaigizi.com

Proses pencarian data nutrisi makanan diimplementasikan ke dalam bentuk kode seperti yang dapat dilihat pada blok program di bawah:

```
def predict_class_with_nutrition(model , images ,
show = True):
    for img in images:
        img = image.load_img(img , target_size=(224 , 224))
        img = image.img_to_array(img)
        img = np.expand_dims(img , axis=0)
        img = preprocess_input(img)

        preds = model.predict(img)
        class_labels = class_names
        pred = np.argmax(preds , axis=-1)
        predicted_class = class_names[pred[0]]
        predicted_prob = preds[0][pred[0]] * 100

    nutrition = data[data['nama makanan'].str.contains(
```

```
predicted_class, case=False)]  
nutrition = nutrition.iloc[:, 1:]  
print("-" * 30)  
for col in nutrition.columns:  
    print(f"{col.capitalize():>20}:  
    {nutrition[col].iloc[0]:}")  
print("-" * 30)  
  
if show:  
    plt.imshow(img[0].astype('uint8'))  
    plt.axis('off')  
    plt.title(f"{predicted_class}  
({predicted_prob:.2f}%)")  
    plt.show()
```

BAB IV

HASIL DAN PEMBAHASAN

Setelah dilakukan penjelasan terhadap langkah-langkah mulai dari pengumpulan data, *data preprocessing*, Pembuatan dan Pelatihan Model, dan Pengujian Model. Selanjutnya, menampilkan hasil dari tahapan-tahapan tersebut. Kinerja model klasifikasi dalam melakukan pengklasifikasian juga akan diperlihatkan untuk mengukur seberapa baik model tersebut dalam mengklasifikasikan citra makanan. *Confussion matrix* dan *classification report* akan digunakan sebagai alat ukur untuk evaluasi pada klasifikasi citra makanan tersebut. Semakin tinggi akurasi klasifikasi berarti performa klasifikasi juga semakin baik.

4.1 Hasil Pengumpulan Data

Hasil pengumpulan data citra makanan yang diperoleh menggunakan teknik *Scraping* berjumlah sebanyak 5000 citra makanan yang terbagi dalam 10 kelas (ayam bakar, bakso, gado gado, gudeg, nasi goreng, pempek, rawon, rendang, sate, dan soto), dimana setiap kelas memiliki total 500 citra makanan.



Gambar 4.1: Hasil pengumpulan data citra makanan

Gambar 4.1 merupakan contoh data citra makanan yang berhasil dikumpulkan menggunakan teknik *scraping* dalam penelitian ini. Gambar 4.1 menunjukkan perbedaan signifikan dari gado-gado dan bakso. Perbedaan signifikan tersebut meliputi bentuk dan warna.

4.2 Hasil *Data Preprocessing*

Hasil dari *data preprocessing* terbagi menjadi menjadi 4 bagian, yaitu:

1. Hasil *Data Cleaning*.
2. Hasil *Undersampling*.
3. Hasil *Split Data*.
4. Hasil *Data Normalization* dan *Data Augmentation*.

4.2.1 Hasil *Data Cleaning*

Setelah data berhasil dikumpulkan menggunakan teknik *scraping*, kemudian dilakukan proses *Data Cleaning*. Tahap ini akan memperlihatkan hasil dari *data cleaning*. Berikut hasil dari *data cleaning* pada Tabel 4.1

Tabel 4.1: Hasil Cleaning Data

Makanan	Jumlah Data
ayam bakar	372
bakso	278
gado gado	235
gudeg	259
nasi goreng	406
pempek	252
rawon	342
rendang	209
sate	204
soto	293

Hal ini dilakukan karena data citra yang didapatkan dari *scraping* banyak yang *duplicate* dan banyak citra yang tidak sesuai dengan makanan yang dimaksud. *Data Cleaning* dilakukan agar dapat memastikan konsistensi data dan model dapat mengenali data yang benar.

4.2.2 Hasil *Undersampling*

Setelah melakukan *data cleaning*, langkah berikutnya adalah *Undersampling* pada data. *Undersampling* dilakukan untuk menangani ketidakseimbangan data yang ada dalam dataset. Ketidakseimbangan data terjadi ketika jumlah citra dalam satu kelas makanan jauh lebih banyak dibandingkan dengan kelas lainnya. Hasil *Undersampling* dapat dilihat pada Tabel 4.2.

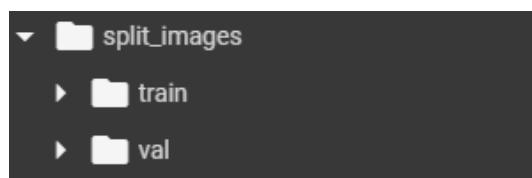
Tabel 4.2: Hasil Undersampling

Makanan	Jumlah Sebelum Undersampling	Jumlah Sesudah Undersampling
ayam bakar	372	200
bakso	278	200
gado gado	235	200
gudeg	259	200
nasi goreng	406	200
pempek	252	200
rawon	342	200
rendang	209	200
sate	204	200
soto	293	200

Setelah dilakukan *Undersampling*, semua data pada tiap kelas menjadi 200 citra. *Undersampling* dilakukan untuk mengurangi adanya bias. Bias biasa terjadi ketika terjadi ketidakseimbangan data.

4.2.3 Hasil *Split Data*

Setelah dilakukan proses *Undersampling*, data kemudian dibagi menjadi menjadi data latih dan data uji. Berikut hasil *Split Data* pada Gambar 4.2.



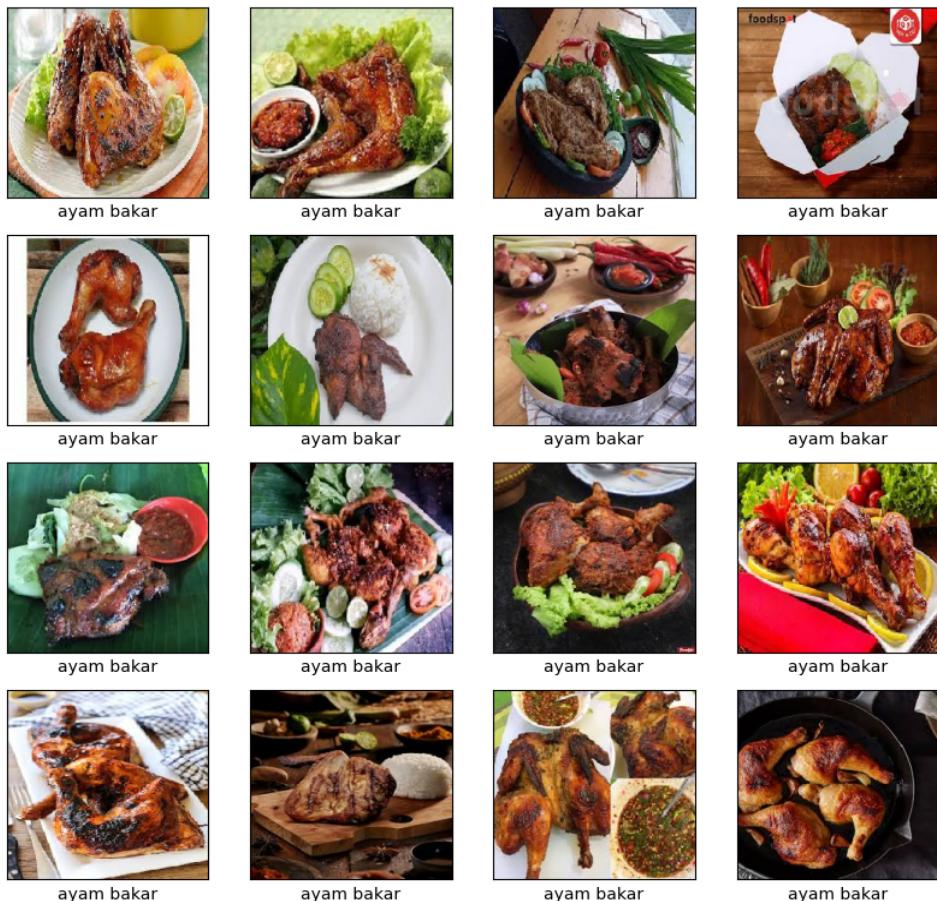
Gambar 4.2: Hasil Split Data

Setelah dilakukan *Split Data*, dataset sekarang mempunyai 2 bagian, yaitu data latih dan data uji. Dengan rasio 80% untuk data latih, dan 20% untuk data uji, data latih mempunyai 160 citra tiap kelasnya, sedangkan data uji mempunyai 40 citra tiap kelasnya.

4.2.4 Hasil *Data Normalization* dan *Data Augmentation*

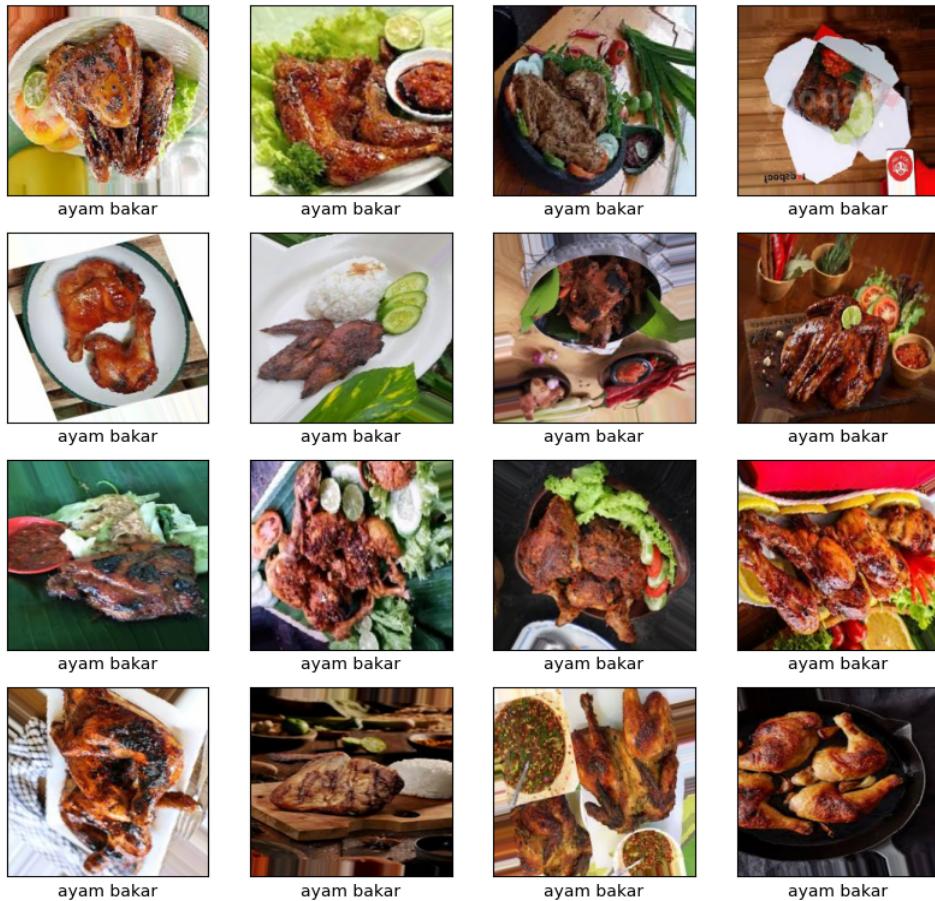
Setelah dilakukan *Split Data* menjadi data uji dan data latih, kemudian dilakukan proses *Data Normalization* dan *Data Augmentation*. Berikut contoh dataset sebelum dilakukan proses *Data Normalization* dan *Data Augmentation* pada Gambar 4.3.

Images example in train dataset



Gambar 4.3: Data Sebelum Proses *Data Normalization* dan *Data Augmentation*

Images example in train dataset after augmentation



Gambar 4.4: Data Sesudah Proses Data Normalization dan Data Augmentation

Pada Gambar 4.4, terlihat citra yang telah dilakukan proses *Data Normalization* dan *Data Augmentation* terlihat berbeda dengan sebelumnya. Setelah dilakukan proses tersebut, terlihat beberapa citra terbalik secara vertikal ataupun horizontal, beberapa citra mengalami rotasi, dan beberapa citra diperbesar.

4.3 Hasil Pembuatan dan Pelatihan Model

Setelah dilakukan proses *Data Preprocessing*, kemudian dilakukan proses Pembuatan dan Pelatihan Model. Hasil Pembuatan dan Pelatihan Model dibagi menjadi 2 bagian, yaitu:

1. Hasil *Transfer Learning* EfficientNetV2

2. Hasil *Training Model*

4.3.1 Hasil *Transfer Learning* EfficientNetV2

block6o_drop (Dropout)	(None, 7, 7, 256)	0	['block6o_project_bn[0][0]']
block6o_add (Add)	(None, 7, 7, 256)	0	['block6o_drop[0][0]', 'block6n_add[0][0]']
top_conv (Conv2D)	(None, 7, 7, 1280)	327680	['block6o_add[0][0]']
top_bn (BatchNormalization)	(None, 7, 7, 1280)	5120	['top_conv[0][0]']
top_activation (Activation)	(None, 7, 7, 1280)	0	['top_bn[0][0]']
global_average_pooling2d (GlobalAveragePooling2D)	(None, 1280)	0	['top_activation[0][0]']
dense (Dense)	(None, 10)	12810	['global_average_pooling2d[0][0]']
<hr/>			
=====			
Total params: 20344170 (77.61 MB)			
Trainable params: 12810 (50.04 KB)			
Non-trainable params: 20331360 (77.56 MB)			

Gambar 4.5: Ringkasan Model Hasil *Transfer Learning* EfficientNetV2

Gambar 4.5 merupakan ringkasan dari model yang telah dibuat. Pada penelitian ini, model yang telah dibuat menggunakan *layer* dari EfficientNetV2, lalu ditambahkan lapisan tambahan yaitu lapisan GlobalAveragePooling2D dan lapisan Dense dengan aktivasi softmax. Berikut adalah penjelasan dari parameter yang terdapat dalam Gambar 4.5:

- **Total params: 20,344,170 (77.61 MB):** Ini adalah jumlah total parameter dalam model yang telah dibuat. Parameter ini mencakup semua bobot (*weights*) dan bias (*biases*) yang ada di dalam model, baik yang dapat dilatih maupun yang tidak dapat dilatih.
- **Trainable params: 12,810 (50.04 KB):** Ini adalah jumlah parameter yang dapat dilatih dalam model yang telah dibuat. Dalam kasus ini, hanya ada 12,810 parameter yang dapat dilatih. Parameter ini berasal dari lapisan tambahan yang ditambahkan setelah model pra-latih EfficientNetV2S, yaitu lapisan GlobalAveragePooling2D dan lapisan Dense dengan aktivasi softmax.
- **Non-trainable params: 20,331,360 (77.56 MB):** Ini adalah jumlah parameter yang tidak dapat dilatih. Parameter ini merupakan parameter

dari model pra-latih EfficientNetV2S yang digunakan. Karena diatur `pre_trained_model.trainable = False`, parameter dalam EfficientNetV2S tidak akan diperbarui selama pelatihan.

4.3.2 Hasil *Training Model*

Setelah dilakukan pembuatan model dengan menggunakan metode *transfer learning*, kemudian dilakukan proses *training model* atau pelatihan model. Pada penelitian ini, model dilatih menggunakan *learning rate* sebesar 0.001, callback ModelCheckPoint untuk menyimpan model terbaik berdasarkan `val_loss` terkecil, dan menggunakan epoch sebanyak 50 epoch. Hasil *Training Model* dibagi menjadi 2 bagian, yaitu:

1. Hasil *Training Model* menggunakan Google Colab GPU T4
2. Hasil *Training Model* menggunakan mesin DGX A100 Universitas Gunadarma.

4.3.2.1 Hasil *Training Model* menggunakan Google Colab GPU T4

Berikut tabel hasil pelatihan model menggunakan Google Colab GPU T4 dengan konfigurasi tersebut:

Tabel 4.3: Hasil Training Model menggunakan Google Colab GPU T4

Epoch	Loss	Accuracy	Val Loss	Val Accuracy
1	1.6711	0.4756	1.1215	0.7075
2	1.0631	0.6938	0.8734	0.7750
3	0.8727	0.7500	0.7371	0.8050
4	0.7815	0.7750	0.6796	0.8125
5	0.7155	0.7987	0.6434	0.8025
6	0.6573	0.8050	0.5976	0.8300
7	0.6139	0.8156	0.5783	0.8325
8	0.6103	0.8150	0.5584	0.8425
9	0.5695	0.8256	0.5399	0.8375

Epoch	Loss	Accuracy	Val Loss	Val Accuracy
10	0.5508	0.8344	0.5320	0.8400
11	0.5284	0.8356	0.5189	0.8325
12	0.5085	0.8531	0.5145	0.8450
13	0.4952	0.8481	0.5069	0.8425
14	0.4727	0.8575	0.4941	0.8450
15	0.4651	0.8606	0.4792	0.8600
16	0.4516	0.8644	0.4829	0.8525
17	0.4356	0.8750	0.4679	0.8400
18	0.4449	0.8669	0.4661	0.8525
19	0.4313	0.8763	0.4571	0.8575
20	0.4063	0.8769	0.4485	0.8500
21	0.3960	0.8800	0.4459	0.8650
22	0.3972	0.8781	0.4390	0.8575
23	0.4013	0.8794	0.4496	0.8625
24	0.3850	0.8913	0.4375	0.8700
25	0.3676	0.8850	0.4280	0.8700
26	0.3660	0.8869	0.4261	0.8625
27	0.3475	0.8950	0.4205	0.8625
28	0.3448	0.8944	0.4269	0.8600
29	0.3507	0.8925	0.4168	0.8575
30	0.3666	0.8913	0.4256	0.8625
31	0.3388	0.8925	0.4077	0.8725
32	0.3214	0.9013	0.4134	0.8725
33	0.3379	0.8919	0.4233	0.8500
34	0.3017	0.9112	0.4130	0.8800
35	0.3210	0.8988	0.4064	0.8650
36	0.3069	0.9156	0.4207	0.8725
37	0.3337	0.8994	0.4099	0.8675
38	0.3253	0.9000	0.4069	0.8725

Epoch	Loss	Accuracy	Val Loss	Val Accuracy
39	0.3086	0.9106	0.4182	0.8700
40	0.3111	0.8988	0.4115	0.8825
41	0.2815	0.9125	0.4112	0.8775
42	0.3118	0.8994	0.4185	0.8700
43	0.3002	0.9094	0.4133	0.8800
44	0.2931	0.9131	0.4078	0.8750
45	0.2639	0.9231	0.4116	0.8775
46	0.2842	0.9112	0.4002	0.8600
47	0.2834	0.9100	0.4093	0.8775
48	0.2660	0.9269	0.3959	0.8725
49	0.2938	0.9112	0.3944	0.8750
50	0.2782	0.9094	0.3964	0.8700

Tabel 4.3 merupakan hasil pelatihan model menggunakan Google Colab GPU T4 sebanyak 50 epoch. Model terbaik yang diambil merupakan model dengan *Val Loss* terkecil, berdasarkan hasil pelatihan model pada Tabel 4.3, model terbaik merupakan model pada Epoch ke-49, dengan *Val Loss* sebesar 0.3944 dan *Val Accuracy* sebesar 0.8750 atau 87.50%. Dengan menggunakan Google Colab GPU T4, pelatihan memakan waktu selama 22 menit.

4.3.2.2 Hasil *Training Model* menggunakan mesin DGX A100 Universitas Gunadarma

Berikut tabel hasil pelatihan model menggunakan mesin DGX A100 Universitas Gunadarma dengan konfigurasi tersebut:

Tabel 4.4: Hasil Training Model menggunakan mesin DGX A100 Universitas Gunadarma

Epoch	Loss	Accuracy	Val Loss	Val Accuracy
1	1.6661	0.4769	1.1257	0.7225
2	1.0714	0.7075	0.8545	0.7800
3	0.8800	0.7519	0.7542	0.7975

Epoch	Loss	Accuracy	Val Loss	Val Accuracy
4	0.7612	0.7813	0.6723	0.8150
5	0.6978	0.7962	0.6406	0.8200
6	0.6447	0.8194	0.6116	0.8300
7	0.6459	0.7987	0.5735	0.8350
8	0.5755	0.8356	0.5565	0.8300
9	0.5942	0.8244	0.5365	0.8525
10	0.5579	0.8231	0.5220	0.8550
11	0.5254	0.8438	0.5199	0.8450
12	0.5267	0.8381	0.5095	0.8425
13	0.4851	0.8637	0.5151	0.8425
14	0.4779	0.8500	0.4901	0.8450
15	0.4677	0.8562	0.4832	0.8500
16	0.4511	0.8669	0.4700	0.8650
17	0.4482	0.8662	0.4554	0.8725
18	0.4306	0.8737	0.4606	0.8625
19	0.4220	0.8725	0.4526	0.8650
20	0.4057	0.8725	0.4497	0.8775
21	0.4081	0.8788	0.4615	0.8600
22	0.4060	0.8756	0.4485	0.8550
23	0.3850	0.8819	0.4457	0.8650
24	0.3878	0.8844	0.4349	0.8600
25	0.3504	0.8919	0.4393	0.8675
26	0.3802	0.8838	0.4369	0.8575
27	0.3701	0.8819	0.4514	0.8600
28	0.3587	0.8963	0.4231	0.8675
29	0.3353	0.8994	0.4215	0.8725
30	0.3624	0.8900	0.4167	0.8775
31	0.3464	0.8938	0.4198	0.8700
32	0.3395	0.8981	0.4322	0.8600

Epoch	Loss	Accuracy	Val Loss	Val Accuracy
33	0.3489	0.8938	0.4217	0.8600
34	0.3135	0.9112	0.4093	0.8675
35	0.3361	0.8969	0.4076	0.8725
36	0.3233	0.9050	0.4176	0.8700
37	0.2991	0.9081	0.4137	0.8650
38	0.3152	0.9013	0.4075	0.8700
39	0.3160	0.9019	0.4044	0.8650
40	0.2874	0.9112	0.4123	0.8700
41	0.3059	0.9075	0.4051	0.8775
42	0.3173	0.9031	0.3951	0.8700
43	0.2998	0.9137	0.3979	0.8775
44	0.3042	0.9106	0.3985	0.8725
45	0.3059	0.9000	0.3955	0.8675
46	0.2976	0.9069	0.4004	0.8750
47	0.2927	0.9081	0.3970	0.8725
48	0.2675	0.9169	0.3947	0.8700
49	0.2677	0.9200	0.4089	0.8725
50	0.2723	0.9162	0.3925	0.8725

Tabel 4.4 merupakan hasil pelatihan model menggunakan mesin DGX A100 Universitas Gunadarma sebanyak 50 epoch. Model terbaik yang diambil merupakan model dengan *Val Loss* terkecil, berdasarkan hasil pelatihan model pada Tabel 4.4, model terbaik merupakan model pada Epoch ke-50, dengan *Val Loss* sebesar 0.3925 dan *Val Accuracy* sebesar 0.8725 atau 87.25%. Dengan menggunakan mesin DGX A100 Universitas Gunadarma, pelatihan memakan waktu selama 11 menit saja.

4.4 Hasil Pengujian Model

Setelah melalui tahap *Training Model*, kemudian dilakukan tahap terakhir yaitu Pengujian Model. Hasil Pengujian Model dibagi menjadi 2 bagian, yaitu:

1. Hasil Pengujian Model Google Colab GPU T4
2. Hasil Pengujian Model mesin DGX A100 Universitas Gunadarma

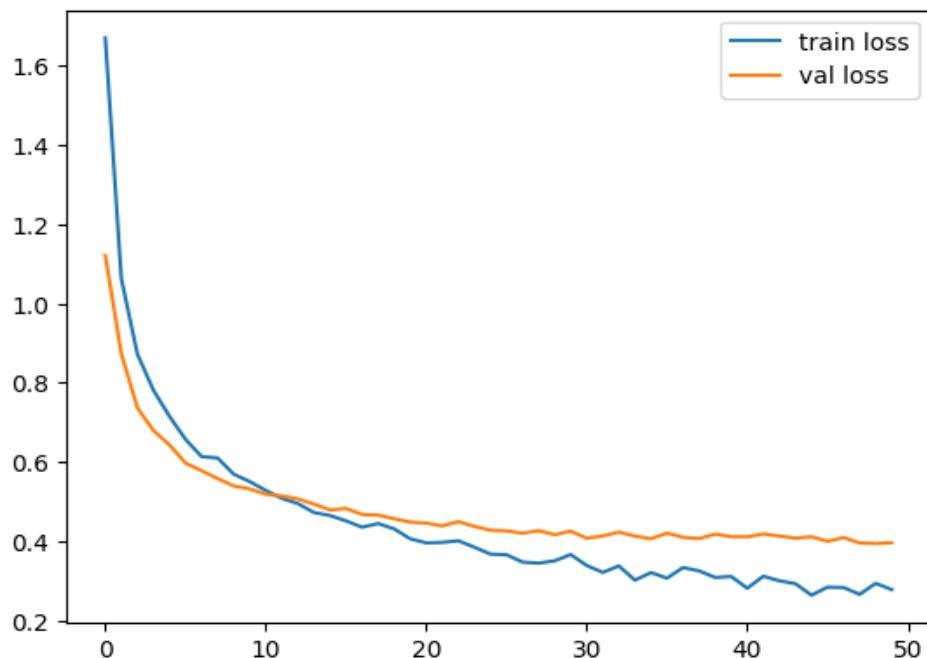
4.4.1 Hasil Pengujian Model Google Colab GPU T4

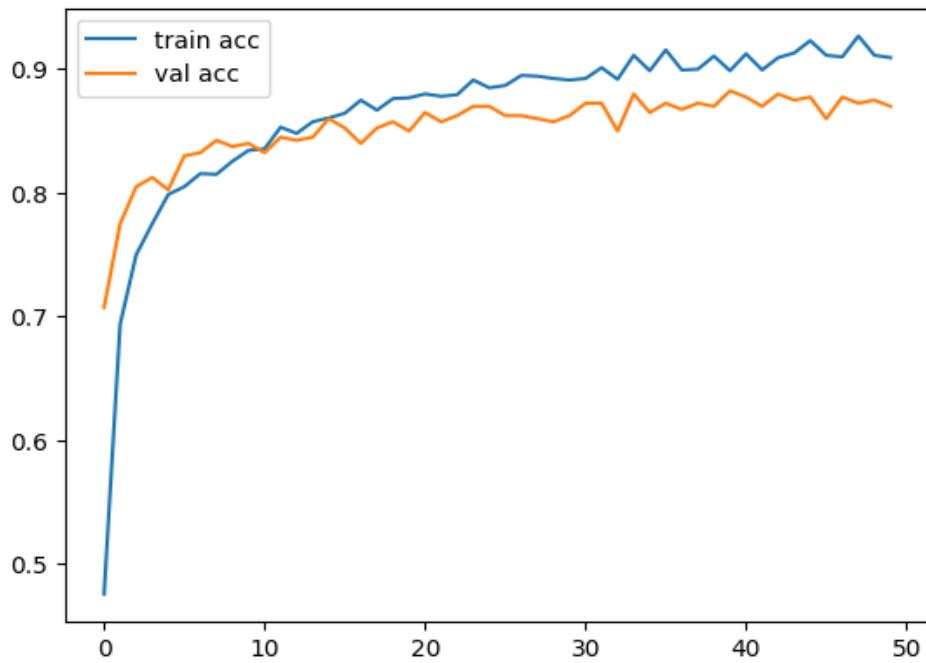
Hasil Pengujian Model Google Colab GPU T4 dibagi menjadi 3 bagian, yaitu:

1. Hasil Evaluasi Model
2. Hasil Visualisasi Grad-CAM
3. Hasil Pengujian Model menggunakan data nutrisi makanan

4.4.1.1 Hasil Evaluasi Model

Tahap pertama pada Evaluasi Model yaitu untuk mengetahui model tersebut termasuk *overfit*, *good fit* atau *underfit*. Untuk mengetahui hal tersebut, dapat dilakukan dengan membuat visualisasi dari hasil pelatihan model dengan diagram garis. Berikut hasil visualisasi pelatihan model dengan diagram garis:





Gambar 4.6: Visualisasi Pelatihan Model Google Colab GPU T4

Pada Gambar 4.6, terlihat *train loss* dan *val loss* mendekati satu sama lain dan terus menurun hingga datar dari beberapa titik hingga akhir. Kemudian, *train acc* dan *val acc* juga mendekati satu sama lain. Hal ini dapat diartikan bahwa model tersebut termasuk *good fit* dan layak dipakai.

Setelah mengetahui bahwa model layak dipakai, kemudian dilakukan *Classification Report* untuk mengetahui performa model berdasarkan metriks *precision*, *recall*, dan *F1-Score*. Berikut hasil dari *Classification Report* menggunakan *library* dari *sklearn*:

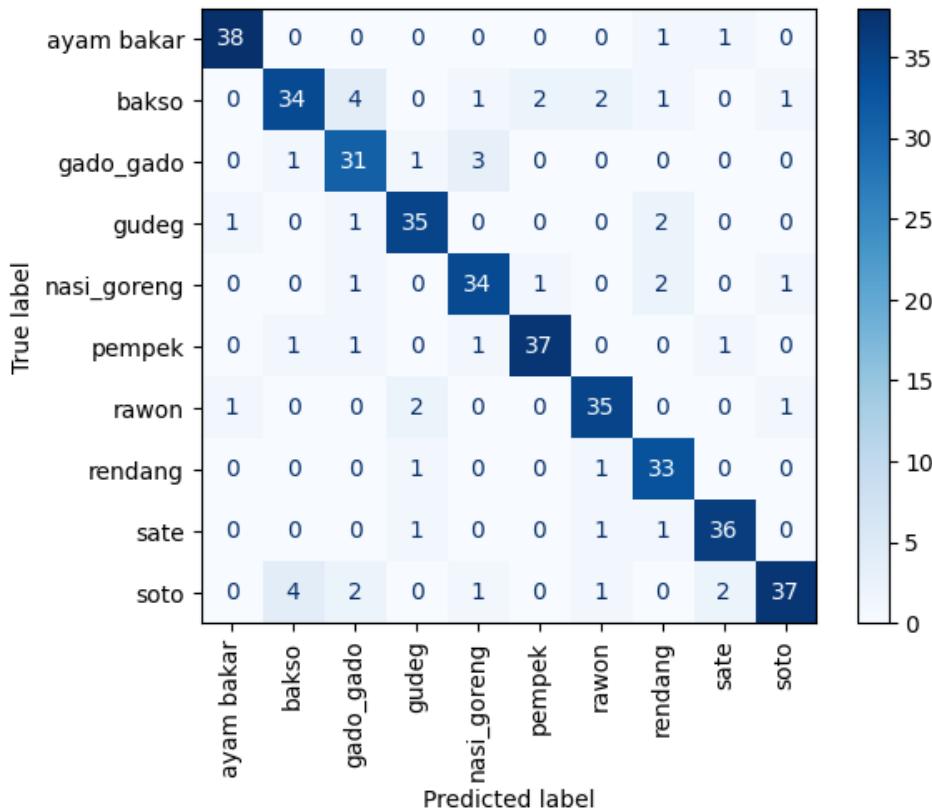
25/25 [=====] - 6s 76ms/step -	
25/25 [=====] - 5s 67ms/step	
precision recall f1-score support	
0 0.95 0.95 0.95 40	
1 0.85 0.76 0.80 45	
2 0.78 0.86 0.82 36	
3 0.88 0.90 0.89 39	
4 0.85 0.87 0.86 39	
5 0.93 0.90 0.91 41	
6 0.88 0.90 0.89 39	
7 0.82 0.94 0.88 35	
8 0.90 0.92 0.91 39	
9 0.93 0.79 0.85 47	
accuracy	
macro avg 0.88 0.88 0.88 400	
weighted avg 0.88 0.88 0.87 400	

Gambar 4.7: Hasil Classification Report Model Google Colab GPU T4

- Accuracy: 0.88, menunjukkan bahwa model ini memiliki tingkat akurasi 88% di seluruh kelas.
- Macro Avg: Rata-rata precision, recall, dan f1-score untuk setiap kelas, tanpa mempertimbangkan jumlah instance di setiap kelas.
- Weighted Avg: Rata-rata precision, recall, dan f1-score untuk setiap kelas, dengan mempertimbangkan jumlah instance di setiap kelas.

Gambar 4.7 menunjukkan bahwa model memiliki performa yang baik di sebagian besar kelas, meskipun ada beberapa kelas yang memiliki nilai *recall* atau *precision* yang lebih rendah. Ada beberapa citra yang salah diklasifikasikan ke kelas lain, contohnya pada kelas 1 (bakso), dan kelas 9 (soto).

Setelah dilakukan *Classification Report*, untuk mempermudah evaluasi model dapat menggunakan *confusion matrix*. Berikut visualisasi menggunakan *confusion matrix*:



Gambar 4.8: Visualisasi Confusion Matrix Model Google Colab GPU T4

Confusion matrix ini menunjukkan hasil dari model klasifikasi yang telah diuji dengan beberapa kategori makanan Indonesia. Berikut adalah interpretasi dari setiap elemen dalam *confusion matrix* ini:

- *True label* (label sebenarnya) berada pada sumbu Y (vertikal).
- *Predicted label* (label prediksi) berada pada sumbu X (horizontal).

Setiap sel dalam matriks menunjukkan jumlah instance yang diprediksi dalam kategori tertentu. Misalnya, sel pada baris "ayam bakar" dan kolom "ayam bakar" menunjukkan jumlah instance "ayam bakar" yang diprediksi benar oleh model. Berikut adalah beberapa poin penting yang dapat diperhatikan dari *confusion matrix* pada Gambar 4.8:

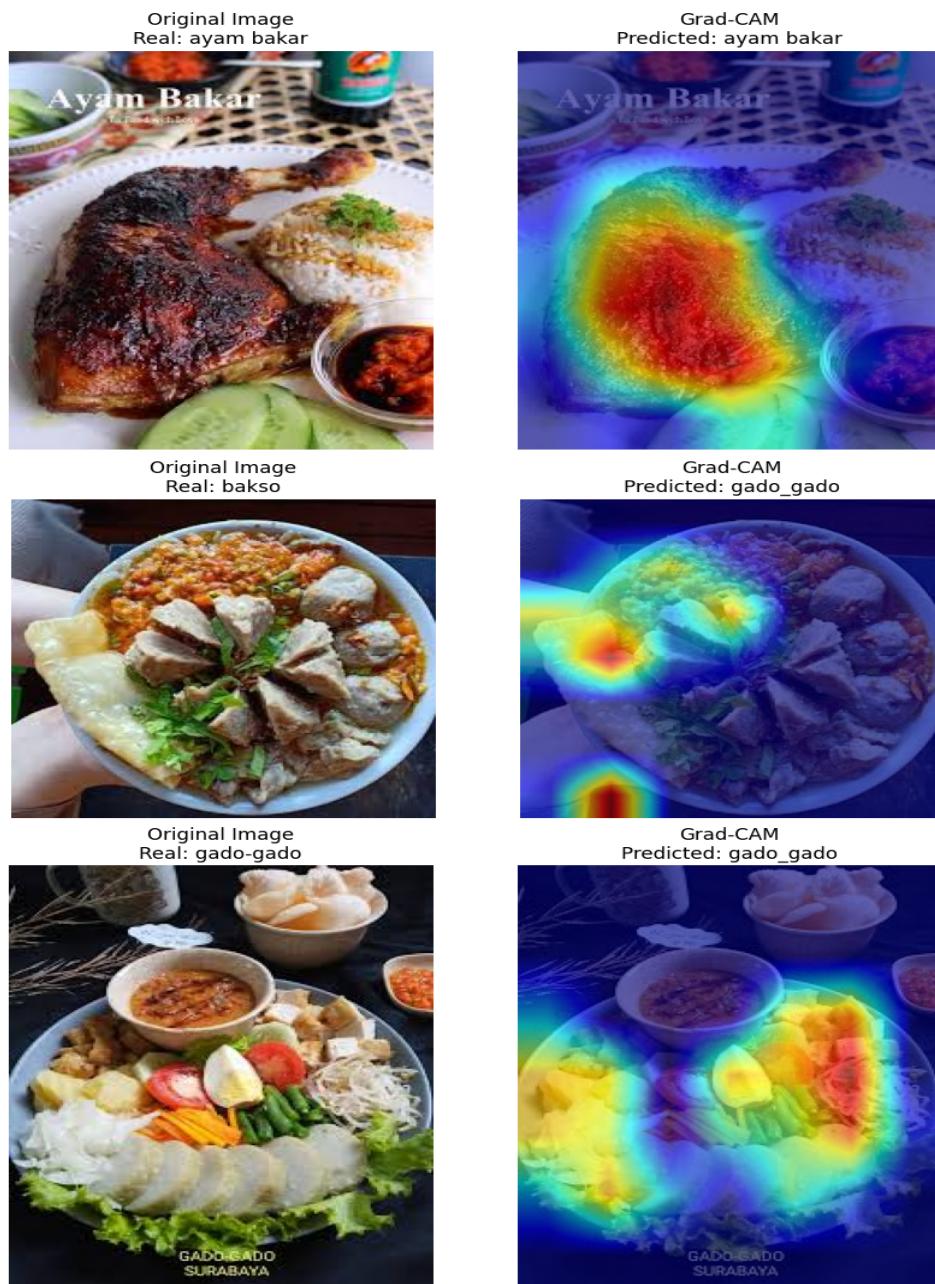
- **Ayam Bakar:** Dari 40 instance "ayam bakar", 38 diprediksi benar sebagai "ayam bakar", 1 diprediksi sebagai "gudeg", dan 1 diprediksi sebagai "rawon".

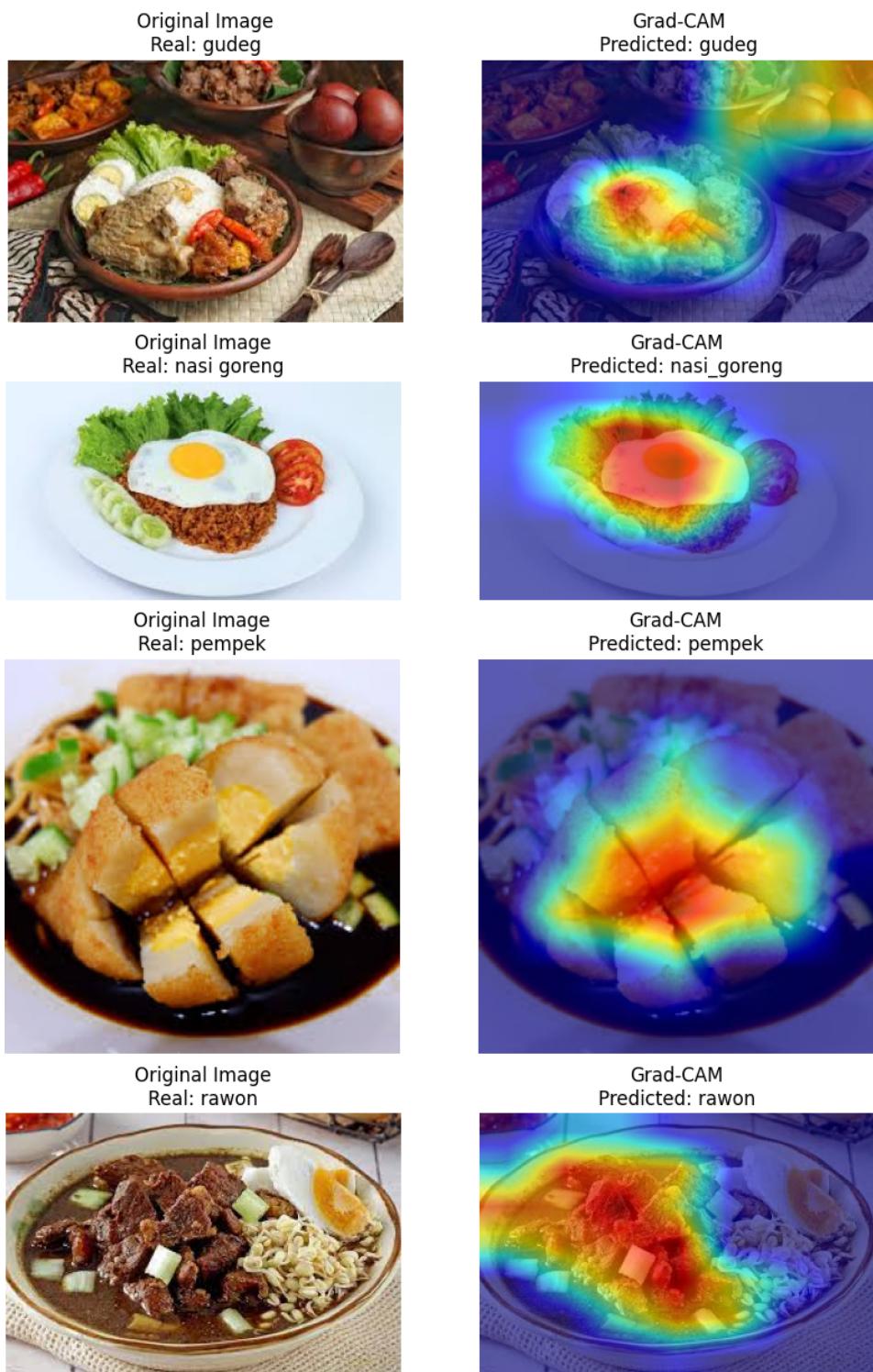
- **Bakso:** Dari 40 instance "bakso", 34 diprediksi benar sebagai "bakso", namun ada beberapa yang salah prediksi ke "gado-gado", "pempek", dan "soto".
- **Gado-gado:** Dari 40 instance "gado-gado", 31 diprediksi benar, namun ada beberapa yang salah prediksi ke "bakso", "gudeg", "nasi goreng", "pempek", dan "soto".
- **Gudeg:** Dari 40 instance "gudeg", 35 diprediksi benar, namun ada beberapa yang salah prediksi ke "gado-gado", "rawon", "rendang", dan "sate".
- **Nasi Goreng:** Dari 40 instance "nasi goreng", 34 diprediksi benar, namun ada beberapa yang salah prediksi ke "bakso", "gado-gado", dan "pempek", dan "soto".
- **Pempek:** Dari 40 instance "pempek", 37 diprediksi benar, namun ada beberapa yang salah prediksi ke "bakso" dan "nasi goreng".
- **Rawon:** Dari 40 instance "rawon", 35 diprediksi benar, namun ada beberapa yang salah prediksi ke "bakso", "rendang", "sate", dan "soto".
- **Rendang:** Dari 40 instance "rendang", 33 diprediksi benar, namun ada beberapa yang salah prediksi ke "ayam bakar", "bakso", , "gudeg", "nasi goreng", "rawon", dan "sate".
- **Sate:** Dari 40 instance "sate", 36 diprediksi benar, namun ada beberapa yang salah prediksi ke "ayam bakar", "pempek", dan "soto".
- **Soto:** Dari 40 instance "soto", 37 diprediksi benar, namun ada beberapa yang salah prediksi ke "bakso", "nasi goreng", dan "rawon".

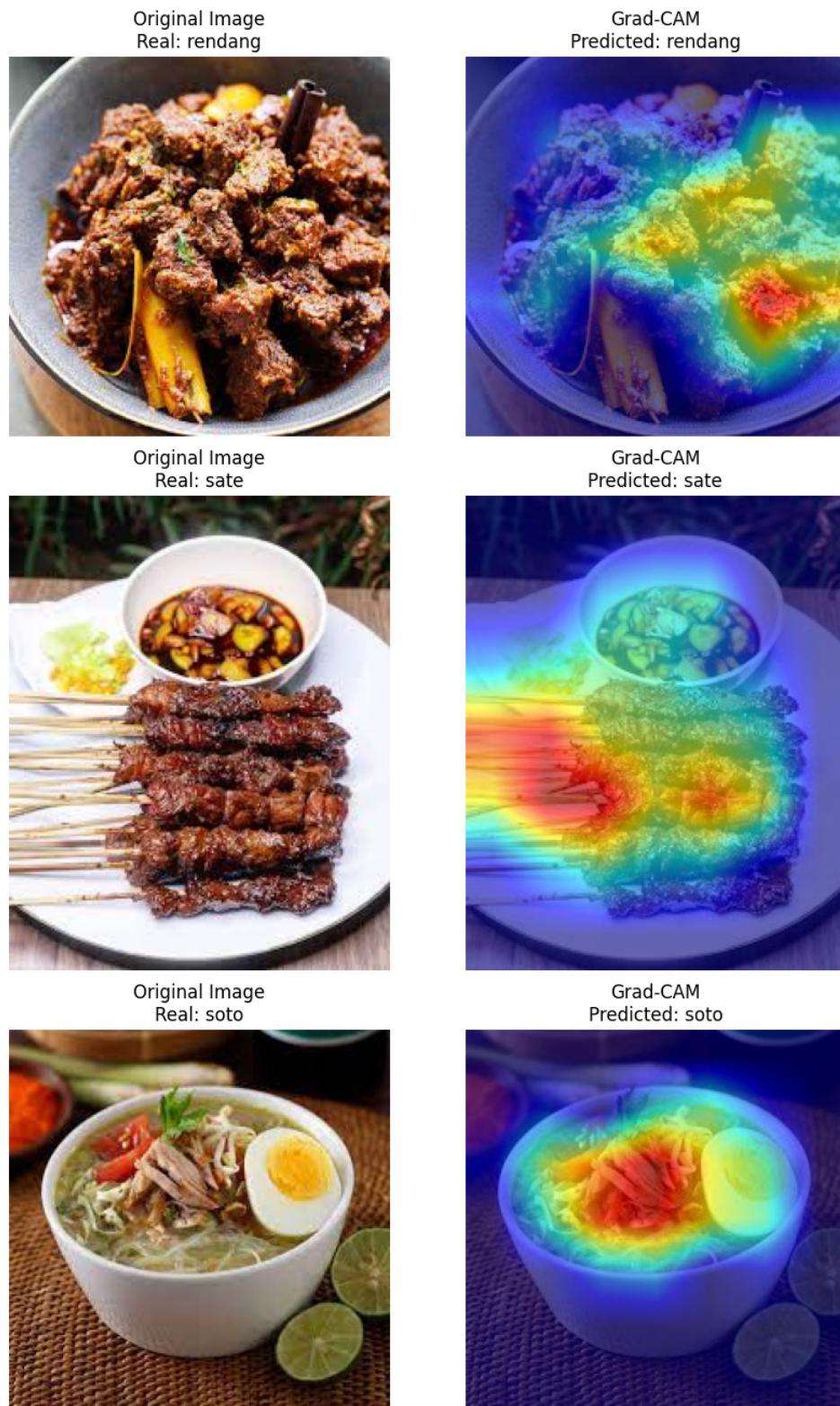
Secara keseluruhan, model menunjukkan performa yang cukup baik dengan sebagian besar prediksi berada di diagonal utama, yang berarti banyak instance yang diklasifikasikan dengan benar. Namun, ada beberapa kesalahan prediksi yang perlu diperhatikan untuk meningkatkan akurasi model lebih lanjut.

4.4.1.2 Hasil Visualisasi Grad-CAM

Setelah melakukan proses Evaluasi Model, selanjutnya dilakukan proses Visualisasi Grad-CAM. Hal ini dilakukan untuk mengetahui bagaimana model melihat citra dan area mana yang dianggap penting oleh model dalam membuat prediksi. Berikut hasil visualisasi menggunakan Grad-CAM:







Gambar 4.9: Hasil Visualisasi Grad-CAM Model Google Colab GPU T4

Pada Gambar 4.9, terlihat bahwa model mampu mengklasifikasikan 9 dari 10 citra dengan tepat. Namun, terdapat satu kelas yang memerlukan perhatian lebih lanjut, yaitu kelas bakso. Dalam visualisasi tersebut, kelas bakso diklasifikasikan sebagai gado-gado. Hal ini menunjukkan bahwa model mengalami kesulitan dalam membedakan antara kedua jenis makanan tersebut.

4.4.1.3 Hasil Pengujian Model menggunakan data nutrisi makanan

Setelah memastikan model memiliki performa yang bagus, selanjutnya dilakukan uji coba menggunakan data nutrisi makanan. Makanan yang berhasil diklasifikasikan kemudian akan ditampilkan informasi nutrisi pada makanan tersebut. Berikut hasil dari uji coba model menggunakan data nutrisi makanan:

```
1/1 [=====] - 0s 51ms/step
-----
      Nama makanan: ayam bakar
      Energi (kkal): 226
      Protein (g): 23.0
      Lemak (g): 15.0
      Karbo (g): 0.1
  Jumlah per sajian (g): 100
-----
ayam bakar (99.97%)

```



Gambar 4.10: Hasil Pengujian Model Google Colab GPU T4 menggunakan data nutrisi makanan

Pada Gambar 4.10 terlihat bahwa model dapat digunakan dan informasi nutrisi dapat ditampilkan dengan baik. Citra juga berhasil diklasifikasikan dengan tepat, citra pertama berhasil diklasifikasikan sebagai ayam bakar dengan *confidence* sebesar 99.97%, kemudian citra kedua berhasil diklasifikasikan sebagai sate dengan *confidence* sebesar 99.92%.

4.4.2 Hasil Pengujian Model mesin DGX A100 Universitas Gunadarma

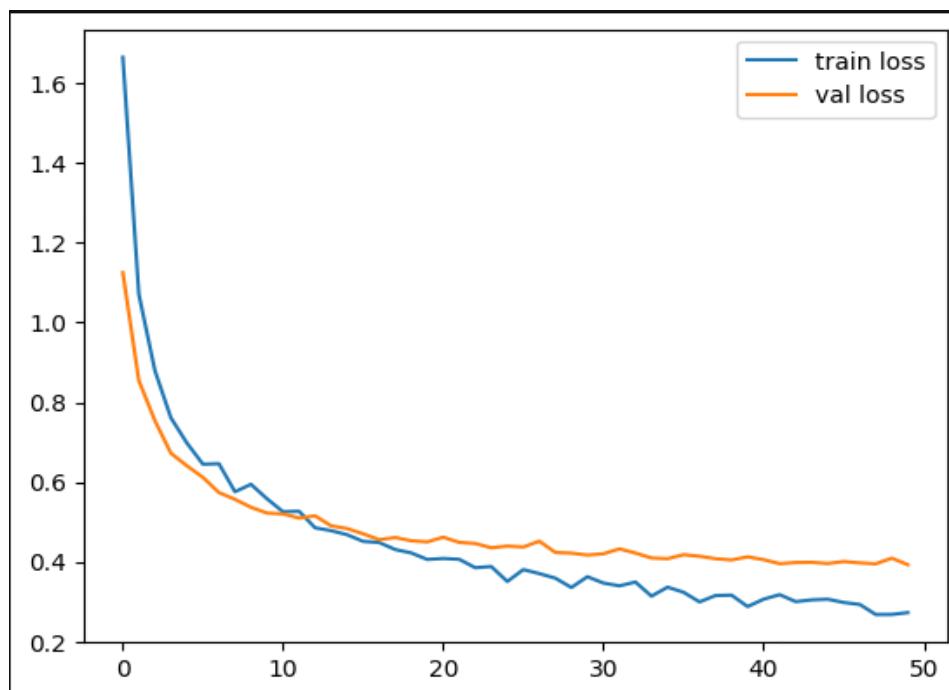
Hasil Pengujian Model mesin DGX A100 Universitas Gunadarma dibagi menjadi 3 bagian, yaitu:

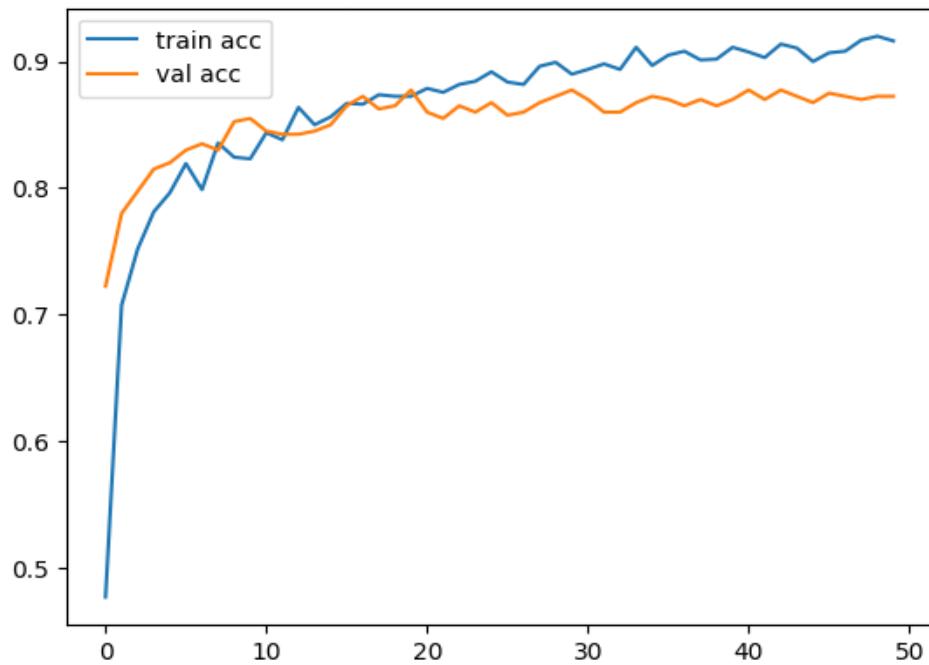
1. Hasil Evaluasi Model
2. Hasil Visualisasi Grad-CAM

3. Hasil Pengujian Model menggunakan data nutrisi makanan

4.4.2.1 Hasil Evaluasi Model

Tahap pertama pada Evaluasi Model yaitu untuk mengetahui model tersebut termasuk *overfit*, *good fit* atau *underfit*. Untuk mengetahui hal tersebut, dapat dilakukan dengan membuat visualisasi dari hasil pelatihan model dengan diagram garis. Berikut hasil visualisasi pelatihan model dengan diagram garis:





Gambar 4.11: Visualisasi Pelatihan Model mesin DGX A100 Universitas Gunadarma

Pada Gambar 4.11, terlihat *train loss* dan *val loss* mendekati satu sama lain dan terus menurun hingga datar dari beberapa titik hingga akhir. Kemudian, *train acc* dan *val acc* juga mendekati satu sama lain. Hal ini dapat diartikan bahwa model tersebut termasuk *good fit* dan layak dipakai.

Setelah mengetahui bahwa model layak dipakai, kemudian dilakukan *Classification Report* untuk mengetahui performa model berdasarkan metriks *precision*, *recall*, dan *F1-Score*. Berikut hasil dari *Classification Report* menggunakan *library* dari *sklearn*:

25/25 [=====]	-	3s	25ms/step -	
25/25 [=====]	-	3s	22ms/step	
	precision	recall	f1-score	support
0	0.95	0.95	0.95	40
1	0.85	0.87	0.86	39
2	0.85	0.77	0.81	44
3	0.85	0.89	0.87	38
4	0.80	0.82	0.81	39
5	0.90	0.90	0.90	40
6	0.88	0.90	0.89	39
7	0.82	0.89	0.86	37
8	0.85	1.00	0.92	34
9	0.97	0.78	0.87	50
accuracy			0.87	400
macro avg	0.87	0.88	0.87	400
weighted avg	0.88	0.87	0.87	400

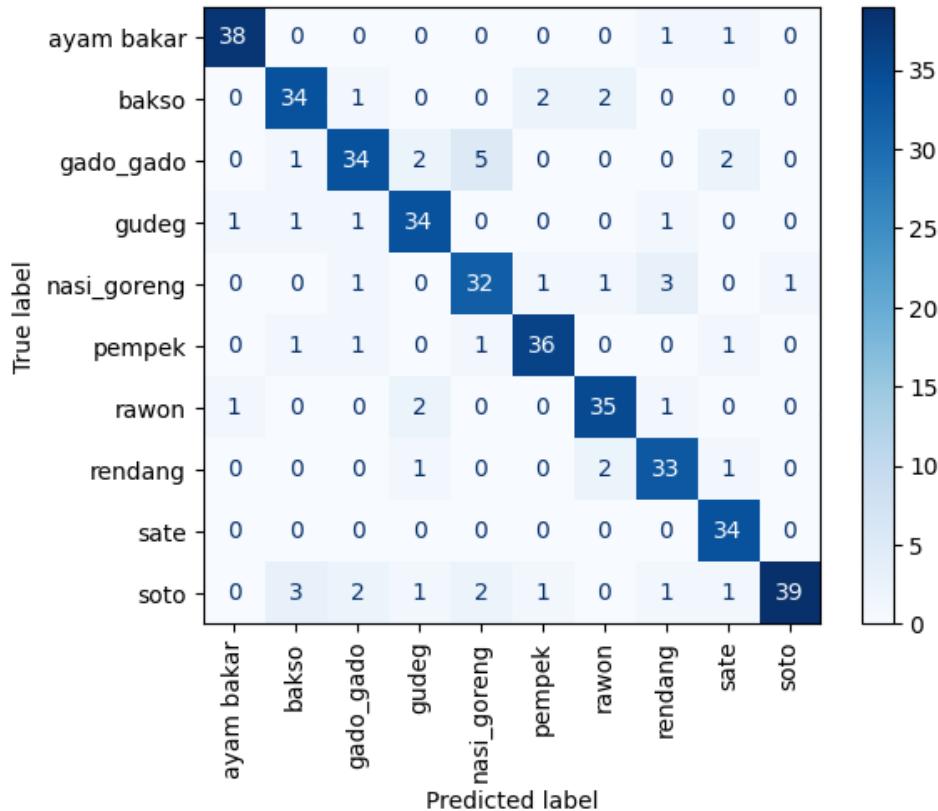
Gambar 4.12: Hasil Classification Report Model mesin DGX A100 Universitas Gunadarma

- Accuracy: 0.87, menunjukkan bahwa model ini memiliki tingkat akurasi 88% di seluruh kelas.
- Macro Avg: Rata-rata precision, recall, dan f1-score untuk setiap kelas, tanpa mempertimbangkan jumlah instance di setiap kelas.
- Weighted Avg: Rata-rata precision, recall, dan f1-score untuk setiap kelas, dengan mempertimbangkan jumlah instance di setiap kelas.

Gambar 4.12 menunjukkan bahwa model memiliki performa yang baik di sebagian besar kelas, meskipun ada beberapa kelas yang memiliki nilai *recall* atau *precision* yang lebih rendah. Ada beberapa citra yang salah diklasifikasikan ke kelas lain, contohnya pada kelas 2 (gado-gado) dan kelas 9 (soto).

Setelah dilakukan *Classification Report*, untuk mempermudah evaluasi model dapat menggunakan *confusion matrix*. Berikut visualisasi menggunakan *confusion matrix*:

Confusion matrix ini menunjukkan hasil dari model klasifikasi yang telah diuji dengan beberapa kategori makanan Indonesia. Berikut adalah interpretasi dari setiap elemen dalam *confusion matrix* ini:



Gambar 4.13: Visualisasi Confusion Matrix Model mesin DGX A100 Universitas Gunadarma

- *True label* (label sebenarnya) berada pada sumbu Y (vertikal).
- *Predicted label* (label prediksi) berada pada sumbu X (horizontal).

Setiap sel dalam matriks menunjukkan jumlah instance yang diprediksi dalam kategori tertentu. Misalnya, sel pada baris "ayam bakar" dan kolom "ayam bakar" menunjukkan jumlah instance "ayam bakar" yang diprediksi benar oleh model. Berikut adalah beberapa poin penting yang dapat diperhatikan dari *confusion matrix* pada Gambar 4.13:

- **Ayam Bakar:** Dari 40 instance "ayam bakar", 38 diprediksi benar sebagai "ayam bakar", 1 diprediksi sebagai "gudeg", dan 1 diprediksi sebagai "rawon".
- **Bakso:** Dari 40 instance "bakso", 34 diprediksi benar sebagai "bakso", 1 diprediksi sebagai "nasi goreng", 1 diprediksi sebagai "gado-gado", 1 diprediksi sebagai "gudeg", dan 3 diprediksi sebagai "soto".

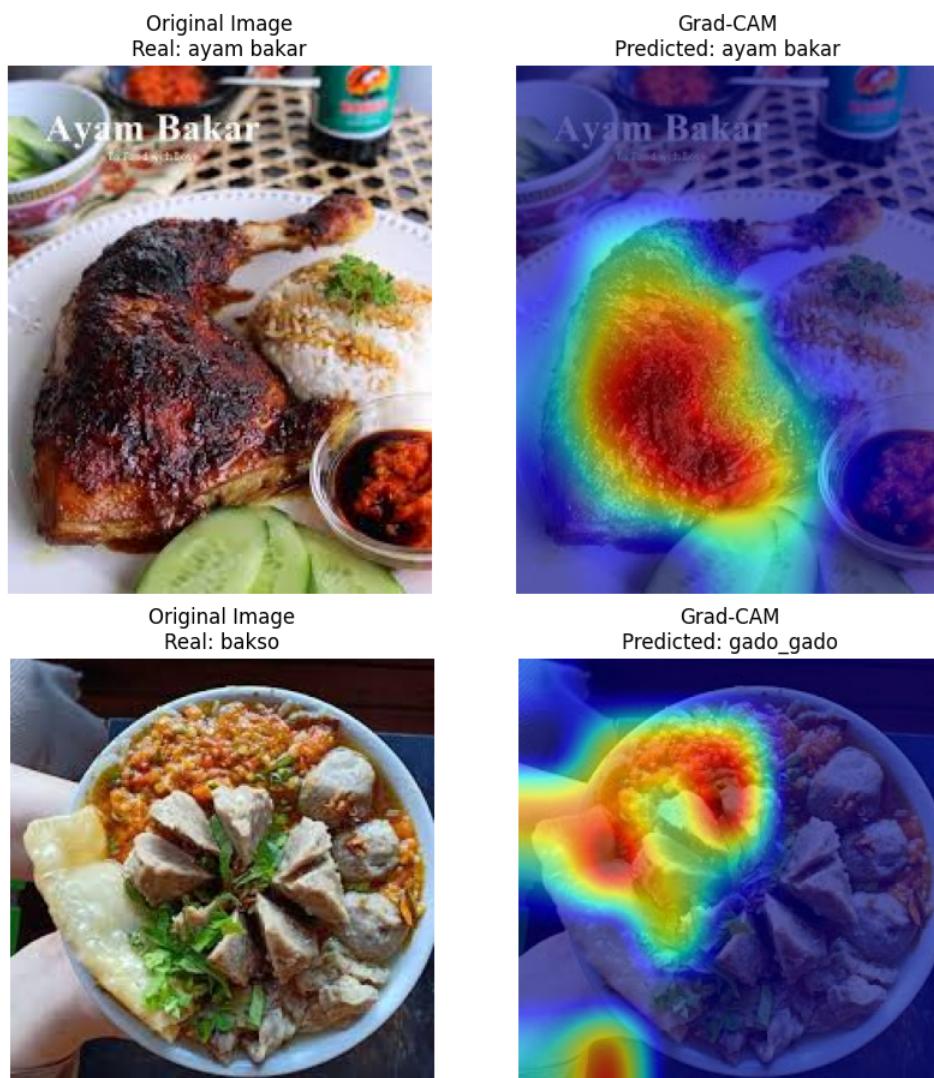
- **Gado-gado:** Dari 40 instance "gado-gado", 34 diprediksi benar sebagai "gado-gado", 1 diprediksi sebagai "bakso", 1 diprediksi sebagai "gudeg", 1 diprediksi sebagai "nasi goreng", 1 diprediksi sebagai "pempek", dan 2 diprediksi sebagai "soto".
- **Gudeg:** Dari 40 instance "gudeg", 34 diprediksi benar sebagai "gudeg", 2 diprediksi sebagai "gado-gado", 2 diprediksi sebagai "rawon", 1 diprediksi sebagai "rendang", dan 1 diprediksi sebagai "soto".
- **Nasi Goreng:** Dari 40 instance "nasi goreng", 32 diprediksi benar sebagai "nasi goreng", 5 diprediksi sebagai "gado-gado", 1 diprediksi sebagai "pempek", dan 2 diprediksi sebagai "soto".
- **Pempek:** Dari 40 instance "pempek", 36 diprediksi benar sebagai "pempek", 2 diprediksi sebagai "bakso", 1 diprediksi sebagai "nasi goreng", dan 1 diprediksi sebagai "soto".
- **Rawon:** Dari 40 instance "rawon", 35 diprediksi benar sebagai "rawon", 2 diprediksi sebagai "bakso", 1 diprediksi sebagai "nasi goreng", dan 2 diprediksi sebagai "rendang".
- **Rendang:** Dari 40 instance "rendang", 33 diprediksi benar sebagai "rendang", 1 diprediksi sebagai "ayam bakar", 1 diprediksi sebagai "gudeg", 3 diprediksi sebagai "nasi goreng", 1 diprediksi sebagai "rawon", dan 1 diprediksi sebagai "soto".
- **Sate:** Dari 40 instance "sate", 34 diprediksi benar sebagai "sate", 1 diprediksi sebagai "ayam bakar", 2 diprediksi sebagai "gado-gado", 1 diprediksi sebagai "pempek", 1 diprediksi sebagai "rendang", dan 1 diprediksi sebagai "soto".
- **Soto:** Dari 40 instance "soto", 39 diprediksi benar sebagai "soto", 1 diprediksi sebagai "nasi goreng".

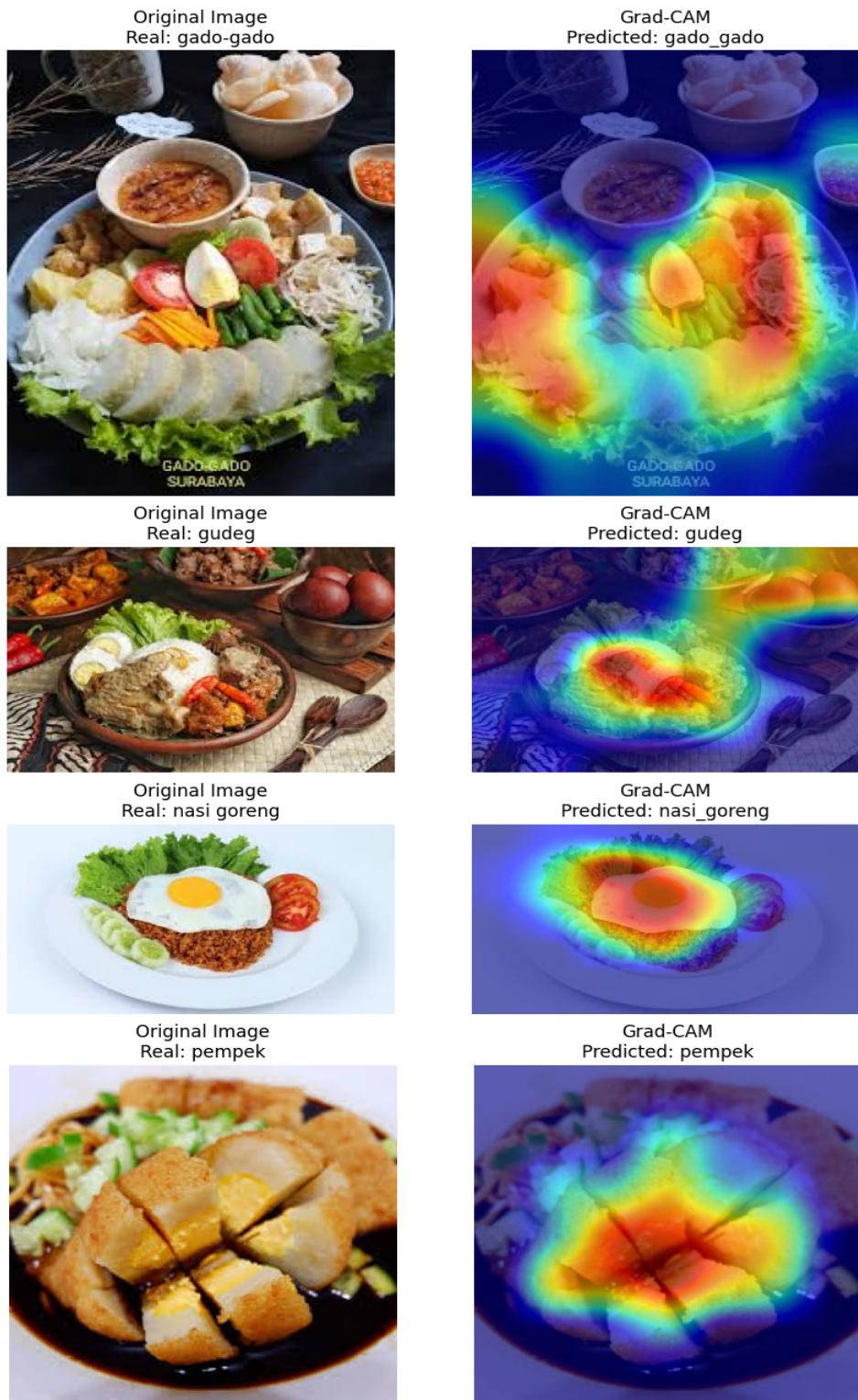
Secara keseluruhan, model menunjukkan performa yang cukup baik dengan sebagian besar prediksi berada di diagonal utama, yang berarti banyak instance yang

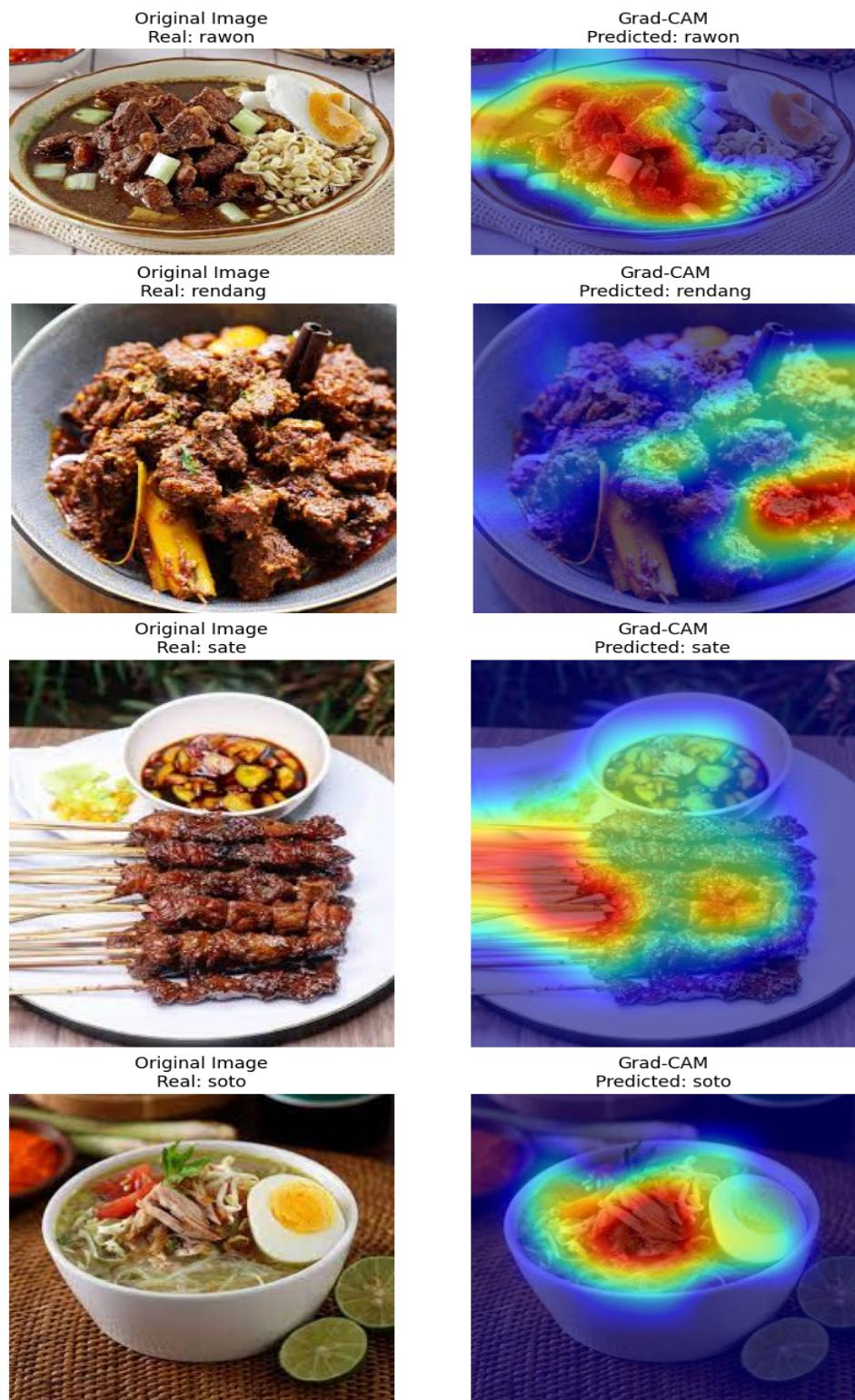
diklasifikasikan dengan benar. Namun, ada beberapa kesalahan prediksi yang perlu diperhatikan untuk meningkatkan akurasi model lebih lanjut.

4.4.2.2 Hasil Visualisasi Grad-CAM

Setelah melakukan proses Evaluasi Model, selanjutnya dilakukan proses Visualisasi Grad-CAM. Hal ini dilakukan untuk mengetahui bagaimana model melihat citra dan area mana yang dianggap penting oleh model dalam membuat prediksi. Berikut hasil visualisasi menggunakan Grad-CAM:







Gambar 4.14: Hasil Visualisasi Grad-CAM Model mesin DGX A100 Universitas Gunadarma

Pada Gambar 4.14, terlihat bahwa model mampu mengklasifikasikan 9 dari 10 citra dengan tepat. Namun, terdapat satu kelas yang memerlukan perhatian lebih lanjut, yaitu kelas bakso. Dalam visualisasi tersebut, kelas bakso diklasifikasikan sebagai gado-gado. Hal ini menunjukkan bahwa model mengalami kesulitan dalam membedakan antara kedua jenis makanan tersebut.

4.4.2.3 Hasil Pengujian Model menggunakan data nutrisi makanan

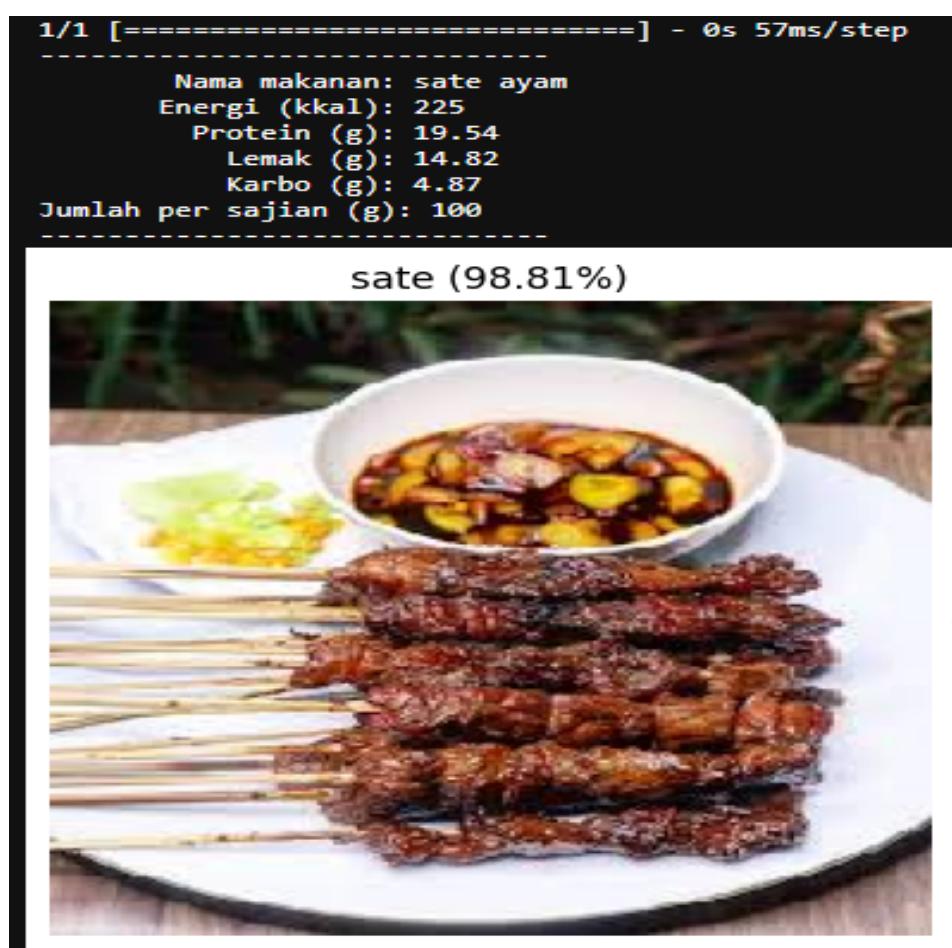
Setelah memastikan model memiliki performa yang bagus, selanjutnya dilakukan uji coba menggunakan data nutrisi makanan. Makanan yang berhasil diklasifikasikan kemudian akan ditampilkan informasi nutrisi pada makanan tersebut. Berikut hasil dari uji coba model menggunakan data nutrisi makanan:

1/1 [=====] - 0s 66ms/step

```
-----  
Nama makanan: ayam bakar  
Energi (kkal): 226  
Protein (g): 23.0  
Lemak (g): 15.0  
Karbo (g): 0.1  
Jumlah per sajian (g): 100
```

ayam bakar (99.96%)





Gambar 4.15: Hasil Pengujian Model mesin DGX A100 Universitas Gunadarma menggunakan data nutrisi makanan

Pada Gambar 4.15 terlihat bahwa model dapat digunakan dan informasi nutrisi dapat ditampilkan dengan baik. Citra juga berhasil diklasifikasikan dengan tepat, citra pertama berhasil diklasifikasikan sebagai ayam bakar dengan *confidence* sebesar 99.96%, kemudian citra kedua berhasil diklasifikasikan sebagai sate dengan *confidence* sebesar 98.81%.

BAB V

PENUTUP

5.1 Kesimpulan

Berdasarkan hasil penelitian yang telah dilakukan untuk identifikasi nilai nutrisi pada makanan populer di Indonesia berbasis citra menggunakan *Convolutional Neural Network* (CNN) dengan arsitektur EfficientNetV2, dapat disimpulkan bahwa:

1. Model *deep learning* berbasis arsitektur EfficientNetV2 berhasil mengklasifikasikan citra makanan dan menampilkan estimasi nilai nutrisinya.
2. Model *deep learning* berbasis arsitektur EfficientNetV2 menggunakan Google Colab dengan GPU T4 berhasil mendapatkan nilai *accuracy* sebesar 91,12% pada data latih dan berhasil mendapatkan nilai *accuracy* sebesar 87,50% pada data validasi, model juga memiliki rata-rata *precision*, *recall*, dan *F1-Score* sebesar 88%. Kemudian pengujian model menggunakan DGX A100 berhasil mendapatkan nilai *accuracy* sebesar 91,62% pada data latih dan berhasil mendapatkan nilai *accuracy* sebesar 87,25% pada data validasi, model juga memiliki rata-rata *precision* sebesar 87%, *recall* sebesar 88%, dan *F1-Score* sebesar 87%.
3. Mesin DGX A100 terbukti dapat melakukan pelatihan model 2x lebih cepat dari Google Colab dengan GPU T4 pada dataset penelitian ini. Model yang dilatih menggunakan Google Colab dengan GPU T4 dengan model yang dilatih menggunakan DGX A100 mempunyai performa dan akurasi yang hampir sama dan tidak berbeda secara signifikan.

5.2 Saran

Saran yang dapat diberikan pada penelitian berikutnya adalah menggunakan arsitektur CNN selain EfficientNetV2, seperti InceptionV3, ResNet, ataupun arsitektur lainnya. Diharapkan penelitian selanjutnya dapat menggunakan teknik *image recognition* agar dapat mengidentifikasi banyak makanan dalam satu citra sehingga

informasi nutrisi makanan lebih akurat. Penelitian ini diharapkan juga dapat dikembangkan lagi untuk diintegrasikan ke aplikasi agar lebih mudah digunakan.

DAFTAR PUSTAKA

- Abbass, Hussein (2021). “Editorial: What is Artificial Intelligence?” In: *IEEE Transactions on Artificial Intelligence* 2.2, pp. 94–95. doi: 10.1109/TAI.2021.3096243.
- Alom, Md Zahangir et al. (2018). *The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches*. arXiv: 1803.01164 [cs.CV]. URL: <https://arxiv.org/abs/1803.01164>.
- Andono, Pulung Nurtantio, T. Sutojo, and Muljono (2017). *Pengolahan citra digital*. Ed. by Arie Pramesta. Yogyakarta: ANDI.
- BISA AI (2024). *Scikit-learn: AI For Everyone*. URL: <https://bisa.ai/course/detail/MzU3/1> (visited on 07/16/2024).
- Bisong, Ekaba (Jan. 2019). *Building Machine Learning and Deep Learning Models on Google Cloud Platform: A Comprehensive Guide for Beginners*. Apress Berkeley, CA. ISBN: 978-1-4842-4469-2. doi: 10.1007/978-1-4842-4470-8.
- Bock, Sebastian and Martin Weiß (2019). “A Proof of Local Convergence for the Adam Optimizer”. In: *2019 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. doi: 10.1109/IJCNN.2019.8852239.
- Chicho, Bahzad and Amira Sallow (Oct. 2021). “A Comprehensive Survey of Deep Learning Models Based on Keras Framework”. In: *Journal of Soft Computing and Data Mining* 2. doi: 10.30880/jscdm.2021.02.02.005.
- cmlabs (2023). *Flowchart: Definition, Functions, Types, and Symbols*. URL: <https://cmlabs.co/en-id/seo-guidelines/flowchart> (visited on 07/16/2024).
- Corporation, NVIDIA (2021). *NVIDIA DGX Systems*. URL: <https://www.nvidia.com/en-us/data-center/dgx-systems/> (visited on 07/31/2024).
- Desai, Parul (2019). *Learning Curve to Identify Overfitting & Underfitting Problems*. URL: <https://towardsdatascience.com/learning-curve-to-identify-overfitting-underfitting-problems-133177f38df5> (visited on 07/31/2024).
- Fitria, Fitria, Nia Musniati, and Devi Annisa Mulyawati (2022). “Gambaran Tingkat Pengetahuan tentang Gizi Seimbang pada Siswa SMA Muhammadiyah 13

- Jakarta”. In: *Muhammadiyah Journal of Nutrition and Food Science (MJNF)* 3.1, pp. 11–16.
- Géron, Aurélien (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. 2nd. O'Reilly Media, Inc. ISBN: 9781492032649.
- Git (2024). *About Git*. URL: <https://git-scm.com/about> (visited on 07/31/2024).
- Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). *Deep Learning*. MIT Press.
- Gunadarma, HPC Hub (2024). *HPC Hub Universitas Gunadarma*. URL: <https://www.hpc-hub.gunadarma.ac.id/> (visited on 07/31/2024).
- Ioffe, Sergey and Christian Szegedy (2015). *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. arXiv: 1502.03167 [cs.LG]. URL: <https://arxiv.org/abs/1502.03167>.
- Jupyter (2021). *Jupyter Notebook*. URL: <https://jupyter.org> (visited on 07/31/2024).
- Karthik, R. et al. (2022). “Eff2Net: An Efficient Channel Attention-based Convolutional Neural Network for Skin Disease Classification”. In: *Biomed Signal Process Control* 73. doi: 10.1016/j.bspc.2021.103406.
- Kelleher, John D. (Sept. 2019). *Deep Learning*. The MIT Press. ISBN: 9780262354899. doi: 10.7551/mitpress/11171.001.0001. URL: <https://doi.org/10.7551/mitpress/11171.001.0001>.
- Mitchell, Ryan (2018). *Web Scraping with Python: Collecting More Data from the Modern Web*. 2nd. O'Reilly Media. ISBN: 978-1491985571.
- Moolayil, Jojo (Jan. 2019). *Learn Keras for Deep Neural Networks: A Fast-Track Approach to Modern Deep Learning with Python*. ISBN: 978-1-4842-4239-1. doi: 10.1007/978-1-4842-4240-7.
- Naik, Prathamesh (2023). “How to used Google Colab for Deep Learning using Python?” In: *All about Machine Learning*. URL: <https://blog.techcraft.org/how-to-used-google-colab-for-deep-learning-using-python/> (visited on 07/16/2024).
- Pang, Bo, Erik Nijkamp, and Ying Nian Wu (2020). “Deep Learning With TensorFlow: A Review”. In: *Journal of Educational and Behavioral Statistics* 45.2, pp. 227–248. doi: 10.3102/1076998619872761. eprint: <https://doi.org/10.3102/1076998619872761>. URL: <https://doi.org/10.3102/1076998619872761>.

- Rajayogi, J R, G Manjunath, and G Shobha (2019). “Indian Food Image Classification with Transfer Learning”. In: *2019 4th International Conference on Computational Systems and Information Technology for Sustainable Solution (CSITSS)*, pp. 1–4. doi: 10.1109/CSITSS47250.2019.9031051.
- Rebala, Gopinath, Ajay Ravi, and Sanjay Churiwala (Jan. 2019). *An Introduction to Machine Learning*. ISBN: 978-3-030-15728-9. doi: 10.1007/978-3-030-15729-6.
- Sarker, Iqbal H (2021). “Machine Learning: Algorithms, Real-World Applications and Research Directions”. In: *SN Computer Science* 2 (3), p. 160. issn: 2661-8907. doi: 10.1007/s42979-021-00592-x. URL: <https://doi.org/10.1007/s42979-021-00592-x>.
- Selvaraju, Ramprasaath R. et al. (2017). “Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization”. In: *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 618–626. doi: 10.1109/ICCV.2017.74.
- Setiawan, Gabriella Alicia and Evelyn Vania (Apr. 2022). *Praktek Pemrograman C++ dan Python*. Jakarta: SCU Knowledge Media.
- Sun, Shiliang et al. (2019). *A Survey of Optimization Methods from a Machine Learning Perspective*. arXiv: 1906.06821 [cs.LG]. URL: <https://arxiv.org/abs/1906.06821>.
- Tan, Mingxing and Quoc V. Le (2020). *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*. arXiv: 1905.11946 [cs.LG]. URL: <https://arxiv.org/abs/1905.11946>.
- (2021). *EfficientNetV2: Smaller Models and Faster Training*. arXiv: 2104.00298 [cs.CV]. URL: <https://arxiv.org/abs/2104.00298>.
- Tian, Youhui (2020). “Artificial Intelligence Image Recognition Method Based on Convolutional Neural Network Algorithm”. In: *IEEE Access* 8, pp. 125731–125744. doi: 10.1109/ACCESS.2020.3006097.
- Vasilev, Ivan et al. (2019). *Python Deep Learning, Second Edition*. Birmingham: Packt Publishing Ltd.
- Yadav, H. (2022). “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”. In: *Journal of Machine Learning Research* 21.1, pp. 1–30.

- Yamashita, Rikiya et al. (2018). “Convolutional neural networks: an overview and application in radiology”. In: *Insights into Imaging* 9 (4), pp. 611–629. issn: 1869-4101. doi: 10.1007/s13244-018-0639-9. url: <https://doi.org/10.1007/s13244-018-0639-9>.
- Zhuang, Fuzhen et al. (2020). *A Comprehensive Survey on Transfer Learning*. arXiv: 1911.02685 [cs.LG]. url: <https://arxiv.org/abs/1911.02685>.

LAMPIRAN

Listing Program

a. Import Library

```
import os
import tensorflow as tf
from keras.preprocessing.image import ImageDataGenerator
from random import sample
import matplotlib.pyplot as plt
import numpy as np
import cv2
import splitfolders
from tensorflow.keras.applications import EfficientNetV2S
from keras.layers import Input
from keras import layers
from tensorflow.keras.callbacks import ModelCheckpoint,
ReduceLROnPlateau
import time
from tensorflow.keras.applications.efficientnet_v2
import preprocess_input
from sklearn.metrics import accuracy_score,
classification_report, confusion_matrix,
ConfusionMatrixDisplay
```

b. Data Preparation

```
# Split Data
splitfolders.ratio('/content/dataset', output="split_images",
seed=1337, ratio=(.8,.2))
```

```
# Define train and val dataset
base_dir = '/content/split_images'
```

```
train_dir = os.path.join(base_dir, 'train')
val_dir = os.path.join(base_dir, 'val')
```

```
# Define class names and n of classes
class_names = sorted(
os.listdir('/content/split_images/train'))
n_classes = len(class_names)

# Print
print("No. Classes : {}".format(n_classes))
print("Classes      : {}".format(class_names))
```

c. Data Preprocessing

```
# Normalization and Augmentation
batch_size = 16

train_datagen = ImageDataGenerator(
    preprocessing_function=preprocess_input,
    rotation_range = 40,
    shear_range = 0.2,
    zoom_range = 0.2,
    horizontal_flip = True,
    vertical_flip = True)

val_datagen = ImageDataGenerator(
    preprocessing_function=preprocess_input)

train_dataset = train_datagen.flow_from_directory(
    train_dir,
    shuffle=True,
    target_size=(224, 224),
    batch_size=batch_size,
```

```

        class_mode='categorical')

val_dataset = val_datagen.flow_from_directory(
    val_dir,
    shuffle=False,
    target_size=(224, 224),
    batch_size=batch_size,
    class_mode='categorical')

```

d. Pembuatan Model Transfer Learning EfficientNetV2

```

# Initialize EfficientNetV2 with imagenet weights
pre_trained_model = EfficientNetV2S(weights='imagenet',
include_top=False, input_tensor=Input(shape=(224, 224, 3)))

# Freeze all layers
pre_trained_model.trainable = False

# Fine-tune model
x = pre_trained_model.output
x = layers.GlobalAveragePooling2D()(x)
outputs = layers.Dense(n_classes, activation='softmax')(x)

model = tf.keras.models.Model(
    pre_trained_model.input, outputs)

model.summary()

```

e. Pelatihan Model

```

# Compile model
model.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=0.001),
    loss='categorical_crossentropy',

```

```
    metrics=['accuracy']
)
```

```
# Callback checkpointer
checkpointer = ModelCheckpoint(
    filepath='best_model.h5',
    monitor='val_loss',
    verbose=1,
    save_best_only=True,
    mode='min'
)
```

```
# Train
num_epochs = 50
start_time = time.time()

history = model.fit(train_dataset,
    epochs=num_epochs,
    validation_data=val_dataset,
    callbacks=[checkpointer]
)

elapsed_time = time.time() - start_time
time.strftime("%H:%M:%S", time.gmtime(elapsed_time))
```

f. Evaluasi Model

```
# Visualisasi Pelatihan
def report_train(history):
    # loss
    plt.plot(history.history['loss'],label='train loss')
    plt.plot(history.history['val_loss'],label='val loss')
    plt.legend()
    plt.show()
```

```

# accuracies
plt.plot(history.history['accuracy'],label='train acc')
plt.plot(history.history['val_accuracy'],label='val acc')
plt.legend()
plt.show()

report_train(history)

```

```

# Classification Report and Confussion Matrix
def report_test(test_set,model=model):

    # evaluating test
    model.evaluate(test_set,batch_size=32)
    y_pred=model.predict(test_set)
    y_pred=np.argmax(y_pred, axis=1)
    accuracy_score(y_pred,test_set.classes)
    print(classification_report(y_pred,test_set.classes))

    # confusion_matrix
    labels = [i for i in train_dataset.class_indices]
    cm =confusion_matrix(y_pred, test_set.classes)
    disp = ConfusionMatrixDisplay(cm,display_labels=labels)
    disp.plot(cmap='Blues',xticks_rotation='vertical')
    plt.show()

report_test(val_dataset)

```

```

# Classification Report and Confussion Matrix best model
from tensorflow.keras.models import load_model
best_model = load_model('best_model.h5')
report_test(val_dataset, best_model)

```

g. Visualisasi Grad-CAM

```

# Grad-CAM heatmap

def get_gradcam_heatmap(model, img_array,
last_conv_layer_name, pred_index=None):
    grad_model = tf.keras.models.Model(
        [model.inputs],
        [model.get_layer(last_conv_layer_name).output,
         model.output]
    )

    with tf.GradientTape() as tape:
        conv_layer_output,
        predictions = grad_model(img_array)
        if pred_index is None:
            pred_index = tf.argmax(predictions[0])
        class_channel = predictions[:, pred_index]

        grads = tape.gradient(class_channel, conv_layer_output)
        pooled_grads = tf.reduce_mean(grads, axis=(0, 1, 2))

        conv_layer_output = conv_layer_output[0]
        heatmap = conv_layer_output @
        pooled_grads[..., tf.newaxis]
        heatmap = tf.squeeze(heatmap)

        heatmap = tf.maximum(heatmap, 0) /
        tf.math.reduce_max(heatmap)
        heatmap = heatmap.numpy()
    return heatmap

```

```

# Display Grad-CAM with labels

def display_gradcam_with_labels(img_path,
heatmap, real_label, predicted_label, alpha=0.4):
    img = cv2.imread(img_path)

```

```

heatmap = cv2.resize(heatmap,
(img.shape[1], img.shape[0]))
heatmap = np.uint8(255 * heatmap)
heatmap = cv2.applyColorMap(
heatmap, cv2.COLORMAP_JET)
superimposed_img = cv2.addWeighted(
img, alpha, heatmap, 1 - alpha, 0)

plt.figure(figsize=(10, 10))
plt.subplot(1, 2, 1)
plt.title(f'Original Image\nReal: {real_label}')
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
plt.axis('off')

plt.subplot(1, 2, 2)
plt.title(f'Grad-CAM\nPredicted: {predicted_label}')
plt.imshow(cv2.cvtColor(
superimposed_img, cv2.COLOR_BGR2RGB))
plt.axis('off')
plt.show()

```

```

# Menampilkan hasil visualisasi Grad-CAM
images_labels = [
('/content/split_images/val/ayam_bakar/ayam_bakar.jpg',
'ayam bakar'),
('/content/split_images/val/bakso/pic_007.jpg',
'bakso'),
('/content/split_images/val/gado_gado/pic_007.jpg',
'gado-gado'),
('/content/split_images/val/gudeg/gudeg_009.jpg',
'gudeg'),
('/content/split_images/val/nasi_goreng/nasi_goreng.jpg',
'nasi goreng'),
('/content/split_images/val/pempek/pempek_007.jpg',

```

```

'pempek'),
('/content/split_images/val/rawon/rawon_007.jpg',
'rawon'),
('/content/split_images/val/rendang/pic_007.jpg',
'rendang'),
('/content/split_images/val/sate/pic_007.jpg',
'sate'),
('/content/split_images/val/soto/soto_007.jpg',
'soto'),
]

for img_path, real_label in images_labels:
    img = image.load_img(img_path, target_size=(224, 224))
    img_array = image.img_to_array(img)
    img_array = np.expand_dims(img_array, axis=0)
    img_array = preprocess_input(img_array)

    preds = best_model.predict(img_array)
    predicted_label = class_names[np.argmax(preds)]

    heatmap = get_gradcam_heatmap(
        best_model, img_array, 'top_conv')
    display_gradcam_with_labels(
        img_path, heatmap, real_label, predicted_label)

```

h. Pengujian Model menggunakan data nutrisi makanan

```

# Membaca file xlsx
data = pd.read_excel('/content/nutrisi.xlsx')
data

```

```

# Fungsi untuk menampilkan data nutrisi makanan
def predict_class_with_nutrition(model, images, show = True):
    for img in images:

```

```

img = image.load_img(img, target_size=(224, 224))
img = image.img_to_array(img)
img = np.expand_dims(img, axis=0)
img = preprocess_input(img)

preds = model.predict(img)
class_labels = class_names
pred = np.argmax(preds, axis=-1)
predicted_class = class_names[pred[0]]
predicted_prob = preds[0][pred[0]] * 100

nutrition = data[data['nama makanan'].str.contains(
predicted_class, case=False)]
nutrition = nutrition.iloc[:, 1:]
print("-" * 30)
for col in nutrition.columns:
    print(f"{col.capitalize():>20}:\n{nutrition[col].iloc[0]:}")
print("-" * 30)

if show:
    plt.imshow(img[0].astype('uint8'))
    plt.axis('off')
    plt.title(f'{predicted_class}\n({predicted_prob:.2f}%)')
    plt.show()

```

```

# Percobaan dengan data ayam bakar
images = []
images.append(
'/content/split_images/val/ayam bakar/ayam bakar (102).jpg')

predict_class_with_nutrition(best_model, images)

```

```
# Percobaan dengan data ayam
images = []
images.append('/content/split_images/val/sate/pic_007.jpg')

predict_class_with_nutrition(best_model, images)
```