

**ANALISIS PERBANDINGAN PERFORMA ALGORITMA
CONVOLUTIONAL NEURAL NETWORK (CNN) CUSTOM DAN
MOBILENETV2 BERBASIS *TRANSFER LEARNING* DALAM
KLASIFIKASI CITRA UANG KERTAS RUPIAH**

SKRIPSI

**Diajukan untuk memenuhi sebagian persyaratan mendapatkan gelar Strata
Satu Program Studi Informatika**



**DESTY BELA PUTRI OKTAVIANI
22EO10004**

**PROGRAM STUDI INFORMATIKA
FAKULTAS MATEMATIKA DAN ILMU KOMPUTER
UNIVERSITAS NAHDLATUL ULAMA AL GHAZALI
CILACAP
2026**

**ANALISIS PERBANDINGAN PERFORMA ALGORITMA
CONVOLUTIONAL NEURAL NETWORK (CNN) CUSTOM DAN
MOBILENETV2 BERBASIS *TRANSFER LEARNING* DALAM
KLASIFIKASI CITRA UANG KERTAS RUPIAH**

SKRIPSI

**Diajukan untuk memenuhi sebagian persyaratan mendapatkan gelar Strata
Satu Program Studi Informatika**



**DESTY BELA PUTRI OKTAVIANI
22EO10004**

**PROGRAM STUDI INFORMATIKA
FAKULTAS MATEMATIKA DAN ILMU KOMPUTER
UNIVERSITAS NAHDLATUL ULAMA AL GHAZALI
CILACAP
2026**

HALAMAN PERNYATAAN KEASLIAN SKRIPSI

Dengan ini saya menyatakan bahwa penulisan skripsi dengan judul “Analisis Perbandingan Performa Algoritma *Convolutional Neural Network* (CNN) *Custom* dan *Mobilenetv2* Berbasis *Transfer Learning* dalam Klasifikasi Citra Uang Kertas Rupiah” adalah hasil karya saya dengan arahan dari pembimbing dan belum diajukan kepada pihak manapun. Sumber informasi yang dikutip dalam skripsi ini telah dicantumkan dalam Daftar Pustaka.

Demikian pernyataan ini dibuat dengan sebenarnya. Apabila di kemudian hari terdapat ketidaksesuaian dalam pernyataan ini, maka saya bersedia menerima sanksi sesuai dengan peraturan yang berlaku.

Cilacap, 5 Agustus 2026



Desty Bela Putri Oktaviani
22EO10004

LEMBAR PENGESAHAN

Skripsi Saudara

Nama : Desty Bela Putri Oktaviani
NIM : 22EO10004
Fakultas/Prodi : Fakultas MIKOM / Informatika
Judul : Analisis Perbandingan Performa Algoritma *Convolutional Neural Network* (CNN) *Custom* dan MobileNetV2 Berbasis *Transfer Learning* dalam Klasifikasi Citra Uang Kertas Rupiah

Telah disidangkan oleh Dewan Penguji Fakultas Matematika dan Ilmu Komputer Universitas Nahdlatul Ulama Al Ghazali (UNUGHA) Cilacap pada hari / tanggal :

Selasa, 28 Juli 2026

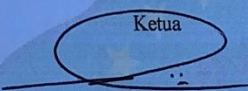
Dan dapat diterima sebagai pemenuhan tugas akhir mahasiswa Program Strata Satu (S.1) Program Studi Informatika (S.Kom) Fakultas Matematika dan Ilmu Komputer (FMIKOM) pada Universitas Nahdlatul Ulama Al Ghazali (UNUGHA) Cilacap.

Cilacap, 05 Agustus 2026

Dewan Sidang

Ketua

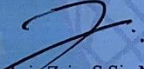
Sekretaris


Tri Anggoro, M.Kom.
NIDN. 0601128103


M.T. A. Aziz Zein, S.Si., M.Kom.
NIDN. 2125098601

Penguji 1

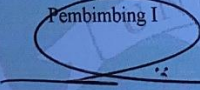
Penguji 2

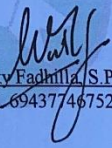

M.T. A. Aziz Zein, S.Si., M.Kom.
NIDN. 2125098601


Dr. M. Khanif, M.Pd.B.I
NIDN. 0617058703

Pembimbing I

Pembimbing II


Tri Anggoro, M.Kom.
NIDN. 0601128103


Widya Rizky Fadhillah, S.Pd., M.Sc.
NUPTK. 6943774675230282

Mengetahui

Dekan Fakultas Matematika dan Ilmu Komputer


M.T. A. Aziz Zein, S.Si., M.Kom.
NIDN. 2125098601

HALAMAN NOTA KONSULTAN

NOTA KONSULTAN

Hal : Skripsi Desty Bela Putri Oktaviani
Lampiran : -

Kepada
Yth. Dekan Fakultas Matematika dan Ilmu Komputer
Universitas Nahdlatul Ulama Al Ghazali Cilacap
di Cilacap

Assalamu'alaikum Wr. Wb.

Setelah membaca, memeriksa dan melakukan perbaikan seperlunya maka skripsi saudara/i:

Nama : Desty Bela Putri Oktaviani
NIM : 22EO10004
Prodi : Informatika
Judul : Analisis Perbandingan Performa Algoritma *Convolutional Neural Network (CNN) Custom* dan *MobileNetV2* Berbasis *Transfer Learning* dalam Klasifikasi Citra Uang Rupiah

Dapat diajukan ke Fakultas Matematika dan Ilmu Komputer, Universitas Nahdlatul Ulama Al Ghazali Cilacap untuk memenuhi syarat memperoleh gelar Strata Satu (S1).

Wassalamu'alaikum Wr. Wb.

Cilacap, 05 Agustus 2026
Konsultan



Dr. M. Khanif, M.Pd.B.I
NIDN. 0617058703

NOTA PEMBIMBING

Cilacap, 17 Juli 2026

Kepada Yth :
Fakultas Matematika dan Komputer (FMIKOM)
UNUGHA Cilacap
di Cilacap

Assalamu 'alaikum Wr. Wb.

Setelah melakukan bimbingan, telaah, arahan dan koreksi tahap penulisan skripsi
saudari:

Nama : Desty Bela Putri Oktaviani
NIM : 22EO10004
Fakultas : Matematika dan Ilmu Komputer
Prodi : Informatika
Judul : Analisis Perbandingan Performa Algoritma *Convolutional Neural Network (CNN) Custom* dan *MobileNetV2* Berbasis *Transfer Learning* dalam Klasifikasi Citra Uang Rupiah

Kami berpendapat bahwa skripsi tersebut sudah dapat diajukan ke sidang skripsi.
Bersamaan ini kami kirimkan skripsi tersebut, semoga dapat segera disidangkan.

Atas perhatiannya kami ucapkan terima kasih.
Wassalamu 'alaikum Wr. Wb.

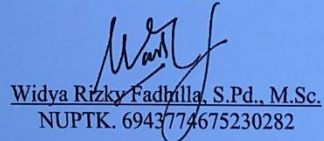
Mengetahui,

Pembimbing I



Tri Anggoro, M.Kom.
NIDN. 0601128103

Pembimbing II



Widya Rizky Fadhillah, S.Pd., M.Sc.
NUPTK. 6943774675230282

HALAMAN MOTO

“Jika bukan karena Allah yang memampukan, aku mungkin sudah lama menyerah.”

(Q.S Al-Insyirah: 5-6)

“Masih banyak makanan enak yang harus dicoba, serial yang harus ditamatkan dan tempat indah yang harus dikunjungi. Mungkin bagi kita hidup menyakitkan, tapi bagi orang-orang di sekeliling kita akan lebih menyakitkan kalau kita nggak hidup. Seberat apa pun keadaan, jangan menyerah ya.”

(Fiersa Besari)

“Anglaras ilining banyu, angeli ananging ora keli”

“Long story short I survived”

(Taylor Swift)

“Hidup adalah seni melukis tanpa penghapus, maka berbuat baiklah!”

HALAMAN PERSEMBAHAN

Segala puji bagi Allah SWT, Rabb semesta alam yang senantiasa memberikan karunia sehingga penulis mampu menyelesaikan penulisan skripsi ini. Karya ini saya persembahkan kepada :

1. Allah SWT, Sang Maha Pemilik Arah. Terima kasih telah menjadi satu-satunya rumah tempatku pulang saat dunia terasa begitu kabur dan bising. Terima kasih karena dengan karunia dan kehendak-Nya telah memberikan kemudahan dan kelancaran sehingga penulis dapat menyelesaikan skripsi ini.
2. Kepada kedua orang tuaku tercinta, meskipun kita bukan keluarga yang terbiasa berbicara dari hati ke hati, penulis tahu bahwa dibalik keheningan itu tersimpan doa yang tak pernah putus untuk penulis. Penulis memahami dan merasakan bahwa dukungan yang kalian berikan selama ini hadir dalam bentuk yang berbeda, kalian mungkin tidak selalu mengucapkannya, tetapi tindakan dan keberadaan kalian adalah bukti kasih sayang yang tak terhingga. Setiap langkah dan pencapaian yang penulis raih, termasuk selesainya skripsi ini, adalah bagian dari doa dan harapan kalian. Terima kasih atas segala pengorbanan, restu, dan kebahagiaan yang telah kalian berikan. Semoga perjuangan saya ini dapat membanggakan bapak dan ibu.
3. Kakak tercinta Dewi Kamsinah, S.Pd. yang selalu menjadi benteng, penyemangat, penasihat, sekaligus penguat. Terima kasih sudah menjadi payung dikala hujan dan menjadi tabir surya dikala terik untuk penulis. Kasih sayang dan dukunganmu sangat berarti dalam perjalanan penulis.
4. Keluarga besar peneliti, dengan tulus dan penuh rasa syukur peneliti ingin mengucapkan terima kasih kepada kalian yang senantiasa memberikan perhatian, kasih sayang, doa dan dukungan yang tiada hentinya.
5. Dosen pembimbing skripsi, bapak Tri Anggoro, M.Kom dan ibu Widya Rizky Fadhillah, S.Pd., M.Sc., yang dengan penuh kesabaran, ketulusan, dan perhatian telah membimbing, mengarahkan, serta memberikan ilmu dan motivasi kepada penulis selama proses penyusunan karya ini. Terima kasih atas setiap masukan, nasihat, dan waktu yang telah diberikan, sehingga penulis mampu menyelesaikan skripsi ini dengan baik.
6. Sahabat-sahabat terkasih penulis Faisya, Marsya, Rheika, Robbi dan sahabat-sahabat lain yang tidak bisa penulis sebutkan satu persatu. Terima kasih atas segala nasihat, arahan, serta bantuan dengan tulus tanpa pamrih, baik dalam proses akademik maupun dalam kehidupan pribadi penulis. Di setiap masa sulit, kalian hadir sebagai penguat dan penunjuk arah. Terima kasih atas kepedulian, kesabaran, dan ketulusan yang selalu kalian berikan. Semoga Allah SWT senantiasa melimpahkan kebahagiaan dan keberkahan dalam setiap langkah hidup kalian.

7. Teman-teman Informatika Kelas A Angkatan 2022, terima kasih atas solidaritas dan kebersamaan yang tidak pernah terasa sendiri. Kita tumbuh bersama dalam proses yang panjang, saling menguatkan dalam keterbatasan, dan saling menjadi saksi bahwa setiap perjuangan selalu menemukan akhirnya.
8. Keluarga besar Himafortic, yang telah menjadi tempat bagi penulis untuk belajar, bertumbuh, dan berkembang selama menjalani proses perkuliahan. Terima kasih atas setiap pengalaman, kebersamaan, pelajaran, serta dukungan yang diberikan, sehingga penulis dapat mengenal banyak hal baru dan menjadi pribadi yang lebih baik. Segala cerita dan perjalanan yang telah dilewati bersama akan selalu menjadi kenangan berharga bagi penulis.
9. Kepada Isman Fajrianto, terima kasih atas dukungan, semangat, serta telah menjadi tempat berkeluh kesah, selalu ada dalam suka maupun duka selama proses penyusunan skripsi ini. Terima kasih atas waktu, doa yang selalu senantiasa dilangitkan, dan seluruh hal baik yang diberikan selama ini.
10. Terakhir kepada saya sendiri sosok jiwa yang berkepala batu, wanita sederhana bermimpi selangit Desty Bela Putri Oktaviani yang telah bertahan hingga saat ini disaat penulis tidak percaya terhadap dirinya sendiri, namun penulis tetap mengingat bahwa setiap langkah kecil yang telah diambil adalah bagian dari perjalanan, meskipun terasa sulit atau lambat. Terima kasih karena mampu mengendalikan diri dari berbagai tekanan diluar kendali dan tidak pernah memutuskan untuk menyerah bahkan ketika semuanya terasa terlalu berat. Meski harus menghadapi kebingungan dan perasaan ingin menyerah, terima kasih karena telah jujur pada rasa takut, namun tidak membiarkan rasa takut itu membatasi langkah, karena keberanian bukanlah ketiadaan rasa takut, melainkan keinginan untuk tetap bergerak meski rasa takut masih melekat erat. Terima kasih tetap memilih berusaha dan tidak lelah mencoba. Ini merupakan pencapaian yang patut dirayakan untuk diri sendiri. Berbahagialah selalu apapun kekurangan dan kelebihanmu mari tetap berjuang untuk kedepan.

KATA PENGANTAR

Puji syukur kepada Allah SWT atas segala limpahan nikmat dan karunia-Nya, sehingga penulis dapat menyelesaikan penulisan skripsi yang berjudul: Analisis Perbandingan Performa Algoritma *Convolutional Neural Network* (CNN) *Custom* dan MobileNetV2 Berbasis *Transfer Learning* dalam Klasifikasi Citra Uang Kertas Rupiah. Sholawat dan salam senantiasa penulis haturkan kepada Nabi Muhammad SAW sebagai pembimbing seluruh umat manusia.

Skripsi ini tidak akan selesai tanpa adanya bantuan dari banyak pihak, karena itu penulis menyampaikan terima kasih kepada :

1. Bapak M.T. Abdul Aziz Zein, S.Si., M.Kom., Dekan Fakultas Matematika dan Ilmu Komputer yang telah memberikan kesempatan menimba ilmu dan dukungan selama proses studi sampai tahap penulisan skripsi ini.
2. Bapak Tri Anggoro, M.Kom. selaku pembimbing I dan ibu Widya Rizky Fadhillah, S.Pd., M.Sc. selaku pembimbing II, yang telah dengan sabar memberikan bimbingan dan petunjuk dalam menyelesaikan skripsi ini.
3. Bapak Tri Anggoro, M.Kom., Kaprodi Informatika yang telah memberikan banyak ilmu selama masa studi ini.
4. Bapak/Ibu dosen Program Studi Informatika Fakultas MIKOM UNUGHA.
5. Bapak/Ibu dosen dan seluruh keluarga besar FMIKOM.
6. Teman-teman Program Studi Informatika yang telah menjadi bagian dari perjalanan studi ini.
7. Semua pihak yang tidak dapat kami sebutkan satu persatu.

Semoga Allah SWT membalas jerih payah dan pengorbanan yang telah diberikan dengan balasan yang lebih baik. Amiin. Penulis berharap semoga karya kecil ini bermanfaat bagi pembaca.

Cilacap, 28 Juli 2026

Desty Bela Putri Oktaviani
22EO10004

ABSTRAK

Desty Bela Putri Oktaviani. ANALISIS PERBANDINGAN PERFORMA ALGORITMA *CONVOLUTIONAL NEURAL NETWORK* (CNN) *CUSTOM* DAN *MOBILENETV2* BERBASIS *TRANSFER LEARNING* DALAM KLASIFIKASI CITRA UANG KERTAS RUPIAH. Dibimbing oleh Tri Anggoro, M.Kom dan Widya Rizky Fadhillah, S.Pd., M.Sc.

Uang kertas Rupiah emisi 2022 memiliki karakteristik visual yang kompleks, seperti variasi warna, pola, tekstur, dan fitur keamanan, sehingga menjadi salah satu objek yang menantang dalam pemrosesan citra berbasis *deep learning*. Salah satu penerapannya adalah klasifikasi citra uang kertas, yang memiliki karakteristik visual kompleks seperti perbedaan warna, pola, tekstur, serta elemen keamanan. Proses identifikasi yang dilakukan secara manual berpotensi menimbulkan kesalahan, sehingga diperlukan pendekatan berbasis kecerdasan buatan yang mampu mengenali dan mengklasifikasikan citra uang kertas secara lebih akurat dan efisien.

Penelitian ini bertujuan untuk membandingkan performa dua pendekatan *deep learning*, yaitu arsitektur *Convolutional Neural Network* (CNN) *Custom* dan *MobileNetV2* berbasis *Transfer Learning*, dalam mengklasifikasikan citra uang kertas Rupiah Tahun Emisi 2022 ke dalam tujuh kelas. Proses penelitian meliputi pengumpulan dataset citra, pra-pemrosesan data, pembagian dataset menjadi data training validation dan testing, perancangan serta pelatihan model, kemudian dilanjutkan dengan evaluasi performa menggunakan metrik *accuracy*, *precision*, *recall*, dan *F1-score*. Selain itu, aspek efisiensi komputasi seperti waktu inferensi dan ukuran model juga dianalisis untuk melihat keseimbangan antara akurasi dan kompleksitas model.

Hasil penelitian menunjukkan bahwa *MobileNetV2* berbasis *Transfer Learning* memperoleh rata-rata *accuracy* sebesar 96,19%, lebih tinggi dibandingkan CNN *Custom* sebesar 94,73%. Selain itu, *MobileNetV2* memiliki ukuran model yang lebih kecil, sedangkan CNN *Custom* menunjukkan waktu inferensi yang lebih cepat. Berdasarkan hasil tersebut, *MobileNetV2* menjadi model yang lebih unggul dalam klasifikasi citra uang kertas Rupiah Tahun Emisi 2022 karena memberikan keseimbangan yang lebih baik antara akurasi dan efisiensi model. Penelitian ini diharapkan dapat menjadi referensi dalam pengembangan sistem klasifikasi citra berbasis *deep learning*, khususnya pada klasifikasi citra uang kertas Rupiah.

Kata kunci : Analisis Komparatif, Klasifikasi Citra Uang, *Convolutional Neural Network*, *MobileNetV2*, *Transfer Learning*

ABSTRACT

Desty Bela Putri Oktaviani. Analisis Perbandingan Performa Algoritma *Convolutional Neural Network (CNN) Custom* dan *MobileNetV2 Berbasis Transfer Learning* dalam Klasifikasi Citra Uang Kertas Rupiah. Supervised Tri Anggoro, M.Kom. and Widya Rizky Fadhilla, S.Pd., M.Sc.

The complex visual characteristics of the 2022 Indonesian Rupiah banknotes, including variations in color, patterns, textures, and security features, make them a challenging object for deep learning-based image processing. One such application is the classification of banknote images, which possess complex visual characteristics such as variations in color, patterns, textures, and security features. Manual identification processes are prone to errors, necessitating an artificial intelligence-based approach capable of recognizing and classifying banknote images more accurately and efficiently.

This study aims to compare the performance of two deep learning approaches a Custom Convolutional Neural Network (CNN) architecture and a Transfer Learning-based MobileNetV2 in classifying images of 2022-issue Rupiah banknotes into seven classes. The research process includes collecting image datasets, data preprocessing, splitting the data into training, validation, and testing sets, designing and training models, and evaluating performance using accuracy, precision, recall, and F1-score metrics. Additionally, computational efficiency aspects such as inference time and model size are analyzed to assess the balance between accuracy and model complexity.

The results of the study show that MobileNetV2, which is based on transfer learning, achieved an average accuracy of 96.19%, which is higher than the 94.73% achieved by the Custom CNN. In addition, MobileNetV2 has a smaller model size, while the Custom CNN demonstrates faster inference time. Based on these results, MobileNetV2 is the superior model for classifying images of 2022-issue Rupiah banknotes because it provides a better balance between accuracy and model efficiency. This study is expected to serve as a reference in the development of deep learning-based image classification systems, particularly for classifying images of Rupiah banknotes.

Keywords: *Comparative Analysis, Currency Image Classification, Convolutional Neural Network, MobileNetV2, Transfer Learning*

DAFTAR ISI

HALAMAN PERNYATAAN KEASLIAN SKRIPSI	iii
LEMBAR PENGESAHAN.....	iv
HALAMAN NOTA KONSULTAN	v
HALAMAN MOTO	vii
HALAMAN PERSEMBAHAN	viii
KATA PENGANTAR	x
ABSTRAK	xi
<i>ABSTRACT</i>	xii
DAFTAR ISI	xiii
DAFTAR GAMBAR.....	xv
DAFTAR TABEL	xvi
BAB I PENDAHULUAN.....	1
A. Latar Belakang	1
B. Rumusan Masalah	3
C. Batasan Masalah	4
D. Tujuan Penelitian	4
E. Manfaat Penelitian	4
BAB II TINJAUAN PUSTAKA.....	6
A. Penelitian Terkait	6
B. Landasan Teori.....	11
BAB III METODOLOGI.....	25
A. Waktu dan Tempat Penelitian.....	25
B. Alat dan Bahan.....	25
C. Prosedur Penelitian.....	25
D. Jadwal Penelitian.....	38
BAB IV HASIL DAN PEMBAHASAN.....	40
A. Hasil	40
B. PEMBAHASAN	52
BAB V KESIMPULAN	56
A. Kesimpulan	56

B. Saran/Rekomendasi.....	56
DAFTAR PUSTAKA	58
LAMPIRAN	63
Lampiran 1. Dataset Penelitian.....	63
Lampiran 2. <i>Source Code</i> CNN Custom	64
Lampiran 3. <i>Source Code</i> MobileNetV2	72
Lampiran 4. <i>Source Code</i> Analisis Hasil Penelitian.....	80

DAFTAR GAMBAR

Gambar 1 <i>Mind Map</i> Penelitian.....	10
Gambar 2 Algoritma <i>Convolutional Neural Network</i>	14
Gambar 3 Algoritma MobileNetV2.....	18
Gambar 4 <i>Confusion Matrix</i>	21
Gambar 5 Alur Penelitian.....	26
Gambar 7 Arsitektur CNN <i>Custom</i>	30
Gambar 8 Arsitektur MobileNetV2	33
Gambar 9 <i>Gantt Chart</i> Jadwal Penelitian	39
Gambar 10 Contoh Dataset	40
Gambar 11 Perbandingan Rata-rata <i>Training</i> dan <i>Validation Accuracy</i> CNN Custom dan MobileNetV2	42
Gambar 12 Perbandingan Rata-rata <i>Training</i> dan <i>Validation Loss</i> CNN <i>Custom</i> dan MobileNetV2	42
Gambar 13 <i>Confusion Matrix</i> CNN <i>Custom Run 1</i>	44
Gambar 14 <i>Confusion Matrix</i> MobileNetV2 <i>Run 4</i>	47
Gambar 15 Grafik Perbandingan Rata-rata Metrik Klasifikasi.....	51
Gambar 16 Grafik Perbandingan Rata-rata Efisiensi Model	51

DAFTAR TABEL

Tabel 1 Penelitian Terkait	6
Tabel 2 Alat dan Bahan beserta keterangannya	25
Tabel 3 Indikator Evaluasi Performa Model	37
Tabel 4 Pembagian Dataset	40
Tabel 5 Hasil Evaluasi CNN <i>Custom</i> pada Lima Percobaan	43
Tabel 6 <i>Classification Report</i> CNN <i>Custom</i> Run 1	45
Tabel 7 <i>Accuracy</i> per Kelas CNN <i>Custom</i> Run 1	45
Tabel 8 <i>Confusion Matrix</i> per Kelas CNN <i>Custom</i> Run 1	46
Tabel 9 Hasil Evaluasi MobileNetV2 pada Lima Percobaan	46
Tabel 10 <i>Classification Report</i> MobileNetV2 Run 4	48
Tabel 11 <i>Accuracy</i> per Kelas MobileNetV2 Run 4	48
Tabel 12 <i>Confusion Matrix</i> per Kelas MobileNetV2 Run 4	49
Tabel 13 Rekapitulasi Perbandingan CNN <i>Custom</i> dan MobileNetV2	50

BAB I PENDAHULUAN

A. Latar Belakang

Transaksi digital di Indonesia mengalami peningkatan yang signifikan dalam beberapa tahun terakhir. Di sisi lain, uang kertas Rupiah tetap menjadi salah satu alat pembayaran utama yang masih digunakan secara luas dalam aktivitas ekonomi masyarakat. Data Bank Indonesia menunjukkan bahwa jumlah uang beredar dalam arti luas (M2) mencapai Rp10.133,1 triliun pada Desember 2025 dengan pertumbuhan tahunan sebesar 9,6%[1]. Hal ini menunjukkan bahwa penggunaan uang tunai masih memegang peranan penting dalam sistem perekonomian nasional. Saat ini terdapat tujuh denominasi uang kertas Rupiah yang beredar, yaitu Rp1.000, Rp2.000, Rp5.000, Rp10.000, Rp20.000, Rp50.000, dan Rp100.000, setiap nominal memiliki karakteristik visual yang berbeda seperti warna dominan, gambar pahlawan nasional, motif batik, serta elemen pengaman khusus.

Pada tahun 2022, Bank Indonesia meluncurkan uang kertas Rupiah Tahun Emisi (TE) 2022 dengan berbagai penyempurnaan desain dan teknologi keamanan, termasuk penggunaan *microlenses* pada benang pengaman, *Optically Variable Magnetic Ink* (OVMI), serta peningkatan kualitas bahan uang [2]. Kompleksitas elemen visual tersebut menghasilkan karakteristik citra yang beragam pada setiap denominasi. Variasi warna dominan, pola, ilustrasi pahlawan nasional, serta detail keamanan pada setiap denominasi menjadikan uang kertas Rupiah sebagai objek yang relevan untuk menguji dan membandingkan kemampuan model *deep learning* dalam mengenali dan membedakan kategori citra secara otomatis. Kondisi ini menuntut adanya metode klasifikasi citra yang mampu mengekstraksi dan membedakan fitur-fitur tersebut secara akurat dan konsisten, pendekatan berbasis *Convolutional Neural Network* (CNN) telah terbukti efektif dalam melakukan ekstraksi fitur hierarkis dan meningkatkan performa klasifikasi citra pada berbagai domain visual [3], [4].

Perkembangan teknologi *Computer Vision* dan *Deep Learning* memberikan peluang besar dalam menyelesaikan permasalahan klasifikasi citra secara otomatis. Salah satu pendekatan yang dominan digunakan adalah *Convolutional Neural Network* (CNN), yang mampu mengekstraksi fitur visual secara hierarkis mulai dari fitur tingkat rendah seperti tepi dan tekstur hingga fitur tingkat tinggi seperti bentuk dan pola kompleks. Kemampuan ini menjadikan CNN efektif dalam berbagai tugas klasifikasi citra, termasuk objek dengan karakteristik visual yang kompleks. CNN telah menunjukkan performa klasifikasi yang tinggi dan konsisten pada berbagai domain citra[5].

Pengembangan model CNN untuk mengklasifikasi citra, memiliki dua pendekatan utama yang umum digunakan, yaitu membangun arsitektur CNN secara *custom* dari awal (*from scratch*) dan menggunakan teknik *transfer learning*. Pendekatan CNN yang dibangun dari awal memberikan fleksibilitas dalam perancangan arsitektur karena peneliti dapat menentukan jumlah lapisan, ukuran kernel, fungsi aktivasi, serta parameter pelatihan lainnya. Beberapa penelitian menunjukkan bahwa arsitektur CNN yang dibangun secara *custom* mampu menghasilkan performa yang baik dalam klasifikasi citra, terutama ketika arsitektur disesuaikan dengan karakteristik dataset yang digunakan. Namun di balik fleksibilitas tersebut, pelatihan model dari awal umumnya memerlukan jumlah data yang relatif besar dan waktu komputasi yang lebih lama, serta berpotensi mengalami *overfitting* apabila dataset yang tersedia terbatas [4]. Beberapa penelitian menunjukkan bahwa arsitektur *Convolutional Neural Network* (CNN) yang dibangun secara *custom* mampu menghasilkan performa yang baik dalam klasifikasi citra. Penelitian oleh Akbar *et al.* [6] menunjukkan bahwa model CNN sederhana mampu mencapai akurasi sebesar 90,6% dalam klasifikasi ekspresi wajah meskipun menggunakan dataset berukuran kecil yaitu 213 citra. Selain itu, penelitian oleh Nurtiwi *et al.* [7] yang mengkaji klasifikasi citra wajah berdasarkan kelompok usia menggunakan lebih dari 33.000 citra menunjukkan bahwa model CNN mampu mencapai akurasi sebesar 78,5% dengan pembagian data sebanyak 23.440 citra sebagai data *training* dan 10.046 citra sebagai data *testing*. Hasil tersebut menunjukkan bahwa CNN memiliki kemampuan yang baik dalam mengekstraksi fitur visual dari citra wajah serta mengelompokkan data ke dalam kategori usia secara efektif.

Sebagai alternatif dari pendekatan pelatihan dari awal (*from scratch*), teknik *transfer learning* menawarkan solusi yang lebih efisien, khususnya ketika dataset yang tersedia terbatas. *Transfer learning* bekerja dengan memanfaatkan model yang telah dilatih sebelumnya pada dataset berskala besar, sehingga model tersebut sudah memiliki kemampuan mengenali berbagai fitur visual umum seperti tepi, tekstur, dan pola bentuk objek. Kemampuan inilah yang dimaksud dengan representasi fitur yang lebih matang, yaitu model tidak perlu membangun kemampuan pengenalan fitur dari nol, melainkan langsung dapat menyesuaikan diri dengan dataset baru secara lebih efisien. Salah satu arsitektur yang banyak digunakan dalam pendekatan ini adalah MobileNetV2, yang dikenal sebagai CNN ringan dengan jumlah parameter yang efisien namun tetap mampu menghasilkan performa klasifikasi yang kompetitif.

Penelitian oleh Syakuroh *et al.* [8] pada klasifikasi citra X-ray paru-paru menunjukkan bahwa MobileNetV2 mampu mencapai akurasi sebesar 99,42% dengan *precision* 98,87%, *recall* 98,88%, dan *F1-score* 98,86%. Selanjutnya, Hartono dan Muslikh [9] dalam studi klasifikasi 20 jenis buah menggunakan 4.000 citra melaporkan bahwa MobileNetV2 mencapai akurasi sebesar 94% dengan rata-

rata *precision*, *recall*, dan *F1-score* pada rentang 0,93–0,94. Pada penelitian identifikasi tumor otak berbasis citra MRI dengan 20.000 data, Somoal dan Dzikirollah [10] menunjukkan bahwa pendekatan *transfer learning* MobileNetV2 menghasilkan akurasi 98%, *precision* 98%, *recall* 97,75%, dan *F1-score* 97,75%.

Meskipun berbagai penelitian telah menerapkan CNN maupun *transfer learning* dalam berbagai tugas klasifikasi citra, kajian yang secara khusus membandingkan performa antara arsitektur CNN *custom* dan pendekatan *transfer learning* berbasis MobileNetV2 pada klasifikasi citra uang kertas Rupiah Tahun Emisi 2022 masih terbatas. Padahal, penentuan arsitektur model yang tepat merupakan langkah penting dalam pengembangan model klasifikasi citra, terutama jika mempertimbangkan keterbatasan sumber daya komputasi. Tanpa adanya analisis komparatif yang mendalam mengenai keseimbangan antara akurasi, ukuran model, dan kecepatan komputasi (*inference time*), sulit untuk menentukan pendekatan yang paling efisien dalam menangani karakteristik visual uang Rupiah yang kompleks.

Berdasarkan perbedaan tersebut, penelitian ini bertujuan untuk membandingkan performa antara arsitektur *Convolutional Neural Network custom* dan MobileNetV2 berbasis *Transfer Learning* dalam mengklasifikasi citra uang kertas Rupiah Tahun Emisi 2022. Evaluasi dilakukan dengan mengukur akurasi, *precision*, *recall*, *F1-score*, ukuran model, serta kecepatan inferensi guna memperoleh rekomendasi pendekatan yang lebih optimal dan efisien. Secara ilmiah, penelitian ini diharapkan dapat memberikan kontribusi berupa evaluasi komprehensif terhadap efektivitas dua pendekatan arsitektur CNN pada domain klasifikasi uang kertas Rupiah serta menjadi referensi bagi penelitian lanjutan dalam bidang klasifikasi citra berbasis *deep learning*. Hingga saat ini, kajian yang secara khusus membandingkan CNN *Custom* dan MobileNetV2 pada citra uang kertas Rupiah TE 2022 dengan turut mempertimbangkan efisiensi komputasi masih sangat terbatas. Oleh karena itu, penelitian ini penting dilakukan untuk mengisi kekosongan tersebut sekaligus menghasilkan rekomendasi yang dapat dijadikan pijakan bagi pengembang dalam memilih pendekatan yang paling tepat saat merancang sistem klasifikasi citra uang kertas ke depannya.

B. Rumusan Masalah

Rumusan masalah pada penelitian ini sebagai berikut :

1. Bagaimana membangun model klasifikasi citra uang kertas Rupiah Tahun Emisi 2022 menggunakan arsitektur Convolutional Neural Network (CNN) Custom dan MobileNetV2 berbasis *Transfer Learning*?
2. Bagaimana hasil analisis perbandingan performa model CNN Custom dan MobileNetV2 berbasis *Transfer Learning* dalam klasifikasi citra uang

kertas Rupiah Tahun Emisi 2022 berdasarkan metrik *accuracy*, *precision*, *recall*, *F1-score*, ukuran model, dan waktu inferensi?

C. Batasan Masalah

Batasan masalah membahas mengenai :

1. Penelitian difokuskan pada klasifikasi citra uang kertas Rupiah Tahun Emisi (TE) 2022 untuk tujuh denominasi yang beredar.
2. Metode yang dibandingkan terbatas pada dua pendekatan, yaitu algoritma CNN *custom (from scratch)* dan MobileNetV2 dengan pendekatan *transfer learning*.
3. Evaluasi performa model dibatasi pada metrik akurasi, *precision*, *recall*, *F1-score*, ukuran model, dan kecepatan inferensi.
4. Penelitian ini berfokus pada analisis komparatif performa model dan tidak membahas pengembangan sistem atau aplikasi implementatif.
5. Penelitian dilakukan menggunakan perangkat komputer dengan spesifikasi minimal prosesor Intel Core i3 atau setara, RAM 8 GB, penyimpanan SSD, dan sistem operasi 64-bit.

D. Tujuan Penelitian

Tujuan penelitian yang diinginkan peneliti dalam penelitian yaitu sebagai berikut:

1. Menganalisis dan membandingkan performa arsitektur CNN *custom* dan MobileNetV2 berbasis *Transfer Learning* dalam klasifikasi citra uang kertas Rupiah Tahun Emisi 2022 berdasarkan metrik akurasi, *precision*, *recall*, dan *F1-score*.
2. Mengevaluasi perbedaan kedua pendekatan tersebut yang meliputi ukuran model, dan kecepatan inferensi untuk menentukan model yang lebih optimal.

E. Manfaat Penelitian

1. Manfaat Teoritis

Penelitian ini memberikan kontribusi dalam pengembangan kajian *Deep Learning* dan *Computer Vision*, khususnya terkait analisis komparatif performa antara arsitektur CNN *custom* dan MobileNetV2 berbasis *Transfer Learning* pada tugas klasifikasi citra. Hasil penelitian ini memperkaya referensi ilmiah mengenai perbandingan tingkat akurasi dan efisiensi komputasi kedua pendekatan tersebut, serta dapat menjadi rujukan bagi penelitian lanjutan yang berfokus pada optimasi model jaringan saraf tiruan untuk klasifikasi citra.

2. Manfaat Praktis

a. Bagi Institusi Pendidikan

Menjadi tambahan literatur akademik dalam bidang Kecerdasan Buatan (*Artificial Intelligence*) dan Pengolahan Citra Digital, serta referensi bagi mahasiswa atau peneliti dalam pengembangan penelitian terkait klasifikasi citra berbasis *deep learning*.

b. Bagi Peneliti dan Pengembang

Memberikan gambaran komparatif mengenai performa arsitektur CNN *custom* dan MobileNetV2, sehingga dapat menjadi bahan pertimbangan dalam pemilihan model yang sesuai untuk kebutuhan klasifikasi citra dengan mempertimbangkan aspek akurasi dan efisiensi komputasi.

BAB II TINJAUAN PUSTAKA

A. Penelitian Terkait

Penelitian terkait berisi dasar ilmiah perkembangan topik penelitian dalam skripsi (*state of the art*) dan memperlihatkan posisi penelitian yang akan dilakukan dalam perkembangan topik. Dasar ilmiah diambil dari penelitian-penelitian sebelumnya menggunakan metode yang sama dalam jurnal/paper, *proceeding*, maupun artikel ilmiah yang terbit pada periode 5 (lima) tahun terakhir. Penelitian terkait disajikan dalam bentuk tabel dan/atau *mind map*.

Tabel 1 Penelitian Terkait

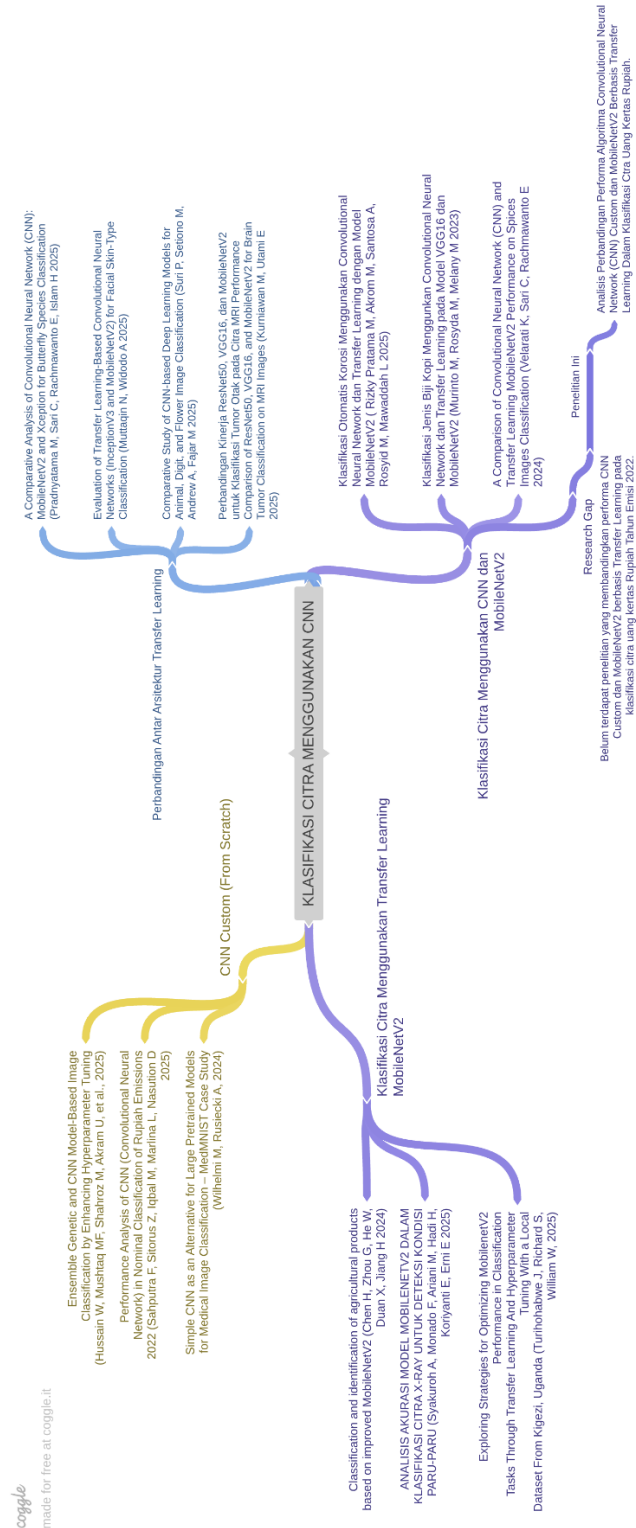
No.	Nama (Tahun)	Judul
(1)	(2)	(3)
1	Velarati K, Sari C, Rachmawanto E (2024)	Judul: <i>A Comparison of Convolutional Neural Network (CNN) and Transfer Learning MobileNetV2 Performance on Spices Images Classification</i> Metode: CNN Custom dan <i>Transfer Learning</i> MobileNetV2 Hasil: Pada eksperimen pertama, CNN tanpa <i>transfer learning</i> mencapai akurasi 93%. Eksperimen kedua menggunakan MobileNetV2, akurasi meningkat mencapai nilai 99% untuk semua kelas rempah-rempah.
2	Pradnyatama et al. (2025)	Judul: <i>A Comparative Analysis of Convolutional Neural Network (CNN): MobileNetV2 and Xception for Butterfly Species Classification</i> Metode: <i>Transfer Learning</i> MobileNetV2 dan Xception Hasil: Hasil eksperimen menunjukkan Xception mencapai akurasi uji 93,40% dibandingkan dengan 93,20% pada MobileNetV2. Tingkat akurasi tinggi ini dicapai melalui <i>transfer learning</i> yang efektif yang mempertahankan kemampuan ekstraksi fitur umum, teknik penyeimbangan kelas yang komprehensif, dan strategi laju pembelajaran yang disesuaikan dengan cermat untuk setiap arsitektur. Meskipun perbedaan kinerja minimal, MobileNetV2 menawarkan keunggulan efisiensi komputasi yang signifikan dengan 4,15 juta parameter dibandingkan dengan 25,27 juta parameter Xception, sementara

No.	Nama (Tahun)	Judul
(1)	(2)	(3)
		Xception memberikan klasifikasi yang sedikit lebih baik.
3	Muttaqin dan Widodo et al. (2025)	Judul: <i>Evaluation of Transfer Learning-Based Convolutional Neural Networks (InceptionV3 and MobileNetV2) for Facial Skin-Type Classification</i> Metode: Transfer Learning (InceptionV3 dan MobileNetV2) Hasil: Metode InceptionV3 mencapai akurasi tertinggi yaitu 90,12% dan skor F1 makro 89,47%, terutama unggul dalam mengidentifikasi kulit normal dan rentan jerawat. MobileNetV2 mencapai akurasi 81,15% dan berkinerja baik pada jenis kulit kering.
4	Suri et al. (2025)	Judul: <i>Comparative Study of CNN-based Deep Learning Models for Animal, Digit, and Flower Image Classification</i> Metode: Model jaringan saraf konvolusional (CNN) – MobileNetV2, DenseNet121, EfficientNetB0, dan InceptionV3 Hasil: Hasil menunjukkan bahwa DenseNet121 secara konsisten mencapai akurasi sebesar 98% pada gambar hewan dan 88% pada gambar bunga, sementara MobileNetV2 memberikan kinerja yang hampir sama (97% dan 82%) tetapi dengan waktu pemrosesan yang jauh lebih cepat, antara 11 dan 55 menit. EfficientNetB0, di sisi lain, berkinerja buruk pada dataset bunga yang lebih kompleks, hanya mencapai akurasi 5%. Temuan ini menunjukkan bahwa DenseNet121 ideal untuk proyek di mana akurasi menjadi prioritas utama, sedangkan MobileNetV2 lebih cocok untuk aplikasi <i>real-time</i> yang memerlukan respons cepat tanpa penurunan akurasi yang signifikan.
5	Chen et al. (2024)	Judul: <i>Classification and identification of agricultural products based on improved MobileNetV2</i> Metode: Improved MobileNetV2 (Res-Inception dan Efficient Multi-scale Attention) Hasil: Model Improved-MobileNetV2 menunjukkan peningkatan performa dibandingkan MobileNetV2

No.	Nama (Tahun)	Judul
(1)	(2)	(3)
		standar. MobileNetV2 memperoleh akurasi sebesar 97,26%, sedangkan Improved-MobileNetV2 mencapai akurasi hingga 99,96% pada dataset klasifikasi produk pertanian. Peningkatan tersebut diperoleh melalui modifikasi arsitektur dengan penambahan modul yang dirancang untuk meningkatkan kemampuan ekstraksi fitur multi-skala. Selain itu, model Improved-MobileNetV2 memiliki jumlah parameter sekitar 3.028.211, jumlah yang relatif lebih kecil dibandingkan beberapa arsitektur CNN lain.
6	Rizky Pratama et al. (2025)	Judul: Klasifikasi Otomatis Korosi Menggunakan <i>Convolutional Neural Network</i> dan <i>Transfer Learning</i> dengan Model MobileNetV2 Metode: CNN vs MobileNetV2 (Transfer Learning) Hasil: MobileNetV2 mencapai akurasi uji 95% dan AUC 0,9893, lebih tinggi dibanding CNN sederhana sebesar 82%. Hal ini menunjukkan efektivitas <i>transfer learning</i>.
7	Sahputra et al. (2025)	Judul: <i>Performance Analysis of CNN (Convolutional Neural Network) in Nominal Classification of Rupiah Emissions 2022</i> Metode: CNN+Adam, CNN+RMSprop, dan VGG16 (transfer learning) Hasil: Hasil menunjukkan bahwa CNN dengan optimizer Adam mencapai akurasi validasi 98,97%, sementara CNN dengan RMSprop mencapai 99,59%. Sementara itu, model VGG16 mencapai akurasi validasi sempurna sebesar 100%, dengan nilai presisi, <i>recall</i>, dan <i>F1-score</i> masing-masing 1.00. Hasil ini menunjukkan bahwa pendekatan <i>transfer learning</i> memberikan kinerja terbaik dibandingkan dengan model CNN konvensional.
8	Syakuroh et al. (2025)	Judul: Analisis Akurasi Model MobileNetV2 dalam Klasifikasi CITRA X-RAY untuk Deteksi Kondisi Paru-Paru Metode: <i>Transfer Learning</i> MobileNetV2 Hasil: Hasil pengujian pada data uji menunjukkan bahwa model yang diusulkan mampu mencapai nilai akurasi sebesar 99,42%, <i>precision</i> 98,87%, <i>recall</i>

No.	Nama (Tahun)	Judul
(1)	(2)	(3)
		98,88%, dan <i>F1-score</i> 98,86%. Seluruh hasil evaluasi ini melampaui standar medis minimal $\geq 90\%$ aplikasi klinis. Temuan penelitian ini menegaskan bahwa MobileNetV2 sangat potensial digunakan sebagai alat bantu diagnosis otomatis berbasis citra X-ray, guna meningkatkan efektivitas deteksi dini penyakit paru-paru pada lingkungan klinis.
9	Murinto M, Rosyda M, Melany M (2023)	Judul: Klasifikasi Jenis Biji Kopi Menggunakan <i>Convolutional Neural Network</i> dan <i>Transfer Learning</i> pada Model VGG16 dan MobileNetV2 Metode: CNN custom, <i>Transfer Learning</i> VGG16, dan <i>Transfer Learning</i> MobileNetV2 Hasil: Model CNN memperoleh akurasi 95%, VGG16 memperoleh akurasi 95%, dan MobileNetV2 memperoleh akurasi tertinggi sebesar 96%. Hasil menunjukkan bahwa pendekatan <i>transfer learning</i> memberikan peningkatan performa dibanding CNN biasa, meskipun peningkatan akurasi relatif kecil.
10	Kurniawan M, Utami E (2025)	Judul: Perbandingan Kinerja ResNet50, VGG16, dan MobileNetV2 untuk Klasifikasi Tumor Otak pada Citra MRI Metode: ResNet50, VGG16, dan MobileNetV2 Hasil: Hasil eksperimen menunjukkan bahwa VGG16 memberikan performa terbaik di antara ketiga model dengan akurasi 94.93%, presisi 94.68%, dan <i>specificity</i> 98.33%, serta waktu pelatihan terpendek, yaitu 47.15 menit. Sementara itu, MobileNetV2 menunjukkan kinerja yang solid dengan recall 94.08%, meskipun membutuhkan waktu pelatihan lebih lama (79.53 menit). ResNet50 mencatatkan akurasi terendah (91.67%), meskipun unggul dalam presisi (91.79%), namun memiliki kekurangan pada recall (91.25%) dan <i>specificity</i> (97.2%).

Gambar 1 merupakan *mind map* penelitian terkait sebagai referensi penulisan dalam bentuk gambar, yaitu sebagai berikut :



Gambar 1 *Mind Map* Penelitian

B. Landasan Teori

Landasan teori pada penelitian “Analisis Perbandingan Performa Algoritma *Convolutional Neural Network (CNN) Custom* dan *Transfer Learning* Berbasis MobileNetV2 dalam Klasifikasi Citra Uang Kertas Rupiah” yaitu:

1. Citra Digital

Citra digital merupakan representasi visual suatu objek atau fenomena yang dinyatakan dalam bentuk data numerik sehingga dapat diproses oleh komputer. Dalam pengolahan citra digital, sebuah gambar tidak hanya dipandang sebagai bentuk visual, melainkan sebagai sekumpulan nilai numerik yang tersusun secara sistematis dalam suatu struktur data. Citra digital tersusun atas elemen-elemen kecil yang disebut piksel (*picture element*), dengan setiap piksel memiliki nilai tertentu yang merepresentasikan tingkat kecerahan atau warna pada posisi tertentu dalam citra [17].

Secara matematis, citra digital dua dimensi direpresentasikan dalam bentuk matriks berukuran $M \times N$, dengan M menyatakan jumlah baris dan N menyatakan jumlah kolom. Setiap elemen matriks tersebut menunjukkan satu piksel dengan nilai intensitas tertentu. Nilai piksel pada posisi tertentu dapat dinyatakan sebagai $f(i, j)$, yaitu nilai intensitas pada koordinat spasial tertentu [18]. Representasi dalam bentuk matriks ini memungkinkan citra diproses menggunakan operasi matematika dan komputasi.

Berdasarkan jenis informasinya, citra digital dapat dibedakan menjadi citra *grayscale* dan citra berwarna. Pada citra *grayscale*, setiap piksel hanya memiliki satu nilai intensitas dalam rentang tertentu, umumnya berada pada rentang 0 hingga 255 pada sistem 8-bit, dengan 0 menunjukkan warna hitam dan 255 menunjukkan warna putih. Sementara itu, citra berwarna umumnya menggunakan model warna RGB (*Red, Green, Blue*), setiap piksel direpresentasikan oleh tiga nilai intensitas yang masing-masing menyatakan komponen merah, hijau, dan biru. Kombinasi ketiga kanal tersebut menghasilkan variasi warna yang luas pada citra digital [18].

Selain representasi piksel dan model warna, kualitas citra digital juga dipengaruhi oleh resolusi. Resolusi citra merujuk pada jumlah piksel yang menyusun suatu citra dan biasanya dinyatakan dalam bentuk $M \times N$. Citra dengan resolusi tinggi memiliki jumlah piksel yang lebih banyak sehingga mampu merepresentasikan detail visual secara lebih jelas, sedangkan resolusi rendah menghasilkan citra yang tampak lebih kasar [17].

Proses pembentukan citra digital dari citra analog melibatkan dua tahapan penting, yaitu *sampling* dan *quantization*. *Sampling* merupakan proses pengambilan titik-titik diskrit dari citra kontinu untuk membentuk *grid* piksel yang menentukan resolusi citra. *Quantization* adalah proses pembatasan nilai intensitas kontinu

menjadi sejumlah level diskrit tertentu, misalnya 256 tingkat pada citra 8-bit [18]. Kedua proses ini sangat menentukan kualitas citra digital yang dihasilkan. Dengan demikian, citra digital menjadi dasar dalam berbagai proses pengolahan dan analisis visual, karena seluruh teknik komputasi dalam bidang ini bekerja berdasarkan representasi piksel dalam bentuk numerik tersebut.

2. *Computer Vision*

Computer vision merupakan bidang ilmu yang mempelajari mengenai proses sistem komputer dapat memperoleh, memproses, dan memahami informasi dari data visual seperti citra maupun video. Tujuan utama bidang ini adalah mengembangkan algoritma yang memungkinkan mesin menafsirkan dunia visual secara otomatis, sehingga komputer tidak hanya membaca nilai piksel, tetapi juga mampu memahami struktur, pola, dan makna yang terkandung di dalam suatu gambar [19].

Secara umum, *computer vision* mencakup berbagai aspek, mulai dari proses pembentukan citra (*image formation*), pengolahan citra (*image processing*), hingga pengenalan dan interpretasi objek (*recognition*) [19]. Berbeda dengan pengolahan citra yang berfokus pada manipulasi atau peningkatan kualitas visual, *computer vision* menitikberatkan pada kemampuan sistem dalam melakukan analisis serta menghasilkan pemahaman terhadap isi visual tersebut. Permasalahan dalam *computer vision* dapat berupa pengenalan objek, deteksi objek, segmentasi citra, serta rekonstruksi tiga dimensi seperti *structure from motion* dan estimasi kedalaman (*depth estimation*) [19]. Berbagai permasalahan tersebut bertujuan untuk mengubah data visual menjadi informasi yang dapat digunakan dalam proses pengambilan keputusan oleh sistem komputasi.

Seiring perkembangan teknologi, pendekatan dalam *computer vision* semakin mengarah pada metode berbasis data (*data-driven approach*), dengan sistem belajar dari sejumlah besar data visual untuk meningkatkan akurasi dan kemampuan generalisasi model [19]. Perkembangan ini kemudian menjadi fondasi bagi munculnya berbagai pendekatan modern dalam analisis visual. Salah satu pendekatan yang berkembang pesat adalah deep learning, yang mampu mempelajari representasi fitur secara otomatis langsung dari data citra tanpa memerlukan proses ekstraksi fitur secara manual. Kemampuan tersebut menjadikan deep learning banyak diterapkan pada berbagai tugas computer vision, termasuk klasifikasi citra dengan karakteristik visual yang kompleks.

3. *Deep Learning*

Deep Learning (DL) merupakan bagian dari *machine learning* yang berfokus pada penggunaan jaringan saraf tiruan berlapis banyak (*multi-layer neural networks*) untuk mempelajari representasi data secara hierarkis. Dalam perkembangannya, *deep learning* muncul sebagai pendekatan yang mampu

mengatasi keterbatasan metode *machine learning* konvensional dalam menangani data mentah berskala besar dan kompleks [20].

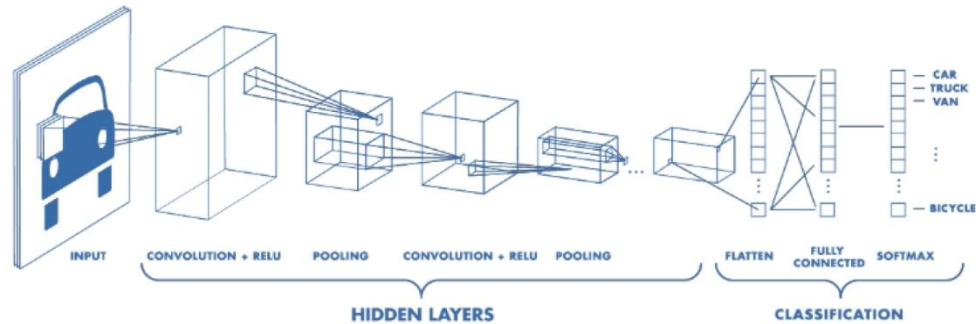
Berbeda dengan pendekatan *machine learning* tradisional yang umumnya memerlukan proses rekayasa fitur (*feature engineering*) secara manual, *deep learning* mengandalkan konsep *representation learning*. Konsep ini memungkinkan sistem secara otomatis mempelajari representasi fitur dari data mentah melalui serangkaian transformasi non-linear pada beberapa lapisan jaringan [20]. Dengan demikian, model tidak lagi sepenuhnya bergantung pada perancangan fitur oleh manusia, melainkan mampu membangun representasi internal yang lebih abstrak dan kompleks.

Istilah “*deep*” dalam *deep learning* merujuk pada kedalaman arsitektur jaringan, yaitu banyaknya lapisan pemrosesan (*processing layers*) yang digunakan dalam model. Setiap lapisan bertugas mengekstraksi pola yang semakin kompleks dari lapisan sebelumnya. Lapisan awal umumnya menangkap fitur sederhana, sedangkan lapisan yang lebih dalam mempelajari representasi tingkat tinggi yang lebih abstrak[20]. Struktur bertingkat ini memungkinkan model memahami hubungan *non-linear* dalam data secara lebih efektif.

Deep learning telah banyak digunakan dalam menangani data tidak terstruktur seperti citra, teks, dan suara. Berbagai arsitektur seperti *Convolutional Neural Network* (CNN), *Recurrent Neural Network* (RNN), dan *Generative Adversarial Network* (GAN) dikembangkan untuk menyelesaikan permasalahan pada domain yang berbeda [20]. Perkembangannya didukung oleh ketersediaan data berskala besar serta peningkatan kemampuan komputasi modern.

Meskipun demikian, *deep learning* juga memiliki sejumlah tantangan, antara lain kebutuhan data pelatihan dalam jumlah besar, waktu pelatihan yang relatif lama, serta kompleksitas model yang tinggi. Selain itu, model *deep learning* berpotensi mengalami *overfitting* apabila tidak dilakukan pengaturan parameter dan strategi regularisasi yang tepat [20]. Dengan karakteristik tersebut, *deep learning* menjadi landasan penting dalam pengembangan berbagai metode analisis data modern.

4. Convolutional Neural Network (CNN)



Gambar 2 Algoritma *Convolutional Neural Network*

Convolutional Neural Network (CNN) merupakan salah satu arsitektur *deep learning* yang dirancang khusus untuk mengolah data berbentuk *grid*, seperti citra dua dimensi. CNN termasuk dalam kategori *Artificial Neural Network* (ANN) dan banyak digunakan dalam bidang *computer vision* karena kemampuannya dalam mengekstraksi fitur visual secara otomatis tanpa memerlukan perancangan fitur manual [21], [22]. Proses ekstraksi fitur tersebut dilakukan secara bertingkat, mulai dari fitur sederhana seperti tepi dan tekstur hingga fitur yang lebih kompleks, sehingga model mampu mengenali pola visual secara lebih efektif. Kemampuan tersebut menjadikan CNN sebagai salah satu metode yang paling banyak digunakan dalam tugas klasifikasi citra, termasuk untuk mengenali objek yang memiliki karakteristik visual kompleks seperti uang kertas.

Berbeda dengan jaringan saraf konvensional yang menggunakan koneksi penuh (*fully connected network*), CNN memanfaatkan prinsip *local connectivity* dan *weight sharing*. Pada mekanisme ini, setiap filter (kernel) hanya bekerja pada area lokal citra dan menggunakan bobot yang sama saat digeser ke seluruh bagian citra. Pendekatan tersebut membuat CNN lebih efisien dalam jumlah parameter serta lebih efektif dalam menangkap pola spasial seperti tepi, tekstur, dan bentuk objek [21], [22].

Secara umum, arsitektur CNN terdiri atas beberapa komponen utama, yaitu *convolution layer*, *pooling layer*, *activation function*, dan *fully connected layer* [22], [23]. Kombinasi lapisan tersebut memungkinkan proses ekstraksi fitur dan klasifikasi dilakukan secara terintegrasi dalam satu model.

a. *Convolution Layer*

Convolution layer merupakan komponen utama dalam CNN. Pada lapisan ini, citra masukan dikonvolusikan dengan sejumlah kernel berukuran kecil, seperti 3×3 atau 5×5 , untuk menghasilkan *feature map* yang merepresentasikan pola tertentu dalam citra [22], [23]. Proses konvolusi dilakukan dengan menggeser kernel ke seluruh bagian citra, kemudian menghitung hasil perkalian

elemen yang bersesuaian dan menjumlahkannya. Parameter penting dalam *convolution layer* meliputi:

- 1) *Kernel size*, yaitu ukuran filter;
- 2) *Stride*, yaitu langkah pergeseran filter;
- 3) *Padding*, yaitu penambahan nilai pada tepi citra untuk mempertahankan dimensi keluaran.

Pengaturan parameter tersebut memengaruhi ukuran *feature map* dan kemampuan model dalam menangkap informasi visual [21], [22].

b. *Pooling Layer*

Pooling layer berfungsi untuk melakukan reduksi dimensi terhadap *feature map* yang dihasilkan oleh *convolution layer*. Tujuannya adalah mengurangi jumlah parameter, meningkatkan efisiensi komputasi, serta membuat model lebih tahan terhadap pergeseran kecil pada citra [22], [23]. Metode *pooling* yang umum digunakan antara lain:

- 1) *Max pooling*, yang mengambil nilai maksimum dalam suatu area;
- 2) *Average pooling*, yang mengambil nilai rata-rata dalam suatu area;

Penggunaan *pooling* membantu mempertahankan informasi penting sekaligus mengurangi kompleksitas data [22].

c. *Activation Function*

Activation Function digunakan untuk memperkenalkan sifat non-linear dalam jaringan. Tanpa fungsi aktivasi, jaringan hanya menghasilkan transformasi linear dan tidak mampu memodelkan hubungan kompleks dalam data. Beberapa fungsi aktivasi yang umum digunakan dalam CNN antara lain:

- 1) *Sigmoid*;
- 2) *Tanh*,
- 3) *ReLU (Rectified Linear Unit)*,
- 4) *Leaky ReLU*,

ReLU menjadi fungsi aktivasi yang paling banyak digunakan karena lebih sederhana secara komputasi dan membantu mengurangi permasalahan *vanishing gradient* [22], [23].

d. *Fully Connected Layer*

Fully Connected layer biasanya ditempatkan pada bagian akhir arsitektur CNN. Lapisan ini menerima hasil ekstraksi fitur yang telah diratakan (*flattened*)

dan melakukan proses klasifikasi berdasarkan fitur tersebut [23]. Pada permasalahan klasifikasi multi-kelas, lapisan akhir umumnya menggunakan fungsi aktivasi seperti *softmax* untuk menghasilkan probabilitas setiap kelas [22].

5. Arsitektur *Convolutional Neural Network Custom*

Arsitektur CNN *custom* merupakan model jaringan saraf konvolusional yang dirancang secara spesifik sesuai dengan kebutuhan permasalahan penelitian. Berbeda dengan model *pre-trained* seperti ResNet atau VGG yang arsitekturnya telah ditentukan sebelumnya dan dilatih pada dataset berskala besar misalnya ImageNet, CNN *custom* dibangun dari awal dengan menentukan sendiri jumlah layer, jumlah filter, ukuran kernel, serta strategi pelatihan yang sesuai dengan kompleksitas dataset yang digunakan.

Secara konseptual, struktur dasar CNN *Custom* tetap mengikuti struktur dasar CNN yang terdiri atas beberapa lapisan, yaitu *convolution layer*, *activation function*, *pooling layer*, dan *fully connected layer*. Namun, kedalaman jaringan dan jumlah parameter dirancang secara adaptif sesuai kebutuhan. Menurut Taye [21] arsitektur *deep learning* dapat dikembangkan secara fleksibel untuk menyesuaikan kompleksitas model dengan karakteristik data yang digunakan. Prinsip ini menjadi dasar dalam perancangan CNN *Custom*, dengan struktur jaringan tidak harus mengikuti arsitektur besar yang telah ditentukan sebelumnya.

Penggunaan CNN *custom* umumnya dipertimbangkan ketika dataset bersifat spesifik atau jumlah data relatif terbatas, sehingga diperlukan model dengan kompleksitas yang lebih terkendali. Selain itu, model *custom* sering kali lebih efisien dari sisi jumlah parameter dan kebutuhan komputasi. Penelitian yang dilakukan oleh Alam *et al.* [24] menunjukkan bahwa CNN *Custom* yang dirancang dengan konfigurasi lapisan yang tepat mampu menghasilkan performa klasifikasi yang kompetitif dibandingkan model *pre-trained*, dengan waktu pelatihan dan ukuran model yang lebih efisien. Dengan demikian, arsitektur CNN *custom* merupakan pendekatan yang relevan dalam pengembangan sistem klasifikasi citra, terutama ketika mempertimbangkan keseimbangan antara akurasi, kompleksitas model, dan efisiensi komputasi.

6. *Transfer Learning*

Transfer learning merupakan salah satu teknik dalam *machine learning* yang memungkinkan pemanfaatan pengetahuan yang telah dilatih sebelumnya untuk menyelesaikan tugas baru yang berkaitan. Berbeda dengan pendekatan *machine learning* tradisional yang melatih model dari awal untuk setiap tugas secara terpisah, *transfer learning* memanfaatkan model atau pengetahuan dari domain sumber untuk meningkatkan performa pembelajaran pada domain target [25]. Konsep utama *transfer learning* adalah memanfaatkan hubungan antara tugas

sebelumnya dengan tugas yang sedang dipelajari. Pendekatan ini membantu model menghasilkan solusi yang lebih cepat dan akurat, terutama ketika jumlah data pada domain target terbatas atau sulit diperoleh [25].

Menurut Gupta *et al.* [26] *transfer learning* merupakan metode yang menggunakan model yang telah dilatih sebelumnya sebagai dasar untuk membangun model baru pada dataset yang berbeda. Model tersebut tidak perlu dilatih dari awal karena sudah memiliki fitur yang diperoleh dari proses pelatihan sebelumnya, sehingga dapat digunakan kembali untuk menyelesaikan permasalahan lain yang memiliki karakteristik serupa. Proses *transfer learning* melibatkan pemindahan pengetahuan dari *source* domain ke target domain. *Source* domain merupakan domain tempat model awal dilatih, sedangkan target domain merupakan domain baru yang menjadi tujuan pembelajaran. Dengan memanfaatkan pengetahuan yang telah dipelajari, model dapat meningkatkan performa pada tugas baru secara lebih efisien [25].

Dalam implementasinya, *transfer learning* sering menggunakan *pretrained model*, yaitu model yang telah dilatih menggunakan dataset berskala besar seperti ImageNet. Model tersebut kemudian disesuaikan dengan dataset baru melalui proses *fine-tuning*. Pendekatan ini memungkinkan model menggunakan fitur yang telah dipelajari sebelumnya, sehingga dapat meningkatkan akurasi dan mengurangi waktu pelatihan [26]. Penggunaan *transfer learning* telah diterapkan dalam berbagai bidang, seperti klasifikasi citra, pengolahan bahasa alami, sistem rekomendasi, dan klasifikasi dokumen. Teknik ini membantu meningkatkan performa model dengan memanfaatkan informasi yang telah tersedia dari domain lain [26].

Transfer learning memiliki beberapa jenis, yaitu *inductive transfer learning*, *transductive transfer learning*, dan *unsupervised transfer learning*. *Inductive transfer learning* digunakan ketika tugas sumber dan target berbeda tetapi masih berkaitan. *Transductive transfer learning* digunakan ketika tugasnya sama namun domainnya berbeda, sedangkan *unsupervised transfer learning* digunakan ketika tidak tersedia data berlabel pada kedua domain [25]. Berdasarkan konsep tersebut, *transfer learning* menjadi salah satu metode yang efektif untuk meningkatkan kinerja model *machine learning*, khususnya ketika data pelatihan terbatas dan waktu pelatihan menjadi pertimbangan penting.

7. MobileNetV2

MobileNetV2 merupakan arsitektur *Convolutional Neural Network* (CNN) yang dirancang untuk menghasilkan model yang ringan dan efisien, khususnya pada perangkat dengan keterbatasan sumber daya komputasi. Arsitektur ini diperkenalkan oleh Sandler *et al.* [27] pada *IEEE Conference on Computer Vision*

and Pattern Recognition (CVPR) tahun 2018 melalui publikasi berjudul *MobileNetV2: Inverted Residuals and Linear Bottlenecks*.

Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	-

Gambar 3 Algoritma MobileNetV2

Pengembangan MobileNetV2 bertujuan untuk meningkatkan efisiensi jumlah parameter dan kompleksitas komputasi, yang diukur melalui operasi *multiply-adds* (MAdds), tanpa menurunkan akurasi secara signifikan [27]. Dengan pendekatan ini, arsitektur mampu mencapai keseimbangan antara performa dan efisiensi komputasi. Inovasi utama dalam MobileNetV2 terletak pada kombinasi tiga komponen arsitektural, yaitu *depthwise separable convolution*, *linear bottleneck*, dan *inverted residual structure*.

a. *Depthwise Separable Convolution*

MobileNetV2 menggunakan *depthwise separable convolution* sebagai mekanisme komputasi utama untuk meningkatkan efisiensi [27]. Teknik ini merupakan pengembangan dari konvolusi standar dengan memisahkan proses konvolusi menjadi dua tahap terpisah. Pada konvolusi standar, setiap filter berukuran $k \times k$ bekerja pada seluruh kanal input dan menghasilkan seluruh kanal output secara langsung. Kompleksitas komputasi konvolusi standar dapat dinyatakan sebagai:

$$h_i \cdot w_i \cdot d_i \cdot d_j \cdot k^2$$

Keterangan:

h_i = tinggi Feature map

w_i = lebar feature map

d_i = jumlah kanal input

d_j = jumlah kanal output

k = ukuran kernel

Kompleksitas, semakin besar jumlah kanal *input* dan *output*, maka jumlah operasi meningkat secara signifikan. Sebaliknya, pada *depthwise separable convolution*, proses konvolusi dipisahkan menjadi dua tahap. Kompleksitas komputasi metode ini menjadi:

$$h_i \cdot w_i \cdot d_i(k^2 + d_j)$$

Pemisahan ini membuat proses perhitungan jauh lebih ringan karena konvolusi spasial (*depthwise*) dilakukan secara terpisah pada setiap kanal, lalu digabungkan melalui konvolusi 1×1 (*pointwise*). Menurut Sandler *et al.* [27] ketika menggunakan kernel 3×3 , pendekatan ini dapat mengurangi biaya komputasi hingga sekitar 8-9 kali dibandingkan konvolusi standar, dengan penurunan akurasi yang relatif kecil. Oleh karena itu, *depthwise separable convolution* menjadi fondasi efisiensi pada arsitektur MobileNetV2.

b. *Linear Bottleneck*

Linear bottleneck merupakan salah satu kontribusi utama yang diperkenalkan dalam MobileNetV2 [27]. Konsep ini dikembangkan berdasarkan analisis bahwa representasi fitur dalam jaringan saraf dalam sering kali berada pada sub ruang berdimensi rendah. Pada kondisi tersebut, penggunaan fungsi aktivasi non-linear secara berulang dapat menyebabkan sebagian informasi penting hilang. Oleh karena itu, MobileNetV2 menerapkan linear bottleneck untuk membantu mempertahankan informasi pada ruang fitur berdimensi rendah sehingga representasi fitur yang dihasilkan tetap efektif untuk proses klasifikasi.

Pada arsitektur CNN konvensional, fungsi aktivasi *non-linear* seperti ReLU diterapkan setelah setiap operasi konvolusi. Namun, Sandler *et al.* [27] menunjukkan bahwa penerapan *non-linearitas* pada layer dengan jumlah kanal kecil (*bottleneck*) dapat menyebabkan hilangnya informasi. Hal ini terjadi karena ReLU memetakan nilai negatif menjadi nol. Ketika fitur telah direduksi ke dimensi rendah, pemotongan tersebut berpotensi menghilangkan informasi penting secara permanen. Untuk mempertahankan kekuatan representasi, MobileNetV2 tidak menggunakan fungsi aktivasi *non-linear* pada lapisan *bottleneck* akhir. Transformasi yang digunakan pada tahap proyeksi akhir bersifat *linear* [27]. Pendekatan ini memungkinkan fitur yang telah dikompresi tetap mempertahankan informasi penting tanpa mengalami distorsi akibat *non-linearitas*.

c. *Inverted Residual Structure*

Inverted residual merupakan inovasi arsitektural utama yang membedakan MobileNetV2 dari versi sebelumnya [27]. Struktur ini memanfaatkan koneksi *residual* pada layer berdimensi kecil (*bottleneck*), bukan pada layer berdimensi

besar seperti pada *residual block* tradisional. Satu blok *inverted residual* terdiri dari tiga tahap utama. Tahap pertama adalah ekspansi dimensi menggunakan konvolusi 1×1 dengan aktivasi ReLU6 untuk memperluas jumlah kanal. Tahap kedua adalah ekstraksi fitur spasial menggunakan *depthwise convolution* berukuran 3×3 . Tahap ketiga adalah proyeksi kembali ke dimensi kecil menggunakan konvolusi *linear* 1×1 tanpa fungsi aktivasi *non-linear*.

Koneksi *residual* diterapkan apabila dimensi *input* dan *output* sama. Desain ini memisahkan proses ekspansi fitur (untuk meningkatkan kapasitas representasi) dan proses kompresi fitur (untuk menjaga efisiensi). Sandler *et al.* [27] menjelaskan bahwa pendekatan ini memberikan efisiensi memori yang lebih baik serta tetap mendukung propagasi gradien secara stabil selama pelatihan. Struktur *inverted residual* bersama *linear bottleneck* menjadikan MobileNetV2 mampu mempertahankan keseimbangan antara efisiensi komputasi dan kemampuan representasi fitur.

8. Klasifikasi Citra

Klasifikasi citra merupakan proses pengelompokan suatu gambar ke dalam kategori tertentu berdasarkan informasi visual yang terkandung di dalamnya. Dalam konteks *computer vision*, tugas ini bertujuan agar sistem komputer mampu mengenali pola pada citra dan menentukan kelas yang paling sesuai secara otomatis [28]. Perkembangan metode klasifikasi citra mengalami perubahan signifikan sejak diperkenalkannya pendekatan berbasis *deep learning*. Sebelumnya, proses klasifikasi sangat bergantung pada fitur yang dirancang secara manual (*handcrafted features*). Namun, pendekatan ini memiliki keterbatasan karena kualitas hasil sangat ditentukan oleh desain fitur yang digunakan. Seiring perkembangan teknologi, *Convolutional Neural Network* (CNN) menjadi pendekatan utama karena mampu mempelajari representasi fitur secara otomatis langsung dari data citra [29].

CNN bekerja dengan cara melakukan ekstraksi fitur secara bertahap melalui beberapa lapisan konvolusi. Pada tahap awal, jaringan menangkap pola dasar seperti tepi atau tekstur. Selanjutnya, informasi tersebut diproses lebih dalam sehingga membentuk representasi yang lebih kompleks, misalnya pola bentuk atau struktur objek tertentu [28]. Mekanisme ini memungkinkan model membangun pemahaman visual secara hierarkis tanpa memerlukan perancangan fitur secara eksplisit.

Menurut Kim *et al.* [29] CNN telah menjadi pendekatan dominan dalam analisis citra karena kemampuannya melakukan pembelajaran secara langsung dari data mentah hingga menghasilkan prediksi kelas. Model dilatih menggunakan fungsi *loss* tertentu dan diperbarui melalui algoritma *backpropagation*, sehingga seluruh proses pembelajaran berlangsung secara terintegrasi (*end-to-end*).

Dalam praktiknya, keterbatasan jumlah data sering menjadi tantangan dalam klasifikasi citra. Oleh karena itu, strategi *transfer learning* banyak digunakan untuk meningkatkan performa model. Kim *et al.* [29] menjelaskan bahwa model yang telah dilatih pada dataset besar dapat dimanfaatkan kembali sebagai dasar pembelajaran untuk tugas klasifikasi lain, baik dengan membekukan sebagian parameter maupun menyesuaikannya kembali melalui proses *fine-tuning*. Pendekatan ini membantu mempercepat proses pelatihan sekaligus meningkatkan akurasi ketika data terbatas. Dengan demikian, klasifikasi citra modern berfokus pada pembelajaran representasi visual secara otomatis melalui arsitektur jaringan saraf dalam, terutama CNN, serta didukung oleh strategi pelatihan seperti *transfer learning* untuk meningkatkan performa dan efisiensi model.

9. Metrik Evaluasi Performa

Evaluasi performa merupakan tahap penting dalam pengembangan model klasifikasi untuk mengukur sejauh mana model mampu menghasilkan prediksi yang sesuai dengan label sebenarnya pada data uji. Pengukuran ini tidak hanya melihat tingkat ketepatan prediksi, tetapi juga mempertimbangkan keseimbangan antar kelas serta efisiensi model dalam implementasi praktis. Dalam sistem klasifikasi, evaluasi umumnya dilakukan menggunakan *confusion matrix* dan beberapa metrik turunan seperti *accuracy*, *precision*, *recall*, dan *F1-score* [30]. Selain itu, dalam pengembangan model *deep learning* modern, aspek efisiensi seperti waktu inferensi dan ukuran model juga menjadi indikator penting, terutama ketika model diimplementasikan pada perangkat dengan sumber daya terbatas [31].

a. *Confusion Matrix*

Confusion matrix merupakan tabel yang digunakan untuk membandingkan hasil prediksi model dengan label sebenarnya [30]. Dalam proses evaluasi model klasifikasi, confusion matrix menghasilkan empat komponen utama yang menjadi dasar perhitungan berbagai metrik evaluasi, yaitu:

		True Class	
		Positive	Negative
Predicted Class	Positive	TP	FP
	Negative	FN	TN

Gambar 4 Confusion Matrix

dengan :

- *True Positive* (TP) : data positif yang diprediksi sebagai positif
- *True Negative* (TN) : data negatif yang diprediksi sebagai negatif
- *False Positive* (FP) : data negatif yang diprediksi sebagai positif
- *False Negative* (FN) : data positif yang diprediksi sebagai negatif

Confusion matrix menjadi dasar dalam perhitungan berbagai metrik evaluasi lainnya. Pada penelitian ini, setiap percobaan (*run*) menghasilkan sebuah *confusion matrix* untuk masing-masing model, kemudian nilai accuracy, precision, recall, dan F1-score dihitung berdasarkan *confusion matrix* tersebut.

b. *Accuracy*

Accuracy mengukur proporsi jumlah prediksi yang benar dibandingkan dengan seluruh data yang di uji [30]. Perhitungan *accuracy* dapat dinyatakan dalam bentuk persamaan berikut:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Metrik ini mudah dipahami karena menunjukkan tingkat ketepatan model secara keseluruhan.

c. *Precision*

Precision menunjukkan tingkat ketepatan prediksi positif, yaitu seberapa banyak prediksi positif yang benar-benar sesuai dengan kondisi sebenarnya [30]. *Precision* dihitung sebagai perbandingan antara jumlah prediksi positif yang benar (TP) terhadap seluruh prediksi positif yang dihasilkan model (TP dan FP), yang dirumuskan sebagai:

$$Precision = \frac{TP}{TP + FP}$$

d. *Recall*

Recall atau sensitivitas menunjukkan kemampuan model dalam mendeteksi seluruh data yang benar-benar termasuk dalam kelas positif. Rumus *recall* adalah:

$$Recall = \frac{TP}{TP + FN}$$

Ketika nilai *recall* tinggi berarti model mampu menangkap sebagian besar data positif yang ada.

e. *F1-score*

F1-score merupakan kombinasi antara *precision* dan *recall* dalam bentuk rata-rata harmonik. Nilai *F1-score* diperoleh dari persamaan berikut:

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

Metrik ini digunakan untuk mengukur konsistensi performa klasifikasi pada setiap kelas secara seimbang, sehingga cocok digunakan pada penelitian dengan klasifikasi multikelas seperti penelitian ini yang melibatkan tujuh kelas denominasi uang kertas.

f. *Inference Time*

Inference time adalah waktu yang dibutuhkan model untuk menghasilkan prediksi terhadap suatu data masukan pada tahap pengujian. Dalam implementasi model *deep learning*, waktu inferensi menjadi indikator efisiensi komputasi, model dengan waktu inferensi yang lebih singkat dinilai lebih efisien dalam implementasi praktis [31].

g. Ukuran Model

Ukuran model merujuk pada jumlah parameter yang dimiliki oleh suatu jaringan saraf. Jumlah parameter berkaitan langsung dengan kebutuhan memori dan kompleksitas komputasi model. Dalam literatur arsitektur *deep learning*, jumlah parameter dan *multiply-adds* (MAdds) sering digunakan untuk menggambarkan *trade-off* antara akurasi dan efisiensi model [31].

h. Bahasa Pemrograman Python

Python merupakan bahasa pemrograman yang diciptakan oleh Guido van Rossum pada tahun 1991 [32]. Python merupakan bahasa pemrograman tingkat tinggi (*high-level programming language*) yang bersifat interpretatif, mudah dipelajari, dan memiliki sintaks yang jelas serta ringkas. Python bersifat terbuka (*open-source*) sehingga dapat digunakan dan dikembangkan secara bebas oleh berbagai kalangan. Python digunakan dalam pengembangan aplikasi, analisis data, serta pengembangan *machine learning* dan *deep learning* [20].

Dalam penelitian ini, Python digunakan sebagai bahasa pemrograman utama dalam proses pengolahan dataset citra uang kertas Rupiah, pembangunan model *Convolutional Neural Network* (CNN) *custom* dan *transfer learning* berbasis MobileNetV2, serta evaluasi performa model menggunakan metrik seperti *accuracy*, *precision*, *recall*, dan *F1-score*. Selain itu, Python juga digunakan dalam proses pelatihan model dan pengujian untuk memperoleh hasil klasifikasi yang optimal.

i. Google Colab

Google *Colaboratory* atau Google Colab merupakan platform berbasis cloud yang menyediakan lingkungan komputasi interaktif untuk menjalankan kode Python melalui *notebook* tanpa memerlukan instalasi perangkat lunak secara lokal. Platform ini berbasis *Jupyter Notebook* dan memungkinkan pengguna untuk menulis, menjalankan, serta mendokumentasikan kode dalam satu lingkungan kerja yang terintegrasi. Selain menyediakan CPU sebagai sumber daya komputasi bawaan, Google Colab juga menyediakan akses terhadap sumber daya komputasi seperti GPU dan TPU yang dapat digunakan untuk mempercepat proses pelatihan model *deep learning*.

Google Colab digunakan dalam bidang pendidikan dan penelitian karena kemudahan akses serta kemampuannya dalam mendukung eksperimen komputasi secara interaktif. Dalam penelitian ini, Google Colab digunakan sebagai media pengembangan dan pelatihan model *Convolutional Neural Network (CNN) custom* dan *transfer learning* berbasis MobileNetV2. Platform ini juga digunakan untuk menjalankan proses pengujian, serta evaluasi performa model secara efisien.

BAB III METODOLOGI

A. Waktu dan Tempat Penelitian

1. Waktu Penelitian

Penelitian ini dilakukan pada bulan Februari 2026 sampai dengan bulan Juni 2026. Penelitian ini diharapkan selesai dalam satu semester.

2. Tempat Penelitian

Penelitian ini dilakukan di Kabupaten Cilacap.

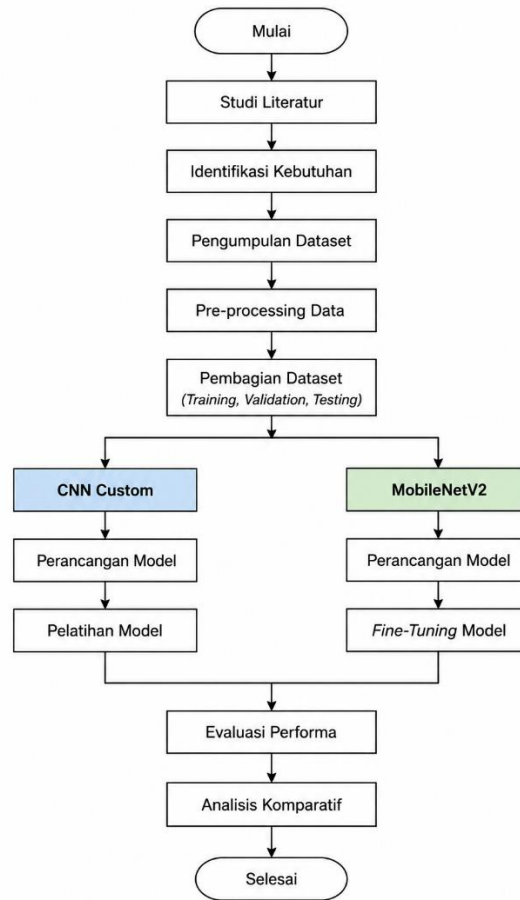
B. Alat dan Bahan

Tabel 2 Alat dan Bahan beserta keterangannya

No	Alat/Bahan	Keterangan
(1)	(2)	(3)
1	Laptop	ASUS Vivobook X1404VA dengan prosesor Intel® Core™ i3-1315U Generasi ke-13, RAM 16 GB, SSD 477 GB, dan sistem operasi Windows 11 64-bit, digunakan untuk pengolahan data, mengakses Google Colab, serta evaluasi performa model.
2	Google Colab	Menyediakan lingkungan komputasi untuk proses pelatihan dan pengujian model deep learning.
3	Pyhton	Bahasa pemrograman yang digunakan dalam implementasi model klasifikasi citra.
4	Dataset Citra Uang Kertas TE 2022	Sumber data penelitian untuk proses pelatihan, validasi, dan pengujian model.
5	Mendeley	Membantu pengelolaan referensi dan sitasi penelitian.

C. Prosedur Penelitian

Penelitian ini dilakukan melalui beberapa tahap untuk memastikan hasil akurat dan sesuai dengan tujuan penelitian. Adapun tahapan yang dilakukan ditunjukkan pada Gambar 5.



Gambar 5 Alur Penelitian

1. Studi Literatur

Studi literatur merupakan proses sistematis dalam mengumpulkan, mengevaluasi, dan mengintegrasikan penelitian-penelitian sebelumnya yang relevan untuk memahami perkembangan topik penelitian serta mengidentifikasi kesenjangan penelitian yang masih ada [33]. Studi literatur yang relevan dengan topik penelitian masih diperlukan karena literatur memuat konsep, metode, dan pendekatan yang digunakan sebagai dasar dalam proses perancangan dan pelaksanaan penelitian. Analisis terhadap berbagai sumber ilmiah berfokus pada topik pengolahan citra digital, *computer vision*, *deep learning*, serta metode klasifikasi citra menggunakan *Convolutional Neural Network* (CNN). Studi literatur juga mencakup pembahasan mengenai arsitektur *CNN custom*, konsep *transfer learning*, serta model MobileNetV2 sebagai salah satu arsitektur *lightweight* yang banyak digunakan dalam klasifikasi citra.

Berdasarkan studi yang dilakukan, arsitektur CNN telah banyak dimanfaatkan dalam berbagai penelitian klasifikasi citra, seperti pengenalan objek, klasifikasi penyakit, hingga deteksi uang kertas. Pendekatan *transfer*

learning menggunakan model *pre-trained* seperti MobileNetV2 juga terbukti mampu meningkatkan efisiensi pelatihan dan performa model, khususnya pada dataset yang terbatas. Namun, sebagian besar penelitian umumnya hanya berfokus pada satu pendekatan saja, baik menggunakan CNN *custom* maupun *transfer learning*, tanpa melakukan analisis komparatif secara langsung dalam konteks permasalahan yang sama.

Penelitian yang secara khusus membandingkan performa CNN *custom* dan MobileNetV2 pada klasifikasi citra uang kertas Rupiah Tahun Emisi 2022 masih relatif terbatas. Selain itu, kajian yang tidak hanya membandingkan akurasi, tetapi juga mempertimbangkan aspek efisiensi komputasi seperti waktu inferensi dan ukuran model, belum banyak dilakukan secara mendalam. Oleh karena itu, penelitian ini diarahkan untuk melakukan analisis komparatif antara arsitektur CNN *custom* dan pendekatan *transfer learning* berbasis MobileNetV2 dalam klasifikasi citra uang kertas Rupiah Tahun Emisi 2022. Pendekatan ini diharapkan dapat memberikan gambaran yang lebih jelas mengenai keunggulan masing-masing metode, baik dari segi performa klasifikasi maupun efisiensi komputasi, sehingga dapat menjadi referensi dalam pengembangan sistem klasifikasi citra yang optimal.

2. Identifikasi Kebutuhan

Identifikasi kebutuhan dilakukan untuk menentukan berbagai komponen yang diperlukan sebelum proses perancangan dan pelatihan model dilakukan. Kebutuhan tersebut meliputi dataset citra uang kertas Rupiah Tahun Emisi 2022, perangkat keras, serta perangkat lunak yang digunakan dalam penelitian. Dataset yang dibutuhkan berupa citra uang kertas Rupiah Tahun Emisi 2022 dengan tujuh kelas denominasi sebagai data utama dalam proses pelatihan dan pengujian model klasifikasi. Selain kebutuhan data, penelitian ini juga memerlukan perangkat keras yang mampu mendukung proses komputasi deep learning serta perangkat lunak seperti Python dan Google Colab untuk proses pengembangan, pelatihan, dan evaluasi model.

3. Perancangan dan Pelatihan Model

a. Akuisisi Dataset

Akuisisi dataset merupakan tahap pengumpulan citra yang akan digunakan untuk melatih dan menguji model klasifikasi. Dataset yang digunakan dalam penelitian ini berupa citra uang kertas Rupiah Tahun Emisi 2022 yang diperoleh dari dua dataset publik pada platform Kaggle, yaitu dataset Citra Uang Rupiah Emisi 2022 oleh Dani Akhmad Maulana dan dataset Uang Emisi 2022 Baru oleh Fanny Zahra Ramadhan. Dataset ini terdiri dari 2.100 citra yang terbagi ke dalam tujuh kelas denominasi, yaitu Rp1.000, Rp2.000, Rp5.000, Rp10.000, Rp20.000, Rp50.000, dan

Rp100.000, dengan masing-masing kelas berisi 300 citra. Jumlah tersebut diperoleh melalui penggabungan citra dari kedua dataset, karena jumlah citra pada beberapa kelas di masing-masing dataset belum mencapai 300 citra per kelas.

Jumlah citra yang sama pada setiap kelas menjadikan dataset ini bersifat seimbang. Kondisi ini penting karena apabila salah satu kelas memiliki jumlah data yang jauh lebih banyak dibandingkan kelas lainnya, model akan cenderung lebih mudah mengenali kelas tersebut dan kurang optimal dalam mengenali kelas dengan jumlah data yang lebih sedikit. Selain itu, citra dalam dataset ini diambil dalam berbagai kondisi pengambilan gambar, seperti perbedaan pencahayaan, sudut pandang, dan latar belakang, sehingga model yang dilatih diharapkan tidak hanya mampu mengenali citra dalam kondisi ideal, tetapi juga tetap mampu melakukan klasifikasi dengan baik terhadap citra yang tidak diikutsertakan dalam proses pelatihan.

b. Pra Pemrosesan Data

Seluruh citra melalui tahap pra-pemrosesan agar memiliki ukuran dan rentang nilai piksel yang seragam sebelum digunakan pada proses pelatihan model [34].

Langkah pertama adalah *resize*, yaitu mengubah ukuran seluruh citra menjadi 224×224 piksel agar seluruh data memiliki dimensi yang seragam pada proses pelatihan model. Ukuran 224×224 piksel dipilih karena merupakan salah satu ukuran input yang umum digunakan pada model klasifikasi citra berbasis CNN, termasuk MobileNetV2 [27]. Ukuran tersebut juga masih mampu mempertahankan informasi visual penting pada citra uang kertas, seperti warna, pola, dan elemen keamanan, tanpa membutuhkan komputasi yang terlalu besar sehingga proses pelatihan dapat berjalan lebih efisien.

Langkah kedua adalah normalisasi, yaitu mengubah nilai piksel setiap citra dari rentang aslinya 0–255 menjadi rentang 0–1 dengan cara membagi setiap nilai piksel dengan angka 255. Normalisasi dilakukan agar seluruh nilai piksel berada pada skala yang seragam sehingga proses pembelajaran menjadi lebih stabil dan model lebih cepat mencapai kondisi optimal [34].

Langkah ketiga adalah augmentasi data, yang hanya diterapkan pada data training. Augmentasi merupakan teknik memperbanyak variasi citra secara buatan tanpa menambahkan citra baru [34]. Pada penelitian ini, augmentasi dilakukan melalui rotasi, *flip horizontal*, *zoom in/out*, dan pergeseran posisi gambar. Tujuannya agar model tidak hanya bergantung pada satu kondisi pengambilan gambar saja, melainkan mampu mengenali uang kertas pada berbagai kondisi pengambilan gambar. Data *validation* dan *testing* tidak

diaugmentasi karena keduanya digunakan untuk mengukur performa model pada kondisi data yang sesungguhnya.

c. Split Data (*Train, Validation, Test*)

Dataset yang telah melalui tahap pra-pemrosesan kemudian dibagi menjadi tiga bagian, yaitu *data training*, *validation*, dan *testing* [35]. Pembagian ini dilakukan agar evaluasi performa model dapat dilakukan secara objektif pada data yang tidak pernah diikuti sertakan dalam proses pelatihan. Rasio yang digunakan adalah 70:15:15 karena umum digunakan pada penelitian *deep learning* dengan dataset berukuran sedang sehingga tetap menyediakan *data training* yang memadai sekaligus menyediakan *data validation* dan *testing* yang *representative* [36]. Jumlah *data training* dibuat lebih besar karena model memerlukan data yang cukup banyak untuk mempelajari pola dan karakteristik pada citra uang kertas, sedangkan *data validation* dan *testing* digunakan sebagai data evaluasi selama dan setelah proses pelatihan.

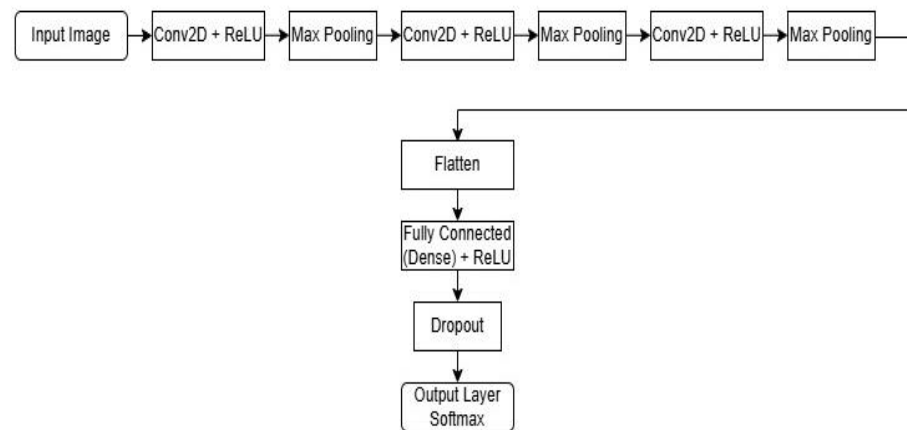
Bagian pertama adalah *data training* sebanyak 1.470 citra (70%), yaitu data yang digunakan langsung oleh model untuk belajar. Pada tahap ini, model melihat citra beserta kelas denominasinya, lalu secara bertahap menyesuaikan bobotnya, yaitu nilai-nilai numerik di dalam jaringan yang menentukan bagaimana model memproses sebuah citra, agar prediksinya semakin mendekati jawaban yang benar.

Bagian kedua adalah *data validation* sebanyak 315 citra (15%). Data ini tidak digunakan untuk melatih model, melainkan untuk memantau perkembangan model selama proses pelatihan berlangsung. Setiap kali satu *epoch* selesai, yaitu ketika model telah melihat seluruh *data training* satu kali penuh, model dievaluasi menggunakan *data validation*. Tujuannya adalah untuk memastikan bahwa peningkatan performa pada *data training* benar-benar mencerminkan kemampuan model secara umum, bukan karena model mempelajari pola *data training* secara berlebihan. Kondisi di mana model menunjukkan performa sangat baik pada *data training* tetapi buruk pada data lainnya disebut *overfitting*, dan *data validation* berperan penting dalam mendeteksi kondisi ini sejak dini [35].

Bagian ketiga adalah *data testing* sebanyak 315 citra (15%). Data ini sama sekali tidak terlibat dalam proses pelatihan maupun validasi, dan hanya digunakan satu kali di akhir setelah proses pelatihan selesai untuk mengukur performa akhir model secara objektif.

d. Perancangan Model CNN Custom

CNN *custom* merupakan arsitektur CNN yang dirancang dan dilatih dari awal (*from scratch*) tanpa memanfaatkan bobot dari model yang telah dilatih sebelumnya, sehingga seluruh parameter jaringan dipelajari langsung dari *data training* yang tersedia. Arsitektur CNN *custom* yang digunakan dalam penelitian ini dapat dilihat pada Gambar 6. Model diawali dengan input citra berukuran 224×224 piksel yang kemudian diproses melalui tiga blok konvolusi, masing-masing terdiri dari Conv2D, fungsi aktivasi ReLU, dan *MaxPooling*. Setelah proses ekstraksi fitur selesai, *feature map* diratakan menggunakan *Flatten layer* sebelum diteruskan ke *Dense layer* dan Dropout. Pada lapisan terakhir, output layer menggunakan fungsi aktivasi *Softmax* untuk menghasilkan probabilitas klasifikasi ke dalam tujuh kelas denominasi uang kertas.



Gambar 6 Arsitektur CNN Custom

Arsitektur ini terdiri atas tiga blok konvolusi. Setiap blok terdiri atas lapisan Conv2D, fungsi aktivasi ReLU, dan *MaxPooling*. Conv2D bekerja dengan cara menggeser filter berukuran 3×3 piksel ke seluruh bagian citra untuk mendeteksi pola visual seperti tepi, tekstur, atau bentuk. ReLU kemudian mengolah hasil konvolusi dengan mengubah semua nilai negatif menjadi nol sehingga jaringan dapat mempelajari pola yang lebih kompleks. *MaxPooling* bertugas menyederhanakan hasil tersebut dengan mengambil nilai tertinggi dari setiap area 2×2 piksel sehingga dimensi *feature map* mengecil dan proses komputasi menjadi lebih ringan.

Jumlah filter pada setiap blok ditingkatkan secara bertahap, yaitu 32 filter pada blok pertama, 64 filter pada blok kedua, dan 128 filter pada blok ketiga, mengikuti prinsip ekstraksi fitur hierarkis di mana lapisan awal menangkap fitur sederhana dan lapisan lebih dalam menangkap fitur yang semakin kompleks [37]. Setelah ketiga blok konvolusi, hasil ekstraksi fitur diratakan

menggunakan *Flatten layer*, yaitu lapisan yang mengubah seluruh *feature map* menjadi vektor satu dimensi sebelum diteruskan ke *fully connected layer* [7].

Hasil tersebut kemudian diteruskan ke *Dense layer* dengan 128 neuron dan fungsi aktivasi ReLU untuk proses klasifikasi. Jumlah neuron tersebut dipilih untuk memberikan kapasitas pembelajaran yang cukup dalam mempelajari hubungan antar fitur hasil ekstraksi tanpa membuat kompleksitas model menjadi terlalu besar. *Dropout* dengan rate 0,5 diterapkan setelah *Dense layer*. Teknik ini secara acak menonaktifkan sebagian neuron selama pelatihan agar model tidak terlalu bergantung pada pola tertentu saja dan lebih mampu melakukan generalisasi [38].

Pada lapisan terakhir, *output layer* menggunakan 7 neuron yang masing-masing mewakili satu kelas denominasi uang kertas, dengan fungsi aktivasi *Softmax* yang mengubah keluaran menjadi nilai probabilitas sehingga kelas dengan probabilitas tertinggi menjadi prediksi akhir model.

e. *Training CNN Custom (Early Stopping)*

Training CNN custom (Early Stopping) Proses *training* merupakan tahap di mana model *CNN custom* mempelajari pola dari data. Model dilatih menggunakan 1.470 citra data training dengan *batch size* 32. Nilai ini dipilih karena *batch size* kecil terbukti mampu menghasilkan generalisasi model yang lebih baik sekaligus lebih efisien dari sisi penggunaan memori selama proses pelatihan [39]. Setiap kali model menghasilkan prediksi, dihitung seberapa jauh hasil prediksi tersebut dari label sebenarnya menggunakan fungsi *loss categorical crossentropy*, yaitu fungsi *loss* yang umum digunakan pada permasalahan klasifikasi multi-kelas [35]. Berdasarkan nilai kesalahan tersebut, bobot jaringan diperbarui melalui proses *backpropagation*, yaitu proses menghitung dan menyebarkan informasi kesalahan ke seluruh lapisan jaringan agar setiap bobot dapat diperbaiki ke arah yang mengurangi kesalahan prediksi [5].

Optimizer yang digunakan adalah Adam dengan *learning rate* 0,001. Adam dipilih karena mampu menyesuaikan besar pembaruan bobot secara adaptif pada setiap parameter sehingga proses pelatihan menjadi lebih stabil dan efisien [40].

Pelatihan dilakukan dengan maksimum 50 *epoch*. Namun, untuk menghindari *overfitting* dan pemborosan waktu komputasi, diterapkan mekanisme *early stopping* dengan *patience* 10 *epoch*. Artinya, apabila dalam 10 *epoch* berturut-turut nilai *validation loss* tidak mengalami penurunan, proses pelatihan akan dihentikan secara otomatis meskipun belum mencapai jumlah *epoch* maksimum [9]. Setiap model dilatih

sebanyak lima kali percobaan dengan konfigurasi parameter yang identik. Pengulangan percobaan dilakukan karena proses pelatihan deep learning dapat menghasilkan nilai performa yang berbeda pada setiap pelatihan meskipun menggunakan konfigurasi parameter yang sama. Perbedaan tersebut dipengaruhi oleh proses inisialisasi bobot awal yang dilakukan secara acak pada setiap pelatihan [35]. Pengulangan ini bertujuan untuk memperoleh gambaran performa model yang lebih konsisten dan mengurangi ketergantungan terhadap hasil dari satu kali pelatihan. Hasil dari setiap percobaan selanjutnya dievaluasi menggunakan data uji berdasarkan metrik evaluasi yang telah ditetapkan dalam penelitian ini.

f. *Evaluasi Test CNN Custom*

Setelah proses pelatihan selesai, model *CNN custom* dievaluasi menggunakan 315 citra *data testing* yang tidak pernah diikutsertakan dalam proses pelatihan maupun validasi. Evaluasi model merupakan tahap pengukuran kemampuan model yang telah dilatih dalam melakukan klasifikasi pada data baru yang belum pernah dilihat sebelumnya [35]. Tahap ini bertujuan untuk mengukur kemampuan model dalam mengklasifikasikan citra uang kertas secara objektif.

Evaluasi dilakukan dengan menghitung enam metrik, yaitu *accuracy*, *precision*, *recall*, *F1-score*, *inference time*, dan ukuran model. *Accuracy* digunakan untuk mengukur proporsi prediksi yang benar secara keseluruhan, *precision* digunakan untuk mengukur ketepatan prediksi pada setiap kelas, sedangkan *recall* digunakan untuk mengukur kemampuan model dalam mendeteksi seluruh citra yang benar-benar termasuk dalam suatu kelas. *F1-score* digunakan sebagai ukuran keseimbangan antara *precision* dan *recall* sehingga mampu memberikan gambaran performa model secara lebih menyeluruh [30].

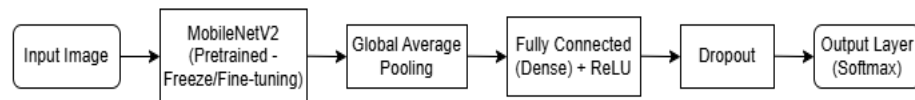
Aspek efisiensi komputasi juga dievaluasi melalui *inference time* dan ukuran model. *Inference time* digunakan untuk mengukur rata-rata waktu yang dibutuhkan model dalam menghasilkan prediksi per citra, sedangkan ukuran model digunakan untuk mengetahui kompleksitas dan kebutuhan memori model. Evaluasi juga menghasilkan *confusion matrix* untuk melihat distribusi prediksi model pada setiap kelas denominasi secara lebih detail sehingga dapat diidentifikasi kelas yang paling sering mengalami kesalahan klasifikasi [30].

g. *Load Pre-trained Model*

Load pre-trained model merupakan tahap memuat bobot model yang telah dilatih sebelumnya pada dataset berskala besar sebagai titik awal pelatihan pada dataset baru [12]. Pendekatan ini memanfaatkan kemampuan

representasi fitur yang telah dipelajari model dari data sebelumnya sehingga model tidak perlu memulai proses pembelajaran dari awal dan proses pelatihan pada dataset target menjadi lebih efisien.

Arsitektur MobileNetV2 yang digunakan dalam penelitian ini dapat dilihat pada Gambar 7. Model diawali dengan input citra berukuran 224×224 piksel yang kemudian diproses menggunakan base model MobileNetV2 sebagai *feature extractor* dengan memanfaatkan bobot *pretrained* dari *ImageNet*. Hasil ekstraksi fitur kemudian diproses melalui *Global Average Pooling* untuk mereduksi dimensi *feature map* menjadi vektor fitur yang lebih ringkas [27], kemudian dilanjutkan ke *Dense layer* dan *Dropout*, hingga pada *output layer* menggunakan fungsi aktivasi *Softmax* untuk menghasilkan probabilitas klasifikasi ke dalam tujuh kelas denominasi uang kertas.



Gambar 7 Arsitektur MobileNetV2

Berbeda dengan CNN *custom* yang dibangun dari awal tanpa bobot *pretrained*, pendekatan MobileNetV2 dimulai dengan memuat model yang telah dilatih sebelumnya pada dataset *ImageNet* [41]. *ImageNet* merupakan dataset berskala besar yang berisi lebih dari 1 juta citra dari 1.000 kategori objek berbeda. Melalui pelatihan pada dataset tersebut, MobileNetV2 telah memiliki kemampuan mengenali berbagai fitur visual umum seperti tepi, tekstur, gradien warna, dan bentuk objek. Kemampuan inilah yang disebut sebagai bobot *pretrained*, yaitu parameter jaringan hasil pelatihan pada *ImageNet* yang tersimpan di dalam setiap lapisan jaringan MobileNetV2. Dengan memanfaatkan bobot ini sebagai titik awal, model tidak perlu memulai proses pembelajaran dari awal sehingga proses pelatihan pada dataset uang kertas Rupiah menjadi lebih efisien.

Lapisan klasifikasi bawaan MobileNetV2 yang pada awalnya dirancang untuk 1.000 kelas *ImageNet* diganti dengan lapisan klasifikasi baru yang disesuaikan dengan kebutuhan penelitian ini, yaitu *Global Average Pooling* untuk merangkum *feature map* menjadi vektor fitur yang lebih ringkas [27]. *Dense layer* dengan 128 neuron dan fungsi aktivasi ReLU ditambahkan untuk memberikan kapasitas pembelajaran tambahan yang spesifik terhadap karakteristik visual uang kertas Rupiah TE 2022, *Dropout* dengan rate 0,5 untuk mencegah *overfitting*, serta *output layer* dengan 7 neuron yang masing-masing mewakili satu kelas denominasi uang kertas. Fungsi aktivasi

Softmax digunakan untuk mengubah keluaran menjadi nilai probabilitas sehingga kelas dengan probabilitas tertinggi menjadi prediksi akhir model.

h. *Fine-Tuning MobileNetV2 (Early Stopping)*

Fine-tuning merupakan tahap penyesuaian bobot model pretrained pada dataset target dengan cara membuka (*unfreeze*) sebagian layer yang sebelumnya dibekukan dan melatihnya kembali menggunakan *learning rate* yang kecil [42]. Pelatihan MobileNetV2 pada penelitian ini dilakukan dalam dua tahap.

Pada tahap pertama, seluruh layer base model MobileNetV2 dibekukan (*freeze*), artinya bobot hasil pelatihan pada dataset ImageNet tidak diubah selama proses pelatihan. Pada tahap ini, hanya lapisan klasifikasi baru yang dilatih. Tujuannya agar kemampuan ekstraksi fitur yang telah dimiliki MobileNetV2 tetap terjaga, sementara lapisan klasifikasi baru dapat mempelajari karakteristik visual citra uang kertas Rupiah. Tahap ini menggunakan *optimizer* Adam dengan *learning rate* sebesar 0,0001. Penggunaan *learning rate* yang lebih kecil dibandingkan CNN custom dilakukan agar proses pembaruan bobot *pretrained* berlangsung lebih stabil dan tidak menyebabkan perubahan signifikan pada representasi fitur yang telah dipelajari sebelumnya [40].

Tahap kedua merupakan proses fine-tuning yang dimulai setelah performa model pada tahap pertama mulai stabil. Pada tahap ini, sejumlah layer terakhir pada base model MobileNetV2 dibuka (*unfreeze*) dan dilatih ulang bersama lapisan klasifikasi baru. Layer-layer akhir dipilih karena cenderung mempelajari representasi fitur tingkat tinggi yang lebih spesifik terhadap dataset sumber, sehingga perlu disesuaikan kembali dengan karakteristik visual uang kertas Rupiah sebagai dataset target. Sementara itu, layer-layer awal tetap dibekukan karena menyimpan fitur-fitur dasar yang bersifat umum seperti tepi, garis, dan tekstur yang masih relevan untuk berbagai jenis citra sehingga tidak perlu dilatih ulang [42]. Jumlah layer yang dibuka pada tahap fine-tuning mengikuti praktik umum, yaitu membuka lapisan-lapisan akhir yang mempelajari fitur spesifik terhadap dataset asal (ImageNet), sementara lapisan awal yang mempelajari fitur umum tetap dibekukan.

Proses *fine-tuning* menggunakan *learning rate* yang lebih kecil yaitu sebesar 0,00001 agar penyesuaian bobot berlangsung secara bertahap tanpa menghilangkan representasi fitur yang telah dipelajari sebelumnya dari ImageNet [42]. Kedua tahap pelatihan menggunakan *batch size* 32, maksimum 50 epoch. Early stopping dengan patience 10 epoch diterapkan untuk memantau nilai *validation loss* dan mencegah *overfitting* [9]. Setiap

model dilatih sebanyak lima kali percobaan dengan konfigurasi parameter yang identik. Pengulangan percobaan dilakukan karena proses pelatihan deep learning dapat menghasilkan nilai performa yang berbeda pada setiap pelatihan meskipun menggunakan konfigurasi parameter yang sama. Perbedaan tersebut dipengaruhi oleh proses inisialisasi bobot awal yang dilakukan secara acak pada setiap pelatihan [35]. Pengulangan ini bertujuan untuk memperoleh gambaran performa model yang lebih konsisten dan mengurangi ketergantungan terhadap hasil dari satu kali pelatihan. Hasil dari setiap percobaan selanjutnya dievaluasi menggunakan data uji berdasarkan metrik evaluasi yang telah ditetapkan dalam penelitian ini.

i. Evaluasi *Test* MobileNetV2

Evaluasi pada *data testing* dilakukan untuk mengukur performa model MobileNetV2 dalam mengklasifikasikan citra uang kertas pada data yang tidak pernah digunakan selama proses pelatihan maupun validasi [35]. Model dievaluasi menggunakan *data testing* yang sama dengan yang digunakan pada CNN *custom*, yaitu sebanyak 315 citra. Penggunaan *data testing* yang sama pada kedua model bertujuan agar proses perbandingan dilakukan secara adil dan objektif, sehingga perbedaan hasil yang diperoleh benar-benar mencerminkan perbedaan performa kedua arsitektur dan bukan disebabkan oleh perbedaan data pengujian.

Evaluasi dilakukan menggunakan metrik yang sama seperti pada CNN *custom*, yaitu *accuracy*, *precision*, *recall*, dan *F1-score* [30]. *Accuracy* digunakan untuk mengukur proporsi prediksi yang benar secara keseluruhan. *Precision* digunakan untuk mengukur ketepatan prediksi pada setiap kelas, sedangkan *recall* digunakan untuk mengukur kemampuan model dalam mendeteksi seluruh citra yang benar-benar termasuk dalam suatu kelas. *F1-score* digunakan sebagai keseimbangan antara *precision* dan *recall* sehingga performa model dapat dievaluasi secara lebih menyeluruh [30].

Aspek efisiensi komputasi model juga dievaluasi melalui *inference time* dan ukuran model [27], [31]. *Inference time* digunakan untuk mengukur rata-rata waktu yang dibutuhkan model dalam menghasilkan prediksi terhadap satu citra, sedangkan ukuran model digunakan untuk mengetahui kompleksitas model dan kebutuhan memori yang diperlukan.

Confusion matrix juga digunakan untuk melihat distribusi hasil prediksi model pada setiap kelas denominasi uang kertas secara lebih rinci [30]. Melalui *confusion matrix*, dapat diidentifikasi kelas-kelas yang memiliki tingkat kesalahan klasifikasi tertinggi sehingga proses evaluasi tidak hanya

berfokus pada nilai akurasi keseluruhan, tetapi juga pada performa model terhadap masing-masing kelas.

j. Analisis Komparatif

Analisis perbandingan dilakukan setelah kedua model, yaitu CNN *custom* dan MobileNetV2, selesai dievaluasi menggunakan *data testing*. Analisis dilakukan melalui dua pendekatan untuk memperoleh gambaran perbandingan performa yang komprehensif [30].

Perbandingan dilakukan dengan membandingkan nilai setiap metrik evaluasi dari kedua model secara berdampingan menggunakan tabel dan grafik perbandingan. Metrik yang dibandingkan meliputi *accuracy*, *precision*, *recall*, dan *F1-score* untuk menilai performa klasifikasi dari berbagai aspek, serta *inference time* dan ukuran model untuk menilai efisiensi komputasi[30]. Perbandingan dilakukan secara menyeluruh, baik pada nilai rata-rata keseluruhan maupun pada nilai per kelas denominasi, agar perbedaan performa antar model dapat diidentifikasi secara lebih rinci.

Confusion matrix dari masing-masing model juga diinterpretasikan untuk memahami pola kesalahan klasifikasi yang terjadi [30]. Melalui *confusion matrix*, dapat diidentifikasi kelas denominasi mana yang paling sering salah diklasifikasikan serta kelas mana yang paling sering menjadi hasil prediksi yang keliru. Hasil interpretasi ini kemudian dikaitkan dengan karakteristik visual uang kertas Rupiah, seperti kemiripan warna dominan atau kesamaan pola antar denominasi tertentu yang kemungkinan menjadi penyebab kesalahan klasifikasi.

Hasil evaluasi kedua model selanjutnya diinterpretasikan berdasarkan karakteristik masing-masing arsitektur. Perbedaan performa CNN *custom* dan MobileNetV2 akan dianalisis mengacu pada kemampuan ekstraksi fitur masing-masing arsitektur, di mana CNN *custom* mempelajari representasi fitur secara langsung dari dataset penelitian ini, sedangkan MobileNetV2 memanfaatkan bobot pretrained dari *ImageNet* yang telah memiliki kemampuan mengenali berbagai fitur visual umum [22] [27]. Analisis ini bertujuan untuk memahami apakah pendekatan transfer learning pada MobileNetV2 mampu memberikan performa yang lebih baik dibandingkan CNN *custom* dalam konteks klasifikasi citra uang kertas Rupiah Tahun Emisi 2022.

k. Kesimpulan

Kesimpulan penelitian ditarik berdasarkan hasil analisis komparatif performa kedua model yang telah dievaluasi menggunakan data testing. Penentuan model yang paling optimal dilakukan dengan

mempertimbangkan seluruh metrik evaluasi secara komplementer, karena setiap metrik memberikan sudut pandang evaluasi yang berbeda terhadap performa model.

Accuracy dijadikan acuan utama dalam menentukan model terbaik karena mengukur proporsi prediksi yang benar secara keseluruhan. Mengingat dataset penelitian ini bersifat seimbang dengan jumlah citra yang sama pada setiap kelas denominasi, *accuracy* merupakan indikator yang representatif untuk melihat performa klasifikasi model secara keseluruhan. *F1-score*, *precision*, dan *recall* tetap digunakan untuk mengevaluasi performa klasifikasi pada setiap kelas denominasi secara lebih detail, sehingga dapat diketahui apakah model memiliki performa yang merata pada seluruh kelas atau hanya unggul pada kelas tertentu saja. Efisiensi komputasi kedua model juga dievaluasi melalui *inference time* dan ukuran model. Kedua aspek ini memberikan gambaran mengenai kecepatan prediksi serta kebutuhan sumber daya yang diperlukan dalam implementasi model.

4. Evaluasi Performa

Evaluasi model dilakukan menggunakan *data testing* yang tidak terlibat dalam proses pelatihan maupun validasi. Tujuan dari tahap ini adalah untuk mengukur kemampuan kedua model dalam mengklasifikasikan citra uang kertas Rupiah secara akurat dan konsisten pada data yang belum pernah dilihat sebelumnya [35].

Evaluasi dilakukan menggunakan beberapa metrik yang saling melengkapi karena setiap metrik memberikan informasi performa yang berbeda. Metrik evaluasi yang digunakan meliputi *Accuracy*, *Precision*, *Recall*, *F1-score*, *Inference Time*, dan Ukuran Model, sebagaimana ditunjukkan pada Tabel 3.

Tabel 3 Indikator Evaluasi Performa Model

No	Metrik Evaluasi	Tujuan Pengukuran
(1)	(2)	(3)
1.	<i>Accuracy</i>	Mengukur proporsi prediksi yang benar secara keseluruhan.
2.	<i>Precision</i>	Mengukur ketepatan prediksi pada tiap kelas.
3.	<i>Recall</i>	Mengukur kemampuan model dalam mendeteksi seluruh data pada tiap kelas.
4.	<i>F1-score</i>	Mengukur keseimbangan antara <i>precision</i> dan <i>recall</i> .
5.	<i>Inference Time</i>	Mengukur rata-rata waktu yang dibutuhkan model untuk menghasilkan satu prediksi.

No	Metrik Evaluasi	Tujuan Pengukuran
(1)	(2)	(3)
6.	Ukuran Model	Mengukur besar file model (MB) sebagai indikator efisiensi penyimpanan model.

Metrik klasifikasi digunakan untuk mengukur performa model dalam melakukan prediksi terhadap setiap kelas denominasi uang kertas. *Accuracy* digunakan untuk melihat performa prediksi secara keseluruhan, sedangkan *precision*, *recall*, dan *F1-score* digunakan untuk mengevaluasi performa klasifikasi pada setiap kelas. Sementara itu, *inference time* dan ukuran model digunakan untuk menilai efisiensi komputasi model dalam hal kecepatan prediksi dan kompleksitas model.

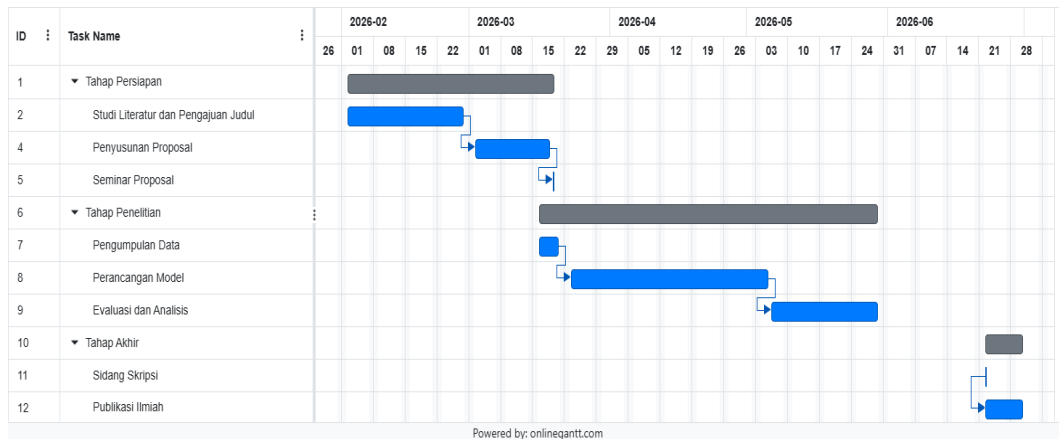
5. Analisis Komparatif

Analisis komparatif dilakukan dengan membandingkan hasil evaluasi dari kedua model, yaitu CNN Custom dan MobileNetV2, berdasarkan metrik yang telah ditentukan. Penentuan model yang paling optimal dilakukan dengan mempertimbangkan seluruh metrik evaluasi secara komplementer, karena setiap metrik memberikan sudut pandang evaluasi yang berbeda terhadap performa model.

Penentuan model terbaik dilakukan dengan menjadikan accuracy sebagai indikator utama karena dataset penelitian ini memiliki jumlah citra yang seimbang pada setiap kelas. Sementara itu, nilai precision, recall, dan F1-score digunakan sebagai informasi pendukung untuk mengevaluasi kualitas klasifikasi pada setiap kelas secara lebih rinci. Selain itu, inference time dan ukuran model digunakan untuk membandingkan efisiensi komputasi kedua model. Hasil evaluasi dari lima kali percobaan pada masing-masing model kemudian direkapitulasi dan dihitung nilai rata-ratanya untuk memperoleh gambaran performa kedua model secara keseluruhan. Berdasarkan hasil rekapitulasi tersebut dilakukan analisis komparatif untuk menentukan model yang memiliki performa klasifikasi dan efisiensi komputasi yang lebih baik.

D. Jadwal Penelitian

Jadwal penelitian disusun untuk mengatur alokasi waktu agar seluruh tahapan berjalan terarah dan terstruktur. Dengan adanya jadwal ini, peneliti dapat memastikan bahwa setiap kegiatan penelitian dilaksanakan sesuai dengan rencana. Jadwal penelitian dibuat dalam bentuk *Gantt Chart* yang memudahkan visualisasi tahapan kegiatan beserta durasi waktunya secara rinci. Berikut ini *Gantt Chart* jadwal penelitian dapat dilihat pada Gambar 8.



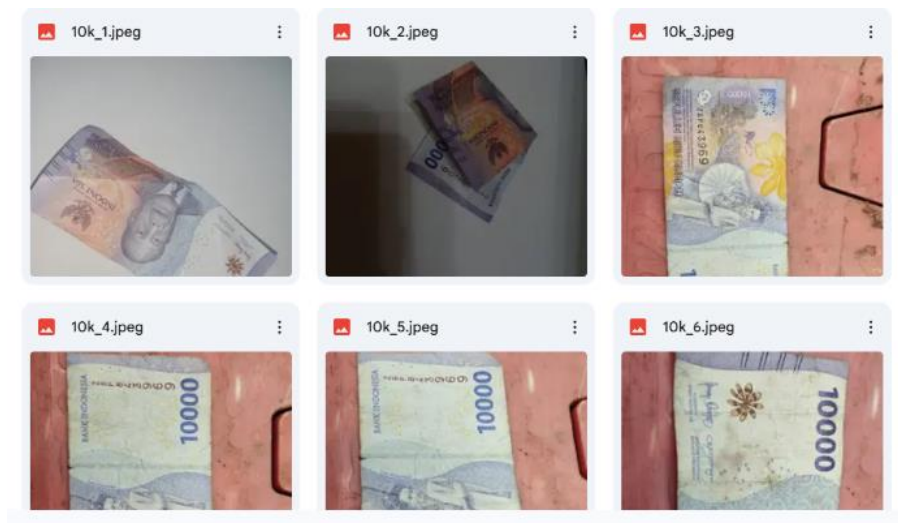
Gambar 8 *Gantt Chart* Jadwal Penelitian

BAB IV HASIL DAN PEMBAHASAN

A. Hasil

1. Dataset Pengujian

Dataset yang digunakan dalam penelitian ini terdiri atas 2.100 citra uang kertas Rupiah Tahun Emisi (TE) 2022. Dataset tersebut mencakup tujuh kelas nominal, yaitu Rp1.000, Rp2.000, Rp5.000, Rp10.000, Rp20.000, Rp50.000, dan Rp100.000. Setiap kelas terdiri atas 300 citra, sehingga jumlah data pada masing-masing kelas bersifat seimbang.



Gambar 9 Contoh Dataset

Dataset diperoleh dari dua sumber di Kaggle yang digabungkan untuk memenuhi jumlah data yang memadai pada setiap kelas nominal. Dataset kemudian dibagi menjadi tiga subset menggunakan rasio 70% : 15% : 15% yang terdiri atas data latih, data validasi, dan data uji. Jumlah data pada masing-masing subset berturut-turut adalah 1.470 citra, 315 citra, dan 315 citra. Dengan pembagian tersebut, setiap kelas memiliki 210 citra pada data latih, 45 citra pada data validasi, dan 45 citra pada data uji. Rincian pembagian dataset ditunjukkan pada Tabel 4.

Tabel 4 Pembagian Dataset

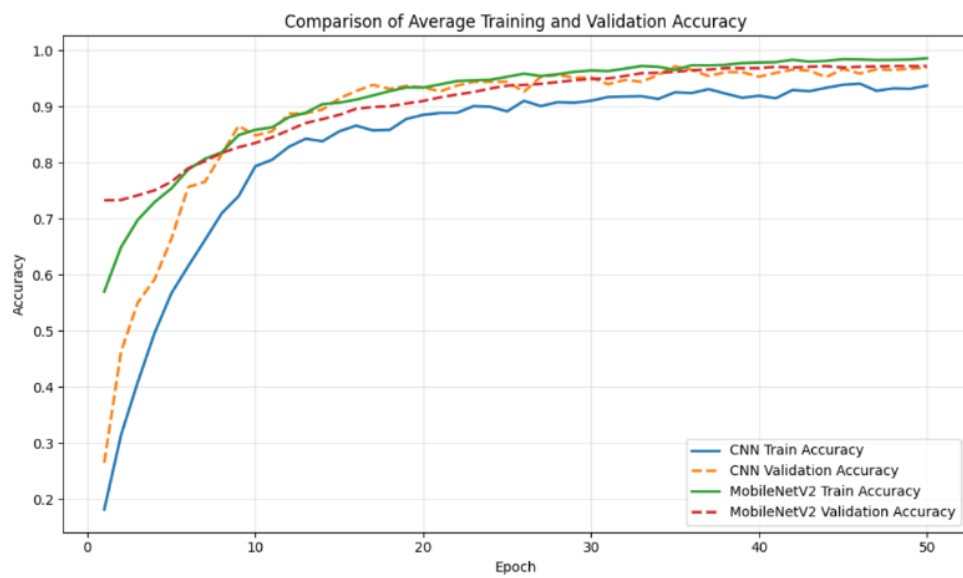
No	Nominal	Data Latih	Data Validasi	Data Uji	Total
(1)	(2)	(3)	(4)	(5)	(6)
1	Rp1.000	210	45	45	300
2	Rp2.000	210	45	45	300
3	Rp5.000	210	45	45	300

No	Nominal	Data Latih	Data Validasi	Data Uji	Total
(1)	(2)	(3)	(4)	(5)	(6)
4	Rp10.000	210	45	45	300
5	Rp20.000	210	45	45	300
6	Rp50.000	210	45	45	300
7	Rp100.000	210	45	45	300
8	Total	1.470	315	315	2.100

Berdasarkan Tabel 4, distribusi jumlah citra pada setiap subset tetap seimbang untuk seluruh kelas nominal. Kondisi tersebut bertujuan untuk mencegah model lebih banyak mempelajari kelas tertentu akibat jumlah data yang tidak seimbang, sehingga setiap kelas memiliki peluang yang sama dalam proses pelatihan. Dengan demikian, proses pelatihan, validasi, dan pengujian dapat dilakukan secara proporsional sehingga hasil evaluasi performa CNN Custom dan MobileNetV2 berbasis Transfer Learning dapat dibandingkan secara lebih objektif.

2. Hasil Pelatihan Model

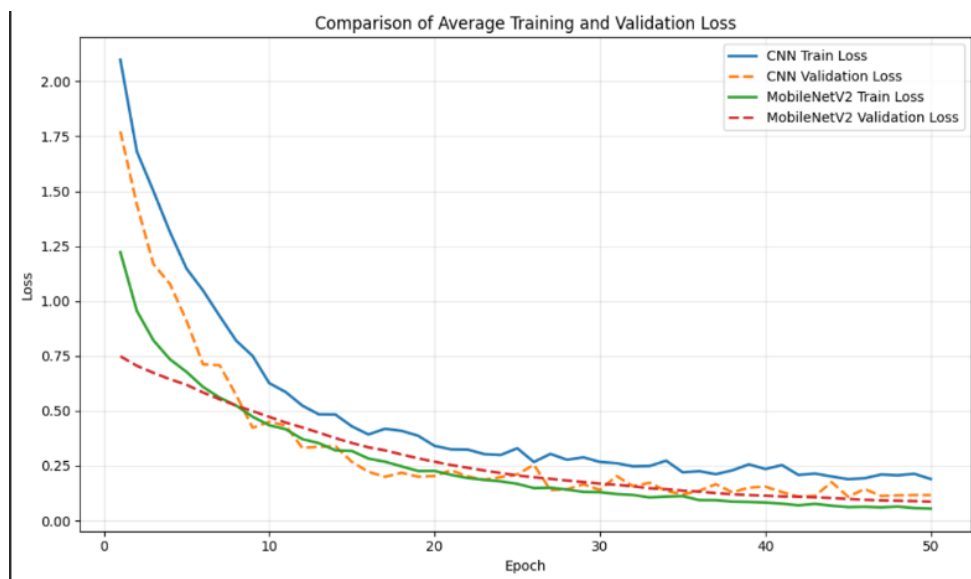
Tahap pelatihan dilakukan untuk mengevaluasi kemampuan model CNN Custom dan MobileNetV2 dalam mempelajari karakteristik citra uang kertas rupiah berdasarkan data latih serta kemampuan model dalam melakukan generalisasi terhadap data validasi. Proses pelatihan diamati melalui perubahan nilai *training accuracy*, *validation accuracy*, *training loss*, dan *validation loss* selama 50 *epoch*. Grafik yang disajikan merupakan rata-rata hasil dari lima kali percobaan sehingga dapat menggambarkan kecenderungan performa kedua model secara lebih representatif.



Gambar 10 Perbandingan Rata-rata *Training* dan *Validation Accuracy* CNN Custom dan MobileNetV2

Berdasarkan Gambar 10, kedua model menunjukkan peningkatan nilai *training accuracy* dan *validation accuracy* seiring bertambahnya jumlah *epoch*. MobileNetV2 mengalami peningkatan akurasi yang lebih cepat dibandingkan CNN Custom, kemudian mencapai kondisi konvergen lebih awal dengan kurva yang relatif stabil hingga akhir pelatihan. Sementara itu, CNN Custom juga menunjukkan peningkatan akurasi yang konsisten, namun membutuhkan jumlah *epoch* yang lebih banyak untuk mencapai kondisi yang stabil.

Pada akhir proses pelatihan, MobileNetV2 menghasilkan rata-rata *accuracy* sebesar 95,81%, lebih tinggi dibandingkan CNN Custom yang memperoleh rata-rata 94,35%. Perbedaan tersebut menunjukkan bahwa MobileNetV2 memiliki kemampuan yang lebih baik dalam mempelajari karakteristik citra uang kertas rupiah selama proses pelatihan. Selain itu, jarak antara kurva *training accuracy* dan *validation accuracy* pada kedua model relatif kecil sehingga menunjukkan bahwa proses pembelajaran berlangsung secara stabil tanpa indikasi *overfitting* yang signifikan.



Gambar 11 Perbandingan Rata-rata *Training* dan *Validation Loss* CNN Custom dan MobileNetV2

Berdasarkan Gambar 11, nilai *training loss* dan *validation loss* pada kedua model mengalami penurunan seiring bertambahnya jumlah *epoch*. Penurunan tersebut menunjukkan bahwa kedua model semakin mampu meminimalkan kesalahan prediksi selama proses pelatihan. Namun demikian, MobileNetV2 memperlihatkan penurunan *loss* yang lebih cepat dan mampu mempertahankan nilai *validation loss* yang lebih rendah dibandingkan CNN Custom hingga akhir pelatihan.

Kondisi tersebut menunjukkan bahwa proses optimasi pada MobileNetV2 berlangsung lebih efektif sehingga model mampu mencapai konvergensi dengan lebih baik. Hal ini juga didukung oleh rata-rata nilai *precision*, *recall*, dan *F1-score* yang masing-masing mencapai 95,98%, 95,81%, dan 95,78%, sedangkan CNN Custom memperoleh 94,89%, 94,35%, dan 94,41%. Hasil tersebut menunjukkan bahwa proses pembelajaran MobileNetV2 berlangsung lebih konsisten dibandingkan CNN Custom dalam mengenali karakteristik citra uang kertas rupiah.

Secara keseluruhan, hasil pelatihan menunjukkan bahwa kedua model mampu mempelajari karakteristik citra uang kertas rupiah dengan baik. Namun, MobileNetV2 memperlihatkan proses pembelajaran yang lebih cepat, lebih stabil, dan menghasilkan performa yang lebih baik dibandingkan CNN Custom. Hasil ini menjadi dasar untuk melakukan analisis lebih lanjut terhadap kemampuan kedua model pada tahap evaluasi.

Setelah proses pelatihan selesai, performa CNN Custom dan MobileNetV2 dievaluasi menggunakan data uji yang tidak terlibat selama proses pelatihan. Evaluasi dilakukan sebanyak lima kali percobaan (*run*) dengan konfigurasi parameter yang sama untuk mengetahui konsistensi performa masing-masing model. Hasil evaluasi setiap model disajikan dalam bentuk nilai *accuracy*, *precision*, *recall*, *F1-score*, *inference time*, dan ukuran model, kemudian dianalisis lebih lanjut menggunakan *confusion matrix*, *classification report*, serta *accuracy* pada setiap kelas. Hasil evaluasi CNN Custom disajikan pada Tabel 5, sedangkan hasil evaluasi MobileNetV2 disajikan pada Tabel 9. Selanjutnya, performa kedua model dibandingkan berdasarkan nilai rata-rata dan standar deviasi untuk memperoleh gambaran menyeluruh mengenai kemampuan masing-masing model dalam mengklasifikasikan citra uang kertas rupiah.

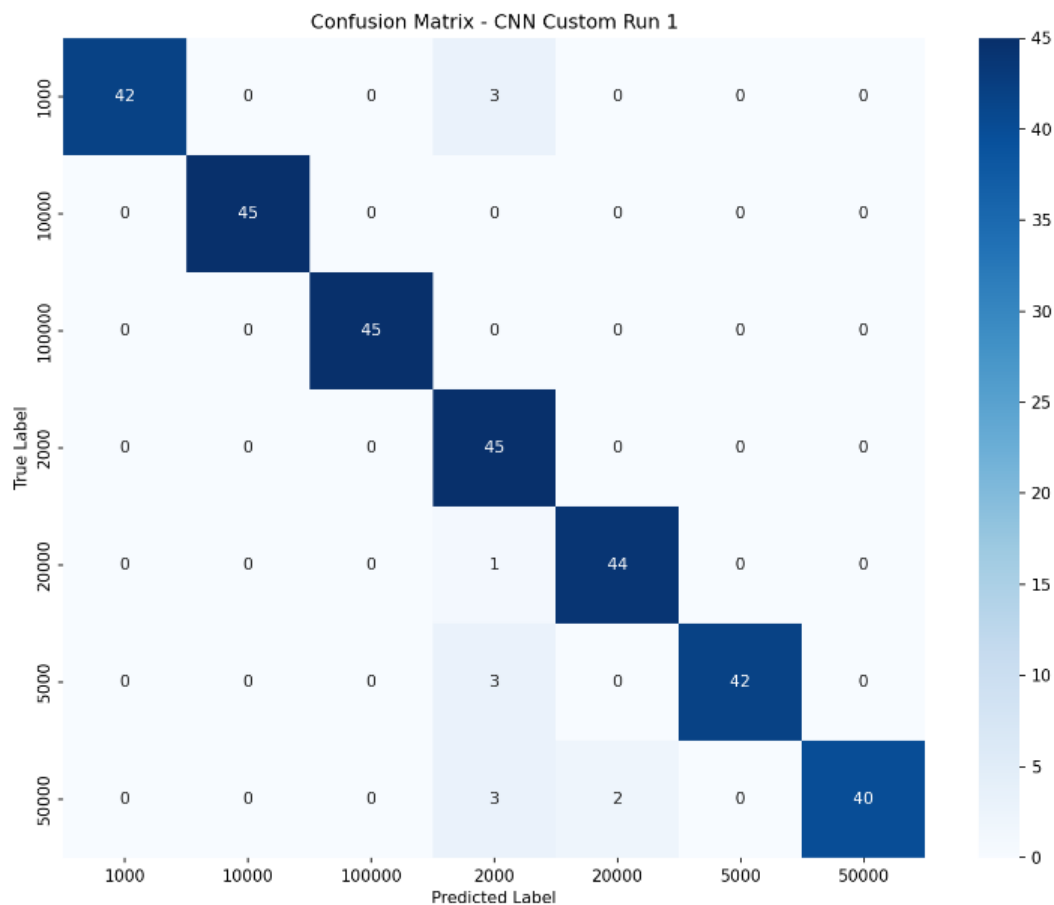
3. Hasil Pengujian Model

Tabel 5 Hasil Evaluasi CNN Custom pada Lima Percobaan

No	Percobaan	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	Inference Time (ms)	Ukuran Model (MB)
(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
1	Run 1	96,19	96,78	96,19	96,27	131,37	127,87
2	Run 2	94,92	95,21	94,92	94,91	130,40	127,87
3	Run 3	94,29	94,74	94,29	94,31	134,30	127,87
4	Run 4	94,60	94,76	94,60	94,58	127,70	127,87
5	Run 5	91,75	92,96	91,75	91,95	128,23	127,87

Berdasarkan Tabel 5, CNN Custom menunjukkan performa klasifikasi yang relatif konsisten pada lima kali percobaan. Nilai *accuracy* berada pada rentang 91,75% hingga 96,19%, sedangkan nilai *precision*, *recall*, dan *F1-score* juga menunjukkan hasil yang tinggi pada setiap *run*. Variasi nilai antarpercobaan menunjukkan adanya pengaruh inisialisasi bobot awal terhadap proses pelatihan, namun tidak menyebabkan penurunan performa yang signifikan. Selain itu, ukuran model tetap sebesar 127,87 MB pada seluruh percobaan, sedangkan *inference time* berada pada rentang 127,70–134,30 ms per citra.

Selanjutnya *Confusion matrix* hasil pengujian CNN Custom Run 1 ditunjukkan pada Gambar 12.



Gambar 12 *Confusion Matrix* CNN Custom Run 1

Berdasarkan Gambar 12, sebagian besar nilai berada pada diagonal utama yang menunjukkan bahwa mayoritas citra berhasil diprediksi sesuai dengan kelas sebenarnya. Meskipun demikian, masih terdapat beberapa kesalahan klasifikasi yang ditunjukkan oleh nilai di luar diagonal. Kelas Rp2.000 memiliki jumlah False Positive (FP) tertinggi, yaitu sebanyak 10 citra, yang berarti terdapat 10 citra dari kelas lain yang diprediksi sebagai Rp2.000. Kesalahan tersebut berasal dari kelas Rp1.000 sebanyak 3 citra, Rp5.000 sebanyak 3 citra, Rp20.000 sebanyak 1

citra, dan Rp50.000 sebanyak 3 citra. Hasil ini menunjukkan bahwa masih terdapat kemiripan karakteristik visual antara kelas Rp2.000 dengan beberapa kelas lainnya.

Classification report, *accuracy* per kelas, dan *confusion matrix* dari CNN Custom Run 1 disajikan pada Tabel 6, Tabel 7, dan Tabel 8.

Tabel 6 *Classification Report* CNN Custom Run 1

No	Nominal	Precision (%)	Recall (%)	F1-Score (%)	Support
(1)	(2)	(4)	(5)	(6)	(7)
1	Rp1.000	100,00	93,33	96,55	45
2	Rp2.000	81,82	100,00	90,00	45
3	Rp5.000	100,00	93,33	96,55	45
4	Rp10.000	100,00	100,00	100,00	45
5	Rp20.000	95,65	97,78	96,70	45
6	Rp50.000	100,00	88,89	94,12	45
7	Rp100.000	100,00	100,00	100,00	45

Berdasarkan Tabel 6, sebagian besar kelas memperoleh nilai precision, recall, dan F1-score yang tinggi, sehingga menunjukkan bahwa model mampu membedakan setiap kelas nominal dengan baik. Kelas Rp10.000 dan Rp100.000 memperoleh nilai 100,00% pada ketiga metrik tersebut. Sementara itu, kelas Rp2.000 memiliki nilai precision terendah, yaitu 81,82%, yang dipengaruhi oleh tingginya jumlah *False Positive* pada kelas tersebut.

Tabel 7 *Accuracy* per Kelas CNN Custom Run 1

No	Nominal	Accuracy (%)
(1)	(2)	(3)
1	Rp1.000	99,05
2	Rp2.000	96,83
3	Rp5.000	99,05
4	Rp10.000	100,00
5	Rp20.000	99,05
6	Rp50.000	98,41
7	Rp100.000	100,00

Berdasarkan Tabel 7, nilai accuracy pada seluruh kelas berada pada kisaran 96,83% hingga 100,00%. Hasil tersebut menunjukkan bahwa CNN Custom mampu mengklasifikasikan setiap nominal dengan tingkat ketepatan yang tinggi. Accuracy tertinggi diperoleh pada kelas Rp10.000 dan Rp100.000 dengan nilai

100,00%, sedangkan accuracy terendah diperoleh pada kelas Rp2.000 sebesar 96,83%. Untuk memberikan gambaran yang lebih rinci mengenai hasil klasifikasi pada setiap kelas, confusion matrix disajikan pada Tabel 8. Melalui confusion matrix dapat diketahui jumlah prediksi yang benar maupun kesalahan klasifikasi pada masing-masing nominal.

Tabel 8 *Confusion Matrix* per Kelas CNN Custom Run 1

No	Nominal	TP	FP	FN	TN
(1)	(2)	(3)	(4)	(5)	(6)
1	Rp1.000	42	0	3	270
2	Rp2.000	45	10	0	260
3	Rp5.000	42	0	3	270
4	Rp10.000	45	0	0	270
5	Rp20.000	44	2	1	268
6	Rp50.000	40	0	5	270
7	Rp100.000	45	0	0	270

Berdasarkan Tabel 8, kelas Rp10.000 dan Rp100.000 memperoleh nilai TP sebesar 45 dengan nilai FP dan FN sebesar 0, yang menunjukkan bahwa seluruh citra pada kedua kelas tersebut berhasil diklasifikasikan dengan benar. Sebaliknya, kelas Rp2.000 memiliki nilai FP tertinggi, yaitu 10, yang sejalan dengan hasil pada confusion matrix. Hal tersebut menunjukkan bahwa masih terdapat beberapa citra dari kelas lain yang diprediksi sebagai Rp2.000.

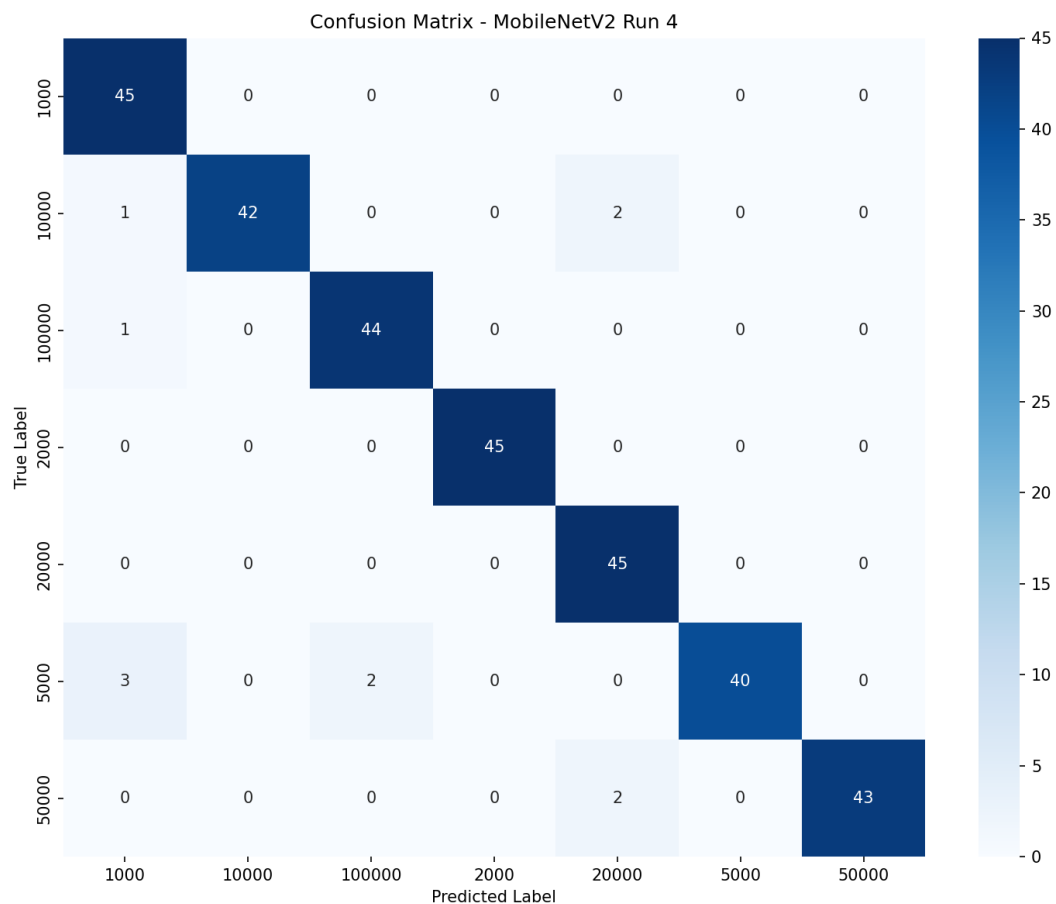
Secara keseluruhan, hasil pengujian menunjukkan bahwa CNN Custom mampu mengklasifikasikan citra uang kertas Rupiah TE 2022 dengan performa yang baik. Run 1 dipilih sebagai representasi performa terbaik karena memperoleh nilai accuracy tertinggi sebesar 96,19%. Hasil pada percobaan tersebut selanjutnya digunakan sebagai dasar perbandingan dengan model MobileNetV2 berbasis *Transfer Learning* pada subbab berikutnya.

Tabel 9 Hasil Evaluasi MobileNetV2 pada Lima Percobaan

No	Percobaan	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	Inference Time (ms)	Ukuran Model (MB)
(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
1	Run 1	96,19	96,23	96,19	96,14	98,82	25,31
2	Run 2	95,87	95,98	95,87	95,83	163,89	25,31
3	Run 3	95,87	96,10	95,87	95,85	146,08	25,31
4	Run 4	96,51	96,78	96,51	96,51	105,57	25,31
5	Run 5	94,60	94,82	94,60	94,56	185,97	25,31

Berdasarkan Tabel 9, MobileNetV2 menunjukkan performa klasifikasi yang konsisten pada lima kali percobaan. Nilai *accuracy* berada pada rentang 94,60% hingga 96,51%, sedangkan nilai *precision*, *recall*, dan *F1-score* juga tetap tinggi pada setiap *run*. Ukuran model tidak mengalami perubahan, yaitu 25,31 MB, sedangkan *inference time* berada pada rentang 98,82–185,97 ms per citra. Hasil tersebut menunjukkan bahwa MobileNetV2 mampu mempertahankan performa klasifikasi yang baik meskipun terdapat variasi waktu inferensi antarpercobaan.

Selanjutnya *Confusion matrix* hasil pengujian MobileNetV2 Run 4 ditunjukkan pada Gambar 18.



Gambar 13 *Confusion Matrix* MobileNetV2 Run 4

Berdasarkan Gambar 21, model berhasil mengklasifikasikan sebagian besar citra dengan benar, yang terlihat dari dominasi nilai pada diagonal utama matriks. Kelas Rp2.000 berhasil diklasifikasikan dengan sempurna karena seluruh citra dikenali dengan benar dan tidak ada citra dari kelas lain yang diprediksi sebagai Rp2.000. Sementara itu, kelas Rp1.000 dan Rp20.000 juga berhasil mengenali seluruh citra dengan benar, namun masih terdapat beberapa citra dari kelas lain yang diprediksi sebagai kedua kelas tersebut. Kelas Rp5.000 memiliki jumlah *False Negative* (FN) tertinggi yaitu sebesar 5 citra, yang berasal dari 3 citra yang

salah diprediksi sebagai Rp1.000 dan 2 citra yang salah diprediksi sebagai Rp100.000. Hasil tersebut menunjukkan bahwa model masih mengalami kesulitan dalam membedakan beberapa kelas nominal yang memiliki karakteristik visual yang mirip.

Classification report, *accuracy* per kelas, dan *confusion matrix* dari MobileNetV2 Run 4 disajikan pada Tabel 10, Tabel 11, dan Tabel 12.

Tabel 10 *Classification Report* MobileNetV2 Run 4

No	Nominal	Precision (%)	Recall (%)	F1-Score (%)	Support
(1)	(2)	(4)	(5)	(6)	(7)
1	Rp1.000	90,00	100,00	94,74	45
2	Rp2.000	100,00	100,00	100,00	45
3	Rp5.000	100,00	88,89	94,12	45
4	Rp10.000	100,00	93,33	96,55	45
5	Rp20.000	91,84	100,00	95,74	45
6	Rp50.000	100,00	95,56	97,73	45
7	Rp100.000	95,65	97,78	96,70	45

Berdasarkan Tabel 10, sebagian besar kelas memperoleh nilai precision, recall, dan F1-score yang tinggi, sehingga menunjukkan bahwa model mampu membedakan setiap kelas nominal dengan baik. Kelas Rp2.000 memperoleh nilai 100,00% pada ketiga metrik tersebut. Sementara itu, kelas Rp1.000 memiliki nilai precision terendah, yaitu 90,00%, yang disebabkan oleh adanya beberapa citra dari kelas lain yang diprediksi sebagai Rp1.000 sebagaimana terlihat pada confusion matrix.

Tabel 11 *Accuracy* per Kelas MobileNetV2 Run 4

No	Nominal	Accuracy (%)
(1)	(2)	(3)
1	Rp1.000	98,41
2	Rp2.000	100,00
3	Rp5.000	98,41
4	Rp10.000	99,05
5	Rp20.000	98,73
6	Rp50.000	99,37
7	Rp100.000	99,05

Berdasarkan Tabel 11, nilai accuracy pada seluruh kelas berada pada kisaran 98,41% hingga 100,00%. Hasil tersebut menunjukkan bahwa MobileNetV2

mampu mengklasifikasikan setiap nominal dengan tingkat ketepatan yang tinggi. Accuracy tertinggi diperoleh pada kelas Rp2.000 sebesar 100,00%, sedangkan accuracy terendah diperoleh pada kelas Rp1.000 dan Rp5.000 sebesar 98,41%. Untuk memberikan gambaran yang lebih rinci mengenai hasil klasifikasi pada setiap kelas, confusion matrix disajikan pada Tabel 12. Melalui confusion matrix dapat diketahui jumlah prediksi yang benar maupun kesalahan klasifikasi pada masing-masing nominal.

Tabel 12 *Confusion Matrix* per Kelas MobileNetV2 Run 4

No	Nominal	TP	FP	FN	TN
(1)	(2)	(3)	(4)	(5)	(6)
1	Rp1.000	45	5	0	265
2	Rp2.000	45	0	0	270
3	Rp5.000	40	0	5	270
4	Rp10.000	42	0	3	270
5	Rp20.000	45	4	0	266
6	Rp50.000	43	0	2	270
7	Rp100.000	44	2	1	268

Berdasarkan Tabel 12, model masih mengalami kesalahan klasifikasi pada beberapa kelas nominal. Kelas Rp5.000 memiliki nilai *False Negative* tertinggi sehingga beberapa citra Rp5.000 diprediksi sebagai kelas lain. Sementara itu, kelas Rp1.000 memiliki nilai *False Positive* tertinggi yang menunjukkan bahwa masih terdapat beberapa citra dari kelas lain yang diprediksi sebagai Rp1.000. Kondisi tersebut mengindikasikan bahwa beberapa kelas nominal masih memiliki karakteristik visual yang cukup mirip sehingga menyebabkan model melakukan kesalahan klasifikasi pada sejumlah citra.

Berdasarkan seluruh hasil pengujian, MobileNetV2 mampu menghasilkan performa klasifikasi yang baik pada dataset citra uang kertas Rupiah TE 2022. Dari lima kali percobaan, Run 4 memberikan hasil terbaik dengan nilai *accuracy* sebesar 96,51%, sehingga hasil pada percobaan tersebut digunakan sebagai representasi MobileNetV2 untuk dibandingkan dengan CNN *Custom* pada subbab berikutnya.

4. Perbandingan Hasil Pengujian

Hasil evaluasi dari lima kali percobaan pada masing-masing model direkapitulasi untuk memperoleh gambaran perbandingan performa secara keseluruhan. Tabel 13 menyajikan nilai rata-rata dan standar deviasi dari seluruh metrik evaluasi kedua model.

Tabel 13 Rekapitulasi Perbandingan CNN Custom dan MobileNetV2

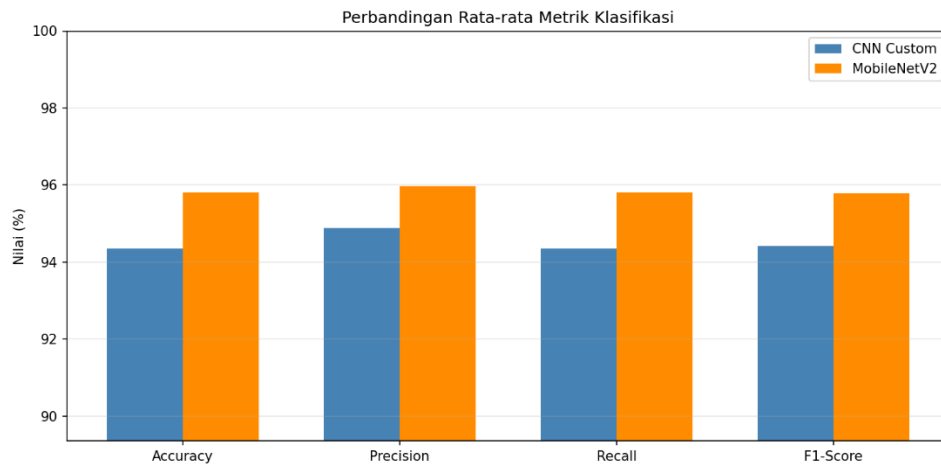
No	Metrik	CNN Custom	MobileNetV2	Selisih	Model Terbaik
(1)	(2)	(3)	(4)	(5)	(6)
1	Accuracy (%)	94,35 $\pm$ 1,62	95,81 $\pm$ 0,72	1,46	MobileNetV2
2	Precision (%)	94,89 $\pm$ 1,36	95,98 $\pm$ 0,72	1,09	MobileNetV2
3	Recall (%)	94,35 $\pm$ 1,62	95,81 $\pm$ 0,72	1,46	MobileNetV2
4	F1-Score (%)	94,41 $\pm$ 1,56	95,78 $\pm$ 0,73	1,37	MobileNetV2
5	Inference Time (ms)	130,40 $\pm$ 2,65	139,96 $\pm$ 37,57	9,56	CNN Custom
6	Ukuran Model (MB)	127,87	25,31	102,56	MobileNetV2

Tabel 13 menunjukkan bahwa MobileNetV2 menghasilkan performa klasifikasi yang lebih baik dibandingkan CNN Custom. Hal tersebut ditunjukkan oleh nilai rata-rata *accuracy*, *precision*, *recall*, dan *F1-score* yang secara konsisten lebih tinggi pada MobileNetV2. Nilai *accuracy* rata-rata MobileNetV2 sebesar 95,81% lebih tinggi dibandingkan CNN Custom yang memperoleh 94,35%, dengan selisih sebesar 1,46%. Nilai rata-rata *precision*, *recall*, dan *F1-score* MobileNetV2 juga lebih tinggi dibandingkan CNN Custom, dengan selisih masing-masing sebesar 1,09%, 1,46%, dan 1,37%.

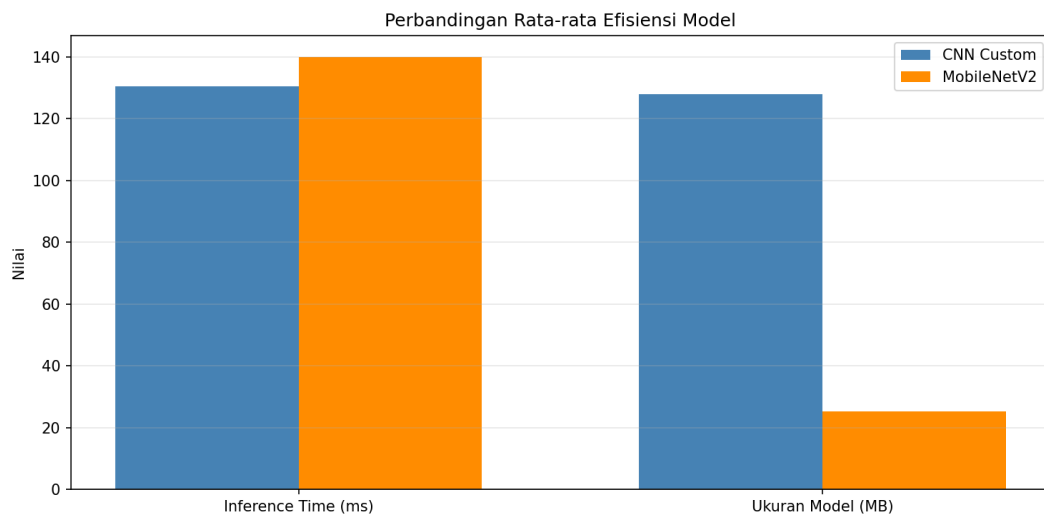
Pada aspek efisiensi, CNN Custom memperoleh waktu inferensi (*inference time*) rata-rata yang lebih singkat, yaitu 130,40 ms dibandingkan MobileNetV2 yang mencapai 139,96 ms, dengan selisih sebesar 9,56 ms. Hal ini menunjukkan bahwa CNN Custom mampu melakukan proses prediksi lebih cepat pada setiap citra. Di sisi lain, MobileNetV2 memiliki ukuran model yang jauh lebih kecil, yaitu 25,31 MB dibandingkan CNN Custom sebesar 127,87 MB, dengan selisih sebesar 102,56 MB. Ukuran model yang lebih kecil menjadi keunggulan karena membutuhkan kapasitas penyimpanan dan memori yang lebih rendah ketika model diimplementasikan.

Dari sisi konsistensi, MobileNetV2 memperoleh nilai standar deviasi *accuracy* sebesar 0,72%, lebih kecil dibandingkan CNN Custom yang mencapai 1,62%. Hal tersebut menunjukkan bahwa performa MobileNetV2 lebih stabil pada lima kali percobaan pelatihan. Konsistensi tersebut mengindikasikan bahwa proses *fine-tuning* pada model pralatih mampu menghasilkan variasi performa yang lebih rendah dibandingkan CNN Custom yang dilatih dari bobot awal secara acak.

Secara keseluruhan, MobileNetV2 dipilih sebagai model yang lebih sesuai untuk klasifikasi citra uang kertas Rupiah TE 2022 karena memberikan keseimbangan yang lebih baik antara performa klasifikasi, efisiensi ukuran model, dan konsistensi hasil. Meskipun CNN Custom memiliki waktu inferensi yang lebih singkat, selisih waktu tersebut relatif kecil dibandingkan keunggulan MobileNetV2 pada metrik klasifikasi, ukuran model, dan kestabilan performa. Perbandingan rata-rata metrik klasifikasi dan efisiensi kedua model secara visual disajikan pada Gambar 19 dan Gambar 20.



Gambar 14 Grafik Perbandingan Rata-rata Metrik Klasifikasi



Gambar 15 Grafik Perbandingan Rata-rata Efisiensi Model

Secara keseluruhan, hasil pengujian menunjukkan bahwa MobileNetV2 mampu menghasilkan performa klasifikasi yang tinggi pada dataset penelitian. Hasil tersebut kemudian dibandingkan dengan CNN Custom untuk mengetahui model yang memberikan performa terbaik pada klasifikasi citra uang kertas Rupiah Tahun Emisi 2022.

B. PEMBAHASAN

1. Perbandingan Performa Klasifikasi

Berdasarkan hasil pengujian, MobileNetV2 memperoleh nilai *accuracy* rata-rata sebesar 95,81%, lebih tinggi dibandingkan CNN Custom yang memperoleh 94,35%. Keunggulan yang sama juga terlihat pada metrik *precision* (95,98% berbanding 94,89%), *recall* (95,81% berbanding 94,35%), dan *F1-score* (95,78% berbanding 94,41%). Perbedaan tersebut menunjukkan bahwa MobileNetV2 mampu menghasilkan performa klasifikasi yang lebih baik secara konsisten dibandingkan CNN Custom pada seluruh metrik evaluasi.

Perbedaan performa tersebut menunjukkan bahwa kedua model memiliki kemampuan yang berbeda dalam mengekstraksi dan merepresentasikan fitur visual pada citra uang kertas Rupiah. Perbedaan kemampuan tersebut dapat dijelaskan melalui karakteristik arsitektur masing-masing model.

Keunggulan MobileNetV2 pada seluruh metrik klasifikasi tidak terlepas dari pendekatan *transfer learning* yang digunakan. Model ini telah dilatih sebelumnya menggunakan dataset ImageNet sehingga bobot awalnya telah memiliki representasi fitur visual yang kaya. Ketika diterapkan pada dataset uang kertas Rupiah, model hanya perlu menyesuaikan bobot tersebut terhadap karakteristik dataset penelitian, sehingga proses pembelajaran menjadi lebih efektif dibandingkan CNN Custom yang mempelajari seluruh representasi fitur dari awal [42].

CNN Custom dilatih sepenuhnya menggunakan dataset penelitian ini sehingga seluruh representasi fitur dipelajari dari awal. Kondisi tersebut menyebabkan proses pembelajaran menjadi lebih menantang dibandingkan pendekatan *transfer learning*, terutama karena model harus membangun representasi fitur tanpa memanfaatkan pengetahuan awal dari model pralatih [35]. Meskipun demikian, CNN Custom tetap menghasilkan *accuracy* rata-rata sebesar 94,35%, sehingga masih mampu memberikan performa klasifikasi yang kompetitif.

Hasil penelitian ini sejalan dengan beberapa penelitian sebelumnya. Velarati *et al.* [11] melaporkan bahwa MobileNetV2 mencapai akurasi 99% pada klasifikasi citra rempah-rempah, lebih tinggi dibandingkan CNN Custom yang memperoleh 93%. Murinto *et al.* [43] juga menemukan bahwa MobileNetV2 memperoleh akurasi tertinggi sebesar 96% dibandingkan CNN Custom sebesar 95% pada klasifikasi biji kopi. Temuan tersebut memperkuat hasil penelitian ini bahwa pendekatan MobileNetV2 berbasis *transfer learning* mampu memberikan performa klasifikasi yang lebih baik dibandingkan CNN Custom pada klasifikasi citra uang kertas Rupiah.

2. Perbandingan Efisiensi Model

Pada aspek efisiensi, hasil pengujian menunjukkan bahwa CNN *Custom* memperoleh *inference time* rata-rata sebesar 130,40 ms per citra, lebih rendah dibandingkan MobileNetV2 yang mencapai 139,96 ms. Temuan ini menarik karena ukuran model MobileNetV2 hanya 25,31 MB, jauh lebih kecil dibandingkan CNN *Custom* yang mencapai 127,87 MB, namun *inference time*-nya justru lebih tinggi.

Perbedaan *inference time* ini dapat dijelaskan oleh karakteristik arsitektur kedua model. MobileNetV2 menggunakan *depthwise separable convolution*, *inverted residual*, dan *linear bottleneck* untuk mengurangi jumlah parameter dan ukuran model [27]. Meskipun lebih efisien dari sisi parameter, rangkaian operasi tersebut tidak selalu menghasilkan waktu inferensi yang lebih cepat pada setiap perangkat keras maupun framework yang digunakan.

Pada penelitian ini, CNN *Custom* menunjukkan waktu inferensi yang lebih rendah dibandingkan MobileNetV2. Perbedaan tersebut dapat dipengaruhi oleh karakteristik implementasi model serta lingkungan komputasi selama proses pengujian. Seluruh eksperimen dilakukan menggunakan *Google Colaboratory* yang memiliki alokasi sumber daya komputasi secara dinamis sehingga waktu inferensi antarpercobaan dapat mengalami variasi.

Dari sisi ukuran model, MobileNetV2 menunjukkan keunggulan yang sangat signifikan dengan selisih sebesar 102,56 MB. Ukuran model yang jauh lebih kecil merupakan konsekuensi dari desain arsitektur MobileNetV2 yang memanfaatkan *depthwise separable convolution*, *inverted residual*, dan *linear bottleneck* untuk mengurangi jumlah parameter tanpa mengurangi kemampuan representasi fitur secara signifikan [27]. Ukuran model yang lebih kecil menjadi keunggulan penting dalam implementasi praktis karena membutuhkan kapasitas penyimpanan dan memori yang lebih sedikit, terutama ketika model akan diterapkan pada perangkat mobile atau sistem embedded.

Secara keseluruhan, kedua model memiliki keunggulan pada aspek efisiensi yang berbeda. CNN *Custom* menunjukkan waktu inferensi yang lebih cepat pada lingkungan pengujian penelitian ini, sedangkan MobileNetV2 memiliki ukuran model yang jauh lebih kecil. Oleh karena itu, pemilihan model perlu disesuaikan dengan kebutuhan implementasi. Apabila efisiensi penyimpanan dan portabilitas model menjadi prioritas, MobileNetV2 merupakan pilihan yang lebih sesuai. Sebaliknya, apabila waktu inferensi lebih diprioritaskan pada kondisi pengujian yang serupa dengan penelitian ini, CNN *Custom* dapat menjadi alternatif yang baik.

3. Perbandingan Konsistensi Model

Keandalan suatu model tidak hanya ditentukan oleh performa rata-rata yang dihasilkan, tetapi juga oleh konsistensi performanya pada setiap percobaan pelatihan. Berdasarkan hasil rekapitulasi pada Tabel 13, MobileNetV2 memperoleh nilai standar deviasi *accuracy* sebesar 0,72%, lebih kecil dibandingkan CNN Custom yang mencapai 1,62%. Pola yang sama juga terlihat pada *F1-score*, di mana standar deviasi MobileNetV2 sebesar 0,73% lebih rendah dibandingkan CNN Custom sebesar 1,56%. Hasil tersebut menunjukkan bahwa MobileNetV2 menghasilkan performa yang lebih konsisten pada setiap percobaan pelatihan.

Perbedaan konsistensi tersebut berkaitan dengan titik awal proses pelatihan masing-masing model. MobileNetV2 memulai pelatihan menggunakan bobot *pretrained* hasil pelatihan pada dataset *ImageNet*, sehingga kondisi awal modelnya lebih terarah dan proses *fine-tuning* hanya perlu menyesuaikan bobot terhadap karakteristik dataset uang kertas Rupiah. Sebaliknya, CNN Custom memulai pelatihan dari bobot yang diinisialisasi secara acak pada setiap percobaan, sehingga proses optimasinya berpotensi mencapai solusi yang berbeda pada setiap percobaan [35]. Kondisi inilah yang menyebabkan CNN Custom menunjukkan variasi performa yang lebih besar dibandingkan MobileNetV2 pada penelitian ini.

Konsistensi yang lebih tinggi pada MobileNetV2 menjadi nilai tambah dalam implementasi praktis karena menunjukkan bahwa model mampu menghasilkan performa yang relatif stabil ketika proses pelatihan diulang menggunakan konfigurasi parameter yang sama. Dengan standar deviasi *accuracy* sebesar 0,72%, MobileNetV2 tidak hanya memberikan performa rata-rata yang lebih tinggi, tetapi juga menunjukkan tingkat keandalan yang lebih baik dibandingkan CNN Custom.

4. Analisis Kesalahan Klasifikasi

Analisis *confusion matrix* pada percobaan terbaik masing-masing model menunjukkan bahwa masih terdapat kesalahan klasifikasi pada kelas-kelas tertentu, meskipun kedua model menghasilkan performa klasifikasi yang baik secara keseluruhan. Pemahaman terhadap pola kesalahan tersebut penting untuk mengetahui keterbatasan model dalam membedakan karakteristik visual antar kelas nominal.

Berdasarkan *confusion matrix* pada kedua model, masih ditemukan beberapa kesalahan klasifikasi meskipun jumlahnya relatif sedikit. CNN Custom menunjukkan jumlah False Positive tertinggi pada kelas Rp2.000 sebanyak 10 citra, sedangkan MobileNetV2 menunjukkan jumlah False Negative tertinggi pada kelas Rp5.000 sebanyak 5 citra. Perbedaan pola kesalahan tersebut

menunjukkan bahwa kedua arsitektur mempelajari representasi fitur yang berbeda sehingga menghasilkan karakteristik kesalahan klasifikasi yang tidak sama.

Pada CNN *Custom* Run 1, kelas Rp2.000 menjadi kelas dengan jumlah *False Positive* (FP) tertinggi, yaitu sebanyak 10 citra. Artinya, terdapat 10 citra dari kelas lain yang salah diprediksi sebagai Rp2.000, yang berasal dari kelas Rp1.000 sebanyak 3 citra, Rp5.000 sebanyak 3 citra, Rp20.000 sebanyak 1 citra, dan Rp50.000 sebanyak 3 citra. Kondisi tersebut menyebabkan nilai precision kelas Rp2.000 menjadi yang terendah, yaitu 81,82%. Hasil ini menunjukkan bahwa model masih mengalami kesulitan dalam membedakan karakteristik visual kelas Rp2.000 dengan beberapa kelas nominal lainnya.

Pada MobileNetV2 Run 4, pola kesalahan yang ditemukan berbeda dibandingkan CNN *Custom*. Kelas Rp5.000 memiliki jumlah *False Negative* (FN) tertinggi, yaitu sebanyak 5 citra, yang terdiri atas 3 citra salah diprediksi sebagai Rp1.000 dan 2 citra salah diprediksi sebagai Rp100.000. Selain itu, kelas Rp1.000 memiliki jumlah *False Positive* tertinggi, yaitu sebanyak 5 citra yang berasal dari kelas Rp5.000, Rp10.000, dan Rp100.000. Meskipun masih terdapat beberapa kesalahan klasifikasi, jumlah kesalahan tersebut relatif sedikit sehingga tidak memberikan pengaruh yang signifikan terhadap performa model secara keseluruhan.

Kesalahan klasifikasi dapat dipengaruhi oleh berbagai faktor, seperti kemiripan karakteristik visual antar nominal, variasi pencahayaan, sudut pengambilan citra, maupun kondisi fisik uang kertas pada dataset. Frekuensi kesalahan yang relatif rendah menunjukkan bahwa CNN *Custom* maupun MobileNetV2 telah mampu mempelajari karakteristik visual uang kertas Rupiah dengan baik sehingga menghasilkan performa klasifikasi yang tinggi secara keseluruhan.

BAB V KESIMPULAN

A. Kesimpulan

Berdasarkan hasil pengujian dan analisis yang telah dilakukan menggunakan metrik accuracy, precision, recall, F1-score, inference time, ukuran model, serta confusion matrix, diperoleh hasil bahwa MobileNetV2 memperoleh nilai rata-rata yang lebih tinggi pada seluruh metrik klasifikasi dibandingkan CNN Custom. MobileNetV2 mencapai *accuracy* rata-rata 95,81%, *precision* 95,98%, *recall* 95,81%, dan *F1-score* 95,78%, sedangkan CNN Custom memperoleh *accuracy* rata-rata 94,35%, *precision* 94,89%, *recall* 94,35%, dan *F1-score* 94,41%. Hasil tersebut menunjukkan bahwa pendekatan MobileNetV2 berbasis *Transfer Learning* lebih efektif dalam mengklasifikasikan citra uang kertas Rupiah TE 2022 dibandingkan CNN Custom yang dilatih dari awal.

Dari aspek efisiensi model, CNN Custom memperoleh waktu inferensi rata-rata yang lebih singkat, yaitu 130,40 ms per citra, dibandingkan MobileNetV2 yang mencapai 139,96 ms. Namun demikian, MobileNetV2 memiliki ukuran model yang jauh lebih kecil, yaitu 25,31 MB dibandingkan CNN Custom sebesar 127,87 MB, serta menunjukkan konsistensi performa yang lebih baik dengan standar deviasi *accuracy* sebesar 0,72% dibandingkan CNN Custom sebesar 1,62%. Meskipun waktu inferensi MobileNetV2 sedikit lebih lama, selisih tersebut relatif kecil dibandingkan keunggulannya pada performa klasifikasi, ukuran model, dan konsistensi hasil. Oleh karena itu, MobileNetV2 direkomendasikan sebagai model yang lebih sesuai untuk klasifikasi citra uang kertas Rupiah TE 2022 karena mampu memberikan keseimbangan yang lebih baik antara akurasi, efisiensi penyimpanan, dan keandalan model. Sementara itu, CNN Custom tetap menjadi alternatif yang baik apabila waktu inferensi menjadi pertimbangan utama dalam implementasi.

B. Saran/Rekomendasi

Berdasarkan hasil penelitian yang telah dilakukan, terdapat beberapa saran yang dapat diberikan untuk pengembangan penelitian selanjutnya.

1. Penelitian selanjutnya dapat menambahkan variasi kondisi citra yang lebih beragam, seperti uang kertas dalam kondisi terlipat, kusut, sobek ringan, atau sebagian tertutup, sehingga ketahanan model terhadap kondisi nyata di lapangan dapat dievaluasi secara lebih komprehensif.
2. Pengujian model pada perangkat mobile atau sistem *embedded* perlu dilakukan untuk mengevaluasi performa kedua model pada kondisi implementasi yang sesungguhnya, mengingat MobileNetV2 dirancang untuk perangkat dengan sumber daya komputasi yang terbatas.

3. Penelitian selanjutnya dapat mengeksplorasi penggunaan arsitektur *deep learning* lainnya sebagai pembanding sehingga diperoleh gambaran yang lebih komprehensif mengenai performa berbagai model dalam klasifikasi citra uang kertas Rupiah.
4. Mengintegrasikan metode Explainable AI (misalnya Grad-CAM) untuk menganalisis fitur visual yang paling berpengaruh terhadap keputusan klasifikasi model.

DAFTAR PUSTAKA

- [1] “Uang Beredar pada Desember 2025 Tumbuh Lebih Tinggi.” Accessed: Feb. 14, 2026. [Online]. Available: https://www.bi.go.id/id/publikasi/ruang-media/news-release/Pages/sp_281726.aspx
- [2] “Bank Indonesia dan Pemerintah Meluncurkan Uang Rupiah Kertas Tahun Emisi 2022.” Accessed: Feb. 14, 2026. [Online]. Available: https://www.bi.go.id/id/publikasi/ruang-media/news-release/Pages/sp_2421922.aspx
- [3] A. Mjahad, A. Polo-Aguado, L. Llorens-Serrano, and A. Rosado-Muñoz, “Optimizing Image Feature Extraction with Convolutional Neural Networks for Chicken Meat Detection Applications,” *Applied Sciences*, vol. 15, no. 2, p. 733, Jan. 2025, doi: 10.3390/app15020733.
- [4] X. Zhao, L. Wang, Y. Zhang, X. Han, M. Deveci, and M. Parmar, “A review of convolutional neural networks in computer vision,” *Artif. Intell. Rev.*, vol. 57, no. 4, Apr. 2024, doi: 10.1007/s10462-024-10721-6.
- [5] F. Sahputra, Z. Sitorus, M. Iqbal, L. Marlina, and D. Nasution, “Performance Analysis of CNN (Convolutional Neural Network) in Nominal Classification of Rupiah Emissions 2022,” *The IJICS (International Journal of Informatics and Computer Science)*, vol. 9, no. 2, pp. 81–89, Jul. 2025, doi: 10.30865/ijics.v9i2.8903.
- [6] A. T. Akbar, S. Saifullah, and H. Prapcoyo, “Klasifikasi Ekspresi Wajah Menggunakan Covolutional Neural Network,” *Jurnal Teknologi Informasi dan Ilmu Komputer*, vol. 11, no. 6, pp. 1399–1412, Dec. 2024, doi: 10.25126/jtiik.2024118888.
- [7] N. Nurtiwi, R. Ruliana, and Z. Rais, “Convolutional Neural Network (CNN) Method for Classification of Images by Age,” *JINAV: Journal of Information and Visualization*, vol. 3, no. 2, pp. 126–130, Dec. 2022, doi: 10.35877/454RI.jinav1481.
- [8] A. Syakuroh, F. Monado, M. Ariani, H. Hadi, E. Koriyanti, and E. Erni, “Analisis Akurasi Model Mobilenetv2 dalam Klasifikasi Citra X-Ray untuk Deteksi Kondisi Paru-Paru,” *JOURNAL ONLINE OF PHYSICS*, vol. 10, no. 3, pp. 67–74, Jul. 2025, doi: 10.22437/jop.v10i3.44453.
- [9] A. C. Hartono and A. R. Muslikh, “Penerapan Transfer Learning MobileNetV2 Pada Klasifikasi Citra Jenis Buah-Buahan Applying MobileNetV2 Transfer Learning for Image Classification of Fruit Types,” *Journal of Information System and Application Development*, vol. 3, no. 2, pp. 103–111, 2025, doi: 10.26905/jisad.v3i2.16187.

- [10] M. G. Somoal and A. R. Dzikrillah, “Komparasi MobileNETV2 dengan Kustomisasi Transfer Learning dan Hyperparameter untuk Identifikasi Tumor Otak,” *Jurnal Teknologi Informasi dan Ilmu Komputer*, vol. 12, no. 1, pp. 229–240, Feb. 2025, doi: 10.25126/jtiik.2025129582.
- [11] K. Velarati, C. A. Sari, and E. H. Rachmawanto, “A Comparison of Convolutional Neural Network (CNN) and Transfer Learning MobileNetV2 Performance on Spices Images Classification,” *Journal of Applied Informatics and Computing (JAIC)*, vol. 8, no. 2, pp. 413–420, Nov. 2024, doi: <https://doi.org/10.30871/jaic.v8i2.8622>.
- [12] M. Pradnyatama, C. A. Sari, E. H. Rachmawanto, and H. M. M. Islam, “A Comparative Analysis of Convolutional Neural Network (CNN): MobileNetV2 and Xception for Butterfly Species Classification,” *Jurnal Masyarakat Informatika*, vol. 16, no. 1, pp. 69–90, May 2025, doi: 10.14710/jmasif.16.1.72957.
- [13] N. H. Muttaqin and A. M. Widodo, “Evaluation of Transfer Learning-Based Convolutional Neural Networks (InceptionV3 and MobileNetV2) for Facial Skin-Type Classification,” *Jurnal Ilmu Komputer dan Informatika*, vol. 5, no. 1, pp. 11–32, Aug. 2025, doi: 10.54082/jiki.264.
- [14] P. A. Suri, M. A. Setiono, A. Andrew, and M. Fajar, “Comparative Study of CNN-based Deep Learning Models for Animal, Digit, and Flower Image Classification,” *Engineering, Mathematics and Computer Science Journal (EMACS)*, vol. 7, no. 3, pp. 321–326, Sep. 2025, doi: 10.21512/emacsjournal.v7i3.14317.
- [15] H. Chen, G. Zhou, W. He, X. Duan, and H. Jiang, “Classification And Identification Of Agricultural Products Based On Improved Mobilenetv2,” *Sci. Rep.*, vol. 14, no. 1, p. 3454, Dec. 2024, doi: 10.1038/s41598-024-53349-w.
- [16] M. H. R. Pratama, M. Akrom, A. P. Santosa, M. R. Rosyid, and L. Mawaddah, “Klasifikasi Otomatis Korosi Menggunakan Convolutional Neural Network dan Transfer Learning dengan Model MobileNetV2,” *Jurnal Algoritma*, vol. 22, no. 1, pp. 138–148, May 2025, doi: 10.33364/algoritma/v.22-1.2182.
- [17] R. Dijaya, *Buku Ajar Pengolahan Citra Digital*. Sidoarjo: Umsida Press, 2023. doi: 10.21070/2023/978-623-464-075-5.
- [18] R. M. Badri and S. Aningsih, *Buku Ajar Teknologi Pengolahan Citra Digital*. Lampung Selatan: ITERA Press, 2025.
- [19] R. Szeliski, *Computer Vision*. Cham: Springer International Publishing, 2022. doi: 10.1007/978-3-030-34372-9.

- [20] S. F. Ahmed *et al.*, “Deep Learning Modelling Techniques: Current Progress, Applications, Advantages, and Challenges,” *Artif. Intell. Rev.*, vol. 56, no. 11, pp. 13521–13617, Nov. 2023, doi: 10.1007/s10462-023-10466-8.
- [21] M. M. Taye, “Theoretical Understanding of Convolutional Neural Network: Concepts, Architectures, Applications, Future Directions,” *Computation*, vol. 11, no. 3, p. 52, Mar. 2023, doi: 10.3390/computation11030052.
- [22] X. Zhao, L. Wang, Y. Zhang, X. Han, M. Deveci, and M. Parmar, “A Review of Convolutional Neural Networks in Computer Vision,” *Artif. Intell. Rev.*, vol. 57, no. 4, p. 99, Mar. 2024, doi: 10.1007/s10462-024-10721-6.
- [23] P. Purwono, A. Ma’arif, W. Rahmانيar, H. I. K. Fathurrahman, A. Z. K. Frisky, and Q. M. ul Haq, “Understanding of Convolutional Neural Network (CNN): A Review,” *International Journal of Robotics and Control Systems*, vol. 2, no. 4, pp. 739–748, Jan. 2023, doi: 10.31763/ijrcs.v2i4.888.
- [24] T. S. Alam, C. B. Jowthi, and A. Pathak, “Comparing Pre-Trained Models for Efficient Leaf Disease Detection: a Study on Custom CNN,” *Journal of Electrical Systems and Information Technology*, vol. 11, no. 1, p. 12, Feb. 2024, doi: 10.1186/s43067-024-00137-1.
- [25] A. Hosna, E. Merry, J. Gyalmo, Z. Alom, Z. Aung, and M. A. Azim, “Transfer learning: a friendly introduction,” *J. Big Data*, vol. 9, no. 1, p. 102, Oct. 2022, doi: 10.1186/s40537-022-00652-w.
- [26] J. Gupta, S. Pathak, and G. Kumar, “Deep Learning (CNN) and Transfer Learning: A Review,” *J. Phys. Conf. Ser.*, vol. 2273, no. 1, p. 012029, May 2022, doi: 10.1088/1742-6596/2273/1/012029.
- [27] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “MobileNetV2: Inverted Residuals and Linear Bottlenecks,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE, Jun. 2018, pp. 4510–4520. doi: 10.1109/CVPR.2018.00474.
- [28] S. Guo, “The Model Predicts the Category of the Input Image,” *Applied and Computational Engineering*, vol. 190, no. 1, pp. 85–91, Nov. 2025, doi: 10.54254/2755-2721/2026.TJ29145.
- [29] H. E. Kim, A. Cosa-Linan, N. Santhanam, M. Jannesari, M. E. Maros, and T. Ganslandt, “Transfer Learning for Medical Image Classification: a Literature Review,” *BMC Med. Imaging*, vol. 22, no. 1, p. 69, Apr. 2022, doi: 10.1186/s12880-022-00793-7.
- [30] O. Rainio, J. Teuho, and R. Klén, “Evaluation metrics and statistical tests for machine learning,” *Sci. Rep.*, vol. 14, no. 1, p. 6086, Mar. 2024, doi: 10.1038/s41598-024-56706-x.
- [31] R. Roslidar, M. R. Syahputra, R. Muharar, and F. Arnia, “Adaptasi Model CNN Terlatih pada Aplikasi Bergerak untuk Klasifikasi Citra Termal

- Payudara,” *Jurnal Rekayasa ElektriKa*, vol. 18, no. 3, pp. 193–200, Sep. 2022, doi: 10.17529/jre.v18i3.8754.
- [32] “The Python Tutorial — Python 3.14.3 documentation.” Accessed: Apr. 01, 2026. [Online]. Available: <https://docs.python.org/3/tutorial/>
- [33] H. Snyder, “Literature Review as a Research Methodology: An Overview And Guidelines,” *J. Bus. Res.*, vol. 104, pp. 333–339, Nov. 2019, doi: 10.1016/j.jbusres.2019.07.039.
- [34] C. Shorten and T. M. Khoshgoftaar, “A Survey on Image Data Augmentation for Deep Learning,” *J. Big Data*, vol. 6, no. 1, p. 60, Dec. 2019, doi: 10.1186/s40537-019-0197-0.
- [35] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, 1st ed. Cambridge, MA: MIT Press, 2016. Accessed: May 08, 2026. [Online]. Available: <https://www.deeplearningbook.org/>
- [36] H. Bichri, A. Chergui, and M. Hain, “Investigating the Impact of Train / Test Split Ratio on the Performance of Pre-Trained Models with Custom Datasets,” *International Journal of Advanced Computer Science and Applications*, vol. 15, no. 2, pp. 331–339, 2024, doi: 10.14569/IJACSA.2024.0150235.
- [37] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” in *Proceedings of the 26th Annual Conference on Neural Information Processing Systems (NIPS 2012)*, Lake Tahoe, Nevada, USA: Curran Associates, Inc., Dec. 2012, pp. 1097–1105. [Online]. Available: <http://code.google.com/p/cuda-convnet/>
- [38] N. Srivastava, G. Hinton, A. Krizhevsky, and R. Salakhutdinov, “Dropout: A Simple Way to Prevent Neural Networks from Overfitting,” *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014.
- [39] Y. Bengio, “Practical Recommendations for Gradient-Based Training of Deep Architectures,” in *Neural Networks: Tricks of the Trade*, 2nd ed., vol. 7700, Berlin, Heidelberg: Springer, 2012, ch. 26, pp. 437–478. doi: 10.1007/978-3-642-35289-8_26.
- [40] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” in *3rd International Conference on Learning Representations (ICLR 2015), Conference Track Proceedings*, San Diego, CA, May 2015. [Online]. Available: <http://arxiv.org/abs/1412.6980>
- [41] O. Russakovsky *et al.*, “ImageNet Large Scale Visual Recognition Challenge,” *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015, doi: 10.1007/s11263-015-0816-y.
- [42] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, “How Transferable are Features in Deep Neural Networks?,” in *Advances in Neural Information*

Processing Systems 27 (NIPS 2014), Montreal, Canada: Curran Associates, Inc., Nov. 2014, pp. 3320–3328. [Online]. Available: <http://arxiv.org/abs/1411.1792>

- [43] M. Murinto, M. Rosyda, and M. Melany, “Klasifikasi Jenis Biji Kopi Menggunakan Convolutional Neural Network dan Transfer Learning pada Model VGG16 dan MobileNetV2,” *JRST (Jurnal Riset Sains dan Teknologi)*, vol. 7, no. 2, p. 183, Sep. 2023, doi: 10.30595/jrst.v7i2.16788.

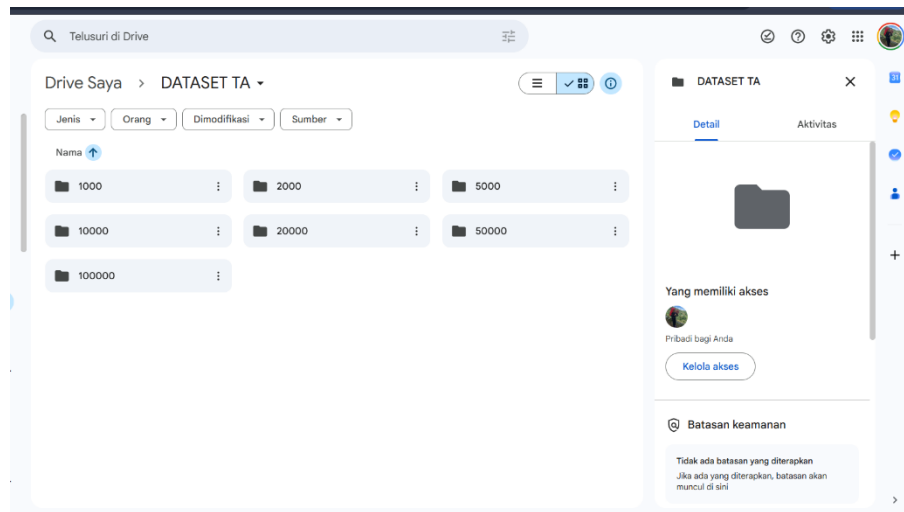
LAMPIRAN

Lampiran 1. Dataset Penelitian

a. Struktur Folder Dataset

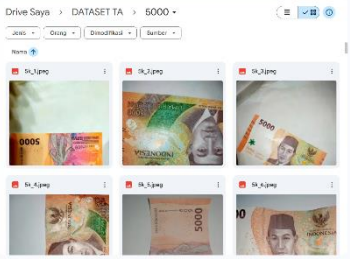
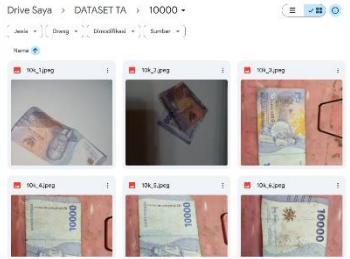
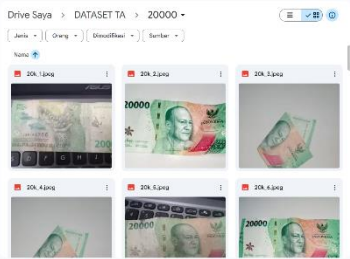
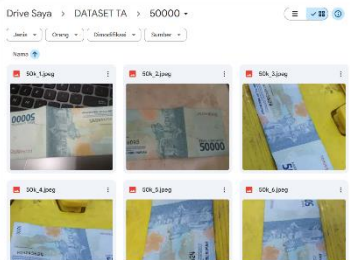
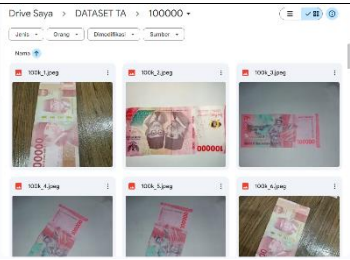
Dataset 1 : <https://www.kaggle.com/daniakhmadmaulana/citra-uang-rupiah-emisi-2022>

Dataset 2 : <https://www.kaggle.com/fannyzahrahmadhan/uang-emisi-2022-baru>



b. Contoh Citra Dataset

No	Nominal	Contoh Citra
(1)	(2)	(3)
1	Rp1.000	
2	Rp2.000	

3	Rp5.000	
4	Rp10.000	
5	Rp20.000	
6	Rp50.000	
7	Rp100.000	

Lampiran 2. Source Code CNN Custom

```
# Cell 1 - Mount Google Drive dan Instalasi Library

from google.colab import drive
drive.mount('/content/drive')
!pip install split-folders -q

# Cell 2 - Import Library

import os                                # untuk operasi path dan folder
import json                             # untuk menyimpan hasil
evaluasi ke Drive
```

```

import numpy as np                # untuk operasi array dan
matriks
import matplotlib.pyplot as plt  # untuk visualisasi grafik
training
import seaborn as sns            # untuk visualisasi confusion
matrix
import splitfolders              # untuk membagi dataset
train/val/test
import tensorflow as tf          # framework utama deep learning
from tensorflow.keras.preprocessing.image import
ImageDataGenerator # augmentasi dan preprocessing gambar
from tensorflow.keras.models import
Sequential                  # untuk membangun arsitektur
CNN Custom
from tensorflow.keras.layers import Conv2D, MaxPooling2D,
Flatten, Dense, Dropout, Input # layer-layer CNN
from tensorflow.keras.callbacks import
EarlyStopping               # untuk menghentikan training
otomatis
from sklearn.metrics import classification_report,
confusion_matrix           # untuk evaluasi metrik dan confusion matrix
import time                 # untuk mengukur inference time

# Cell 3 - Split Dataset (70% Train, 15% Validasi, 15% Test)

dataset_path = '/content/drive/MyDrive/DATASET TA'
output_path = '/content/dataset_split'

splitfolders.ratio(
    dataset_path,
    output=output_path,
    seed=42,
    ratio=(0.7, 0.15, 0.15)
)

print("Split selesai.")
print("Isi folder hasil split:")
for folder in ['train', 'val', 'test']:
    total = 0
    for kelas in os.listdir(os.path.join(output_path, folder)):
        jumlah = len(os.listdir(os.path.join(output_path,
folder, kelas)))
        total += jumlah
    print(f" {folder}: {total} gambar")

# Cell 4 - Preprocessing dan Augmentasi Data

img_size = (224, 224)
batch_size = 32

train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    horizontal_flip=True,
    zoom_range=0.2,
    width_shift_range=0.2,
    height_shift_range=0.2
)

```

```

val_test_datagen = ImageDataGenerator(rescale=1./255)

train_data = train_datagen.flow_from_directory(
    os.path.join(output_path, 'train'),
    target_size=img_size,
    batch_size=batch_size,
    class_mode='categorical'
)

val_data = val_test_datagen.flow_from_directory(
    os.path.join(output_path, 'val'),
    target_size=img_size,
    batch_size=batch_size,
    class_mode='categorical'
)

test_data = val_test_datagen.flow_from_directory(
    os.path.join(output_path, 'test'),
    target_size=img_size,
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=False
)

print("Label kelas:", train_data.class_indices)

# Cell 4b - Visualisasi Teknik Augmentasi Data

import random
import matplotlib.pyplot as plt
import numpy as np
import os
from tensorflow.keras.preprocessing.image import load_img,
img_to_array, ImageDataGenerator

# Ambil 1 gambar secara acak dari kelas tertentu
test_folder = os.path.join(output_path, 'test')
kelas_list = sorted(os.listdir(test_folder))
img_path = os.path.join(test_folder, kelas_list[5],
random.choice(os.listdir(os.path.join(test_folder,
kelas_list[5]))))
img_asli = load_img(img_path, target_size=(224, 224))
img_array = img_to_array(img_asli) / 255.0
img_expanded = np.expand_dims(img_array, axis=0)

augmentasi = {
    'Asli': ImageDataGenerator(),
    'Rotasi': ImageDataGenerator(rotation_range=20),
    'Flip Horizontal': ImageDataGenerator(horizontal_flip=True),
    'Zoom': ImageDataGenerator(zoom_range=0.2),
    'Geser Horizontal':
ImageDataGenerator(width_shift_range=0.2),
    'Geser Vertikal': ImageDataGenerator(height_shift_range=0.2)
}

fig, axes = plt.subplots(1, 6, figsize=(20, 4))
fig.suptitle('Teknik Augmentasi Data', fontsize=14)

for idx, (nama, aug) in enumerate(augmentasi.items()):

```

```

        aug_iter = aug.flow(img_expanded, batch_size=1)
        aug_img = next(aug_iter)[0]
        axes[idx].imshow(aug_img)
        axes[idx].set_title(nama, fontsize=10)
        axes[idx].axis('off')

plt.tight_layout()
plt.savefig('/content/drive/MyDrive/visualisasi_teknik_augmentas
i.png', dpi=150, bbox_inches='tight')
plt.show()
print("Gambar visualisasi teknik augmentasi tersimpan ke
Drive.")

# Cell 5 - Arsitektur CNN Custom

model_cnn = Sequential([
    Input(shape=(224, 224, 3)),

    # Blok Konvolusi 1
    Conv2D(32, (3, 3), activation='relu'),
    MaxPooling2D(pool_size=(2, 2)),

    # Blok Konvolusi 2
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D(pool_size=(2, 2)),

    # Blok Konvolusi 3
    Conv2D(128, (3, 3), activation='relu'),
    MaxPooling2D(pool_size=(2, 2)),

    # Flatten dan Fully Connected Layer
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),

    # Output Layer
    Dense(7, activation='softmax')
])

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

model_cnn.summary()

# Cell 6 - Training CNN Custom

early_stopping = EarlyStopping(
    monitor='val_loss',
    patience=10,
    restore_best_weights=True
)

history_cnn = model_cnn.fit(
    train_data,
    epochs=50,
    validation_data=val_data,

```

```

        callbacks=[early_stopping]
    )

    model_cnn.save('/content/drive/MyDrive/model_cnn_run1.keras')
    print("Model CNN Custom Run 1 tersimpan ke Drive.")

    history_dict = {key: [float(v) for v in val] for key, val in
        history_cnn.history.items()}
    with open('/content/drive/MyDrive/history_cnn_run1.json', 'w')
    as f:
        json.dump(history_dict, f)
    print("History CNN Custom Run 1 tersimpan ke Drive.")

    # Load model CNN Custom Run 1 dari Drive (jalankan jika sesi
    reset)

    from tensorflow.keras.models import load_model

    model_cnn =
    load_model('/content/drive/MyDrive/model_cnn_run1.keras')
    print("Model CNN Custom Run 1 berhasil di-load.")
    # Load history CNN Custom Run 1 dari Drive (jalankan jika sesi
    reset)

    with open('/content/drive/MyDrive/history_cnn_run1.json', 'r')
    as f:
        history_dict = json.load(f)

    class HistoryObj:
        def __init__(self, d):
            self.history = d

    history_cnn = HistoryObj(history_dict)

    print("History CNN Custom Run 1 berhasil di-load.")

    # Cell 7 - Evaluasi CNN Custom Run 1

    test_loss_cnn, test_acc_cnn = model_cnn.evaluate(test_data,
        verbose=0)
    print(f"Test Accuracy CNN Custom : {test_acc_cnn:.4f}")
    print(f"Test Loss CNN Custom      : {test_loss_cnn:.4f}")

    y_pred_cnn = model_cnn.predict(test_data)
    y_pred_cnn_classes = np.argmax(y_pred_cnn, axis=1)
    y_true = test_data.classes

    class_names = list(test_data.class_indices.keys())
    print("\nClassification Report CNN Custom Run 1:")
    print(classification_report(y_true, y_pred_cnn_classes,
        target_names=class_names))

    sample_batch = next(iter(test_data))[0][:1]
    start = time.time()
    for _ in range(100):
        model_cnn.predict(sample_batch, verbose=0)
    end = time.time()
    inference_time_cnn = (end - start) / 100 * 1000

```

```

print(f"Inference Time CNN Custom: {inference_time_cnn:.2f} ms
per gambar")

model_size_cnn =
os.path.getsize('/content/drive/MyDrive/model_cnn_run1.keras') /
(1024 * 1024)
print(f"Ukuran Model CNN Custom : {model_size_cnn:.2f} MB")

# Simpan hasil evaluasi lengkap ke Drive
from sklearn.metrics import precision_score, recall_score,
f1_score
hasil_run1 = {
    'accuracy': float(test_acc_cnn),
    'loss': float(test_loss_cnn),
    'precision': float(precision_score(y_true,
y_pred_cnn_classes, average='macro')),
    'recall': float(recall_score(y_true, y_pred_cnn_classes,
average='macro')),
    'f1_score': float(f1_score(y_true, y_pred_cnn_classes,
average='macro')),
    'inference_time_ms': float(inference_time_cnn),
    'model_size_mb': float(model_size_cnn)
}
with open('/content/drive/MyDrive/hasil_cnn_run1.json', 'w') as
f:
    json.dump(hasil_run1, f)
print("\nHasil evaluasi CNN Custom Run 1 tersimpan ke Drive.")

# Cell 8 - Tabel Evaluasi Per Kelas CNN Custom Run 1
# Accuracy dihitung sesuai rumus Bab II

from sklearn.metrics import classification_report,
multilabel_confusion_matrix
import pandas as pd

# Classification report dalam bentuk dictionary
report = classification_report(
    y_true,
    y_pred_cnn_classes,
    target_names=class_names,
    output_dict=True,
    zero_division=0
)

# Multilabel Confusion Matrix
mcm = multilabel_confusion_matrix(y_true, y_pred_cnn_classes)

hasil = []

for i, class_name in enumerate(class_names):

    tn, fp, fn, tp = mcm[i].ravel()

    # Accuracy sesuai rumus Bab II
    accuracy = (tp + tn) / (tp + tn + fp + fn)

    hasil.append({
        "Nominal": class_name,
        "Accuracy (%)": round(accuracy * 100, 2),

```

```

        "Precision (%)": round(report[class_name]["precision"] *
100, 2),
        "Recall (%)": round(report[class_name]["recall"] * 100,
2),
        "F1-Score (%)": round(report[class_name]["f1-score"] *
100, 2),
        "Support": int(report[class_name]["support"]))
    })

df_hasil = pd.DataFrame(hasil)

print("Tabel Evaluasi Per Kelas CNN Custom Run 1")
display(df_hasil)

# Cell 8b - Visualisasi Grafik Training CNN Custom Run 1

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(14, 5))

# Grafik Accuracy
ax1.plot(history_cnn.history['accuracy'], label='Train
Accuracy')
ax1.plot(history_cnn.history['val_accuracy'], label='Val
Accuracy')
ax1.set_title('CNN Custom Run 1 - Accuracy')
ax1.set_xlabel('Epoch')
ax1.set_ylabel('Accuracy')
ax1.legend()
ax1.grid(True)

# Grafik Loss
ax2.plot(history_cnn.history['loss'], label='Train Loss')
ax2.plot(history_cnn.history['val_loss'], label='Val Loss')
ax2.set_title('CNN Custom Run 1 - Loss')
ax2.set_xlabel('Epoch')
ax2.set_ylabel('Loss')
ax2.legend()
ax2.grid(True)

plt.tight_layout()
plt.savefig('/content/drive/MyDrive/grafik_cnn_run1.png',
dpi=150, bbox_inches='tight')
plt.show()

# Cell 9 - Confusion Matrix CNN Custom Run 1

cm_cnn = confusion_matrix(y_true, y_pred_cnn_classes)
class_names = list(test_data.class_indices.keys())

plt.figure(figsize=(10, 8))
sns.heatmap(cm_cnn, annot=True, fmt='d', cmap='Blues',
            xticklabels=class_names,
            yticklabels=class_names)
plt.title('Confusion Matrix - CNN Custom Run 1')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.tight_layout()
plt.savefig('/content/drive/MyDrive/cm_cnn_run1.png', dpi=150,
bbox_inches='tight')
plt.show()

```

```

# Cell - TP, TN, FP, FN per Kelas CNN Custom Run 1
# Menghitung nilai TP, TN, FP, dan FN untuk setiap kelas
berdasarkan confusion matrix

from sklearn.metrics import multilabel_confusion_matrix
import pandas as pd

mcm = multilabel_confusion_matrix(y_true, y_pred_cnn_classes)
class_names = list(test_data.class_indices.keys())

hasil_tp = []
print("TP, TN, FP, FN per Kelas - CNN Custom Run 1")
print("=" * 60)
for i, class_name in enumerate(class_names):
    tn, fp, fn, tp = mcm[i].ravel()
    hasil_tp.append({
        "Kelas": class_name,
        "TP": tp,
        "FP": fp,
        "FN": fn,
        "TN": tn
    })

df_tp = pd.DataFrame(hasil_tp)
df_tp = df_tp[["Kelas", "TP", "FP", "FN", "TN"]]
df_tp.index = df_tp.index + 1
display(df_tp)

# Cell - TP, TN, FP, FN per Kelas CNN Custom Run 1
# Menghitung nilai TP, TN, FP, dan FN untuk setiap kelas
berdasarkan confusion matrix

from sklearn.metrics import multilabel_confusion_matrix
import pandas as pd

mcm = multilabel_confusion_matrix(y_true, y_pred_cnn_classes)
class_names = list(test_data.class_indices.keys())

hasil_tp = []
print("TP, TN, FP, FN per Kelas - CNN Custom Run 1")
print("=" * 60)
for i, class_name in enumerate(class_names):
    tn, fp, fn, tp = mcm[i].ravel()
    hasil_tp.append({
        "Kelas": class_name,
        "TP": tp,
        "FP": fp,
        "FN": fn,
        "TN": tn
    })

df_tp = pd.DataFrame(hasil_tp)
df_tp = df_tp[["Kelas", "TP", "FP", "FN", "TN"]]
df_tp.index = df_tp.index + 1
display(df_tp)

```

Lampiran 3. Source Code MobileNetV2

```
# Cell 1 - Mount Google Drive dan Instalasi Library

from google.colab import drive
drive.mount('/content/drive')

!pip install split-folders -q

# Cell 2 - Import Library

import os                    # operasi path dan folder
import json                  # simpan hasil ke file JSON
import numpy as np           # operasi array
import matplotlib.pyplot as plt # visualisasi grafik
import seaborn as sns        # visualisasi confusion matrix
import splitfolders          # split dataset
import tensorflow as tf      # framework deep learning
from tensorflow.keras.preprocessing.image import
ImageDataGenerator # augmentasi dan preprocessing
from tensorflow.keras.models import
Model              # Functional API untuk
MobileNetV2
from tensorflow.keras.layers import Dense, Dropout,
GlobalAveragePooling2D, Input # layer klasifikasi baru
from tensorflow.keras.applications import
MobileNetV2            # pretrained MobileNetV2
from tensorflow.keras.callbacks import
EarlyStopping          # early stopping
from sklearn.metrics import classification_report,
confusion_matrix, precision_score, recall_score, f1_score #
metrik evaluasi
import time            # ukur inference time

# Cell 3 - Split Dataset (70% Train, 15% Validasi, 15% Test)

dataset_path = '/content/drive/MyDrive/DATASET TA'
output_path = '/content/dataset_split'

splitfolders.ratio(
    dataset_path,
    output=output_path,
    seed=42,
    ratio=(0.7, 0.15, 0.15)
)

print("Split selesai.")
print("Isi folder hasil split:")
for folder in ['train', 'val', 'test']:
```

```

        total = 0
        for kelas in os.listdir(os.path.join(output_path, folder)):
            jumlah = len(os.listdir(os.path.join(output_path,
folder, kelas)))
            total += jumlah
print(f"  {folder}: {total} gambar")

# Cell 4 - Preprocessing dan Augmentasi Data

img_size = (224, 224)
batch_size = 32

# Augmentasi hanya untuk data train
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    horizontal_flip=True,
    zoom_range=0.2,
    width_shift_range=0.2,
    height_shift_range=0.2
)

# Val dan test hanya dinormalisasi
val_test_datagen = ImageDataGenerator(rescale=1./255)

train_data = train_datagen.flow_from_directory(
    os.path.join(output_path, 'train'),
    target_size=img_size,
    batch_size=batch_size,
    class_mode='categorical'
)

val_data = val_test_datagen.flow_from_directory(
    os.path.join(output_path, 'val'),
    target_size=img_size,
    batch_size=batch_size,
    class_mode='categorical'
)

test_data = val_test_datagen.flow_from_directory(
    os.path.join(output_path, 'test'),
    target_size=img_size,
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=False
)

print("Label kelas:", train_data.class_indices)

# Cell 4b - Visualisasi Teknik Augmentasi Data

import random
import matplotlib.pyplot as plt
import numpy as np
import os
from tensorflow.keras.preprocessing.image import load_img,
img_to_array, ImageDataGenerator

```

```

# Ambil 1 gambar secara acak dari kelas tertentu
test_folder = os.path.join(output_path, 'test')
kelas_list = sorted(os.listdir(test_folder))
img_path = os.path.join(test_folder, kelas_list[2],
random.choice(os.listdir(os.path.join(test_folder,
kelas_list[2]))))
img_asli = load_img(img_path, target_size=(224, 224))
img_array = img_to_array(img_asli) / 255.0
img_expanded = np.expand_dims(img_array, axis=0)

augmentasi = {
    'Asli': ImageDataGenerator(),
    'Rotasi': ImageDataGenerator(rotation_range=20),
    'Flip Horizontal': ImageDataGenerator(horizontal_flip=True),
    'Zoom': ImageDataGenerator(zoom_range=0.2),
    'Geser Horizontal':
ImageDataGenerator(width_shift_range=0.2),
    'Geser Vertikal': ImageDataGenerator(height_shift_range=0.2)
}

fig, axes = plt.subplots(1, 6, figsize=(20, 4))
fig.suptitle('Teknik Augmentasi Data', fontsize=14)

for idx, (nama, aug) in enumerate(augmentasi.items()):
    aug_iter = aug.flow(img_expanded, batch_size=1)
    aug_img = next(aug_iter)[0]
    axes[idx].imshow(aug_img)
    axes[idx].set_title(nama, fontsize=10)
    axes[idx].axis('off')

plt.tight_layout()
plt.savefig('/content/drive/MyDrive/visualisasi_teknik_augmentas
i.png', dpi=150, bbox_inches='tight')
plt.show()
print("Gambar visualisasi teknik augmentasi tersimpan ke
Drive.")

# Cell 5 - Load Pre-trained MobileNetV2

# Memuat MobileNetV2 dengan bobot ImageNet, tanpa lapisan
klasifikasi bawaan
base_model = MobileNetV2(
    input_shape=(224, 224, 3),
    include_top=False,
    weights='imagenet'
)

# Bekukan semua layer base model untuk training tahap 1
base_model.trainable = False

print(f"Jumlah layer base model: {len(base_model.layers)}")
print(f"Trainable: {base_model.trainable}")

# Cell 6 - Bangun Arsitektur MobileNetV2

inputs = Input(shape=(224, 224, 3)) # titik masuk data, gambar
224x224 RGB
x = base_model(inputs, training=False) # alirkan input ke base
model, training=False agar BatchNorm pakai statistik ImageNet

```

```

x = GlobalAveragePooling2D()(x) # rangkum feature map 7x7x1280
jadi vektor 1280
x = Dense(128, activation='relu')(x) # fully connected layer
untuk klasifikasi
x = Dropout(0.5)(x) # nonaktifkan 50% neuron untuk cegah
overfitting
outputs = Dense(7, activation='softmax')(x) # output 7 kelas,
softmax mengubah ke probabilitas

model_mobilenet = Model(inputs, outputs) # gabungkan semua
layer jadi satu model

model_mobilenet.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=0.0001), #
    LR lebih kecil dari CNN Custom agar tidak merusak bobot
    pretrained
    loss='categorical_crossentropy', # loss standar untuk
    klasifikasi multi-kelas
    metrics=['accuracy']
)

model_mobilenet.summary()

# Cell 7 - Training MobileNetV2 Tahap 1 (Base Model Freeze)

early_stopping = EarlyStopping(
    monitor='val_loss',
    patience=10,
    restore_best_weights=True
)

history_mobilenet_1 = model_mobilenet.fit(
    train_data,
    epochs=50,
    validation_data=val_data,
    callbacks=[early_stopping]
)

# Simpan history tahap 1 ke Drive
history_dict_1 = {key: [float(v) for v in val] for key, val in
history_mobilenet_1.history.items()}
with open('/content/drive/MyDrive/history_mobilenet1_run4.json',
'w') as f:
    json.dump(history_dict_1, f)
print("History MobileNetV2 tahap 1 Run 4 tersimpan ke Drive.")

# Cell 8 - Fine-Tuning MobileNetV2 Tahap 2

# Buka layer 100-153 base model untuk dilatih ulang
base_model.trainable = True
for layer in base_model.layers[:100]:
    layer.trainable = False # layer awal tetap dibekukan

# Compile ulang dengan learning rate lebih kecil
model_mobilenet.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=0.00001),
    # LR sangat kecil agar tidak merusak fitur pretrained
    loss='categorical_crossentropy',

```

```

        metrics=['accuracy']
    )

    early_stopping = EarlyStopping(
        monitor='val_loss',
        patience=10,
        restore_best_weights=True
    )

    history_mobilenet_2 = model_mobilenet.fit(
        train_data,
        epochs=50,
        validation_data=val_data,
        callbacks=[early_stopping]
    )

    # Simpan model dan history tahap 2 ke Drive
    model_mobilenet.save('/content/drive/MyDrive/model_mobilenet_run
4.keras')
    print("Model MobileNetV2 Run 4 tersimpan ke Drive.")

    history_dict_2 = {key: [float(v) for v in val] for key, val in
history_mobilenet_2.history.items()}
    with open('/content/drive/MyDrive/history_mobilenet2_run4.json',
'w') as f:
        json.dump(history_dict_2, f)
    print("History MobileNetV2 tahap 2 Run 4 tersimpan ke Drive.")

    # Load model MobileNetV2 Run 4 dari Drive (jalankan jika sesi
reset)

    from tensorflow.keras.models import load_model

    model_mobilenet =
load_model('/content/drive/MyDrive/model_mobilenet_run4.keras')
    print("Model MobileNetV2 Run 4 berhasil di-load.")

    # Load history MobileNetV2 Run 4 dari Drive (jalankan jika sesi
reset)

    with open('/content/drive/MyDrive/history_mobilenet1_run4.json',
'r') as f:
        h1 = json.load(f)

    with open('/content/drive/MyDrive/history_mobilenet2_run4.json',
'r') as f:
        h2 = json.load(f)

    class HistoryObj:
        def __init__(self, d):
            self.history = d

    history_mobilenet_1 = HistoryObj(h1)
    history_mobilenet_2 = HistoryObj(h2)

    print("History MobileNetV2 Run 4 berhasil di-load.")

    # Cell 9 - Evaluasi MobileNetV2 Run 4

```

```

test_loss_mobilenet, test_acc_mobilenet =
model_mobilenet.evaluate(test_data, verbose=0)
print(f"Test Accuracy MobileNetV2 : {test_acc_mobilenet:.4f}")
print(f"Test Loss MobileNetV2      : {test_loss_mobilenet:.4f}")

y_pred_mobilenet = model_mobilenet.predict(test_data)
y_pred_mobilenet_classes = np.argmax(y_pred_mobilenet, axis=1)
y_true = test_data.classes

class_names = list(test_data.class_indices.keys())
print("\nClassification Report MobileNetV2 Run 4:")
print(classification_report(y_true, y_pred_mobilenet_classes,
target_names=class_names))

# Inference time diukur 100x untuk hasil yang lebih stabil
sample_batch = next(iter(test_data))[0][:1]
start = time.time()
for _ in range(100):
    model_mobilenet.predict(sample_batch, verbose=0)
end = time.time()
inference_time_mobilenet = (end - start) / 100 * 1000
print(f"Inference Time MobileNetV2 :
{inference_time_mobilenet:.2f} ms per gambar")

model_size_mobilenet =
os.path.getsize('/content/drive/MyDrive/model_mobilenet_run4.ker
as') / (1024 * 1024)
print(f"Ukuran Model MobileNetV2      : {model_size_mobilenet:.2f}
MB")

# Simpan hasil evaluasi ke Drive
hasil_run4 = {
    'accuracy': float(test_acc_mobilenet),
    'loss': float(test_loss_mobilenet),
    'precision': float(precision_score(y_true,
y_pred_mobilenet_classes, average='macro')),
    'recall': float(recall_score(y_true,
y_pred_mobilenet_classes, average='macro')),
    'f1_score': float(f1_score(y_true, y_pred_mobilenet_classes,
average='macro')),
    'inference_time_ms': float(inference_time_mobilenet),
    'model_size_mb': float(model_size_mobilenet)
}
with open('/content/drive/MyDrive/hasil_mobilenet_run4.json',
'w') as f:
    json.dump(hasil_run4, f)
print("\nHasil evaluasi MobileNetV2 Run 4 tersimpan ke Drive.")

# Cell - Tabel Evaluasi Per Kelas MobileNetV2 Run 4
# Accuracy dihitung sesuai rumus Bab II

from sklearn.metrics import classification_report,
multilabel_confusion_matrix
import pandas as pd

# Classification Report dalam bentuk dictionary
report = classification_report(
    y_true,
    y_pred_mobilenet_classes,

```

```

        target_names=class_names,
        output_dict=True,
        zero_division=0
    )

    # Multilabel Confusion Matrix
    mcm = multilabel_confusion_matrix(y_true,
    y_pred_mobilenet_classes)

    hasil = []

    for i, class_name in enumerate(class_names):

        tn, fp, fn, tp = mcm[i].ravel()

        accuracy = (tp + tn) / (tp + tn + fp + fn)

        hasil.append({
            "Nominal": class_name,
            "Accuracy (%)": round(accuracy * 100, 2),
            "Precision (%)": round(report[class_name]["precision"] *
100, 2),
            "Recall (%)": round(report[class_name]["recall"] * 100,
2),
            "F1-Score (%)": round(report[class_name]["f1-score"] *
100, 2),
            "Support": int(report[class_name]["support"])
        })

    df_hasil = pd.DataFrame(hasil)

    print("Tabel Evaluasi Per Kelas MobileNetV2 Run 4")
    display(df_hasil)

# Cell 10 - Visualisasi Grafik Training MobileNetV2 Run 4

# Gabungkan history tahap 1 dan tahap 2
history_combined = {
    'accuracy': history_mobilenet_1.history['accuracy'] +
history_mobilenet_2.history['accuracy'],
    'val_accuracy': history_mobilenet_1.history['val_accuracy']
+ history_mobilenet_2.history['val_accuracy'],
    'loss': history_mobilenet_1.history['loss'] +
history_mobilenet_2.history['loss'],
    'val_loss': history_mobilenet_1.history['val_loss'] +
history_mobilenet_2.history['val_loss'],
}

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(14, 5))

# Grafik Accuracy
ax1.plot(history_combined['accuracy'], label='Train Accuracy')
ax1.plot(history_combined['val_accuracy'], label='Val Accuracy')
ax1.axvline(x=50, color='red', linestyle='--', label='Fine-
tuning dimulai')
ax1.set_title('MobileNetV2 Run 4 - Accuracy')
ax1.set_xlabel('Epoch')
ax1.set_ylabel('Accuracy')

```

```

ax1.legend()
ax1.grid(True)

# Grafik Loss
ax2.plot(history_combined['loss'], label='Train Loss')
ax2.plot(history_combined['val_loss'], label='Val Loss')
ax2.axvline(x=50, color='red', linestyle='--', label='Fine-
tuning dimulai')
ax2.set_title('MobileNetV2 Run 4 - Loss')
ax2.set_xlabel('Epoch')
ax2.set_ylabel('Loss')
ax2.legend()
ax2.grid(True)

plt.tight_layout()
plt.savefig('/content/drive/MyDrive/grafik_mobilenet_run4.png',
dpi=150, bbox_inches='tight')
plt.show()

# Cell 11 - Confusion Matrix MobileNetV2 Run 4

cm_mobilenet = confusion_matrix(y_true,
y_pred_mobilenet_classes)
class_names = list(test_data.class_indices.keys())

plt.figure(figsize=(10, 8))
sns.heatmap(cm_mobilenet, annot=True, fmt='d', cmap='Blues',
            xticklabels=class_names,
            yticklabels=class_names)
plt.title('Confusion Matrix - MobileNetV2 Run 4')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.tight_layout()
plt.savefig('/content/drive/MyDrive/cm_mobilenet_run4.png',
dpi=150, bbox_inches='tight')
plt.show()
# Cell - TP, TN, FP, FN per Kelas MobilenetV2 Run 4
# Menghitung nilai TP, TN, FP, dan FN untuk setiap kelas
berdasarkan confusion matrix

from sklearn.metrics import multilabel_confusion_matrix
import pandas as pd

mcm = multilabel_confusion_matrix(y_true,
y_pred_mobilenet_classes)
class_names = list(test_data.class_indices.keys())

hasil_tp = []
print("TP, TN, FP, FN per Kelas - MobileNetV2 Run 4")
print("=" * 60)
for i, class_name in enumerate(class_names):
    tn, fp, fn, tp = mcm[i].ravel()
    hasil_tp.append({
        "Kelas": class_name,
        "TP": tp,
        "FP": fp,
        "FN": fn,
        "TN": tn
    })

```

```
df_tp = pd.DataFrame(hasil_tp)
df_tp = df_tp[["Kelas", "TP", "FP", "FN", "TN"]]
df_tp.index = df_tp.index + 1
display(df_tp)
```

Lampiran 4. Source Code Analisis Hasil Penelitian

```
# Cell 1 - Mount Google Drive dan Import Library
# Mengimpor library yang diperlukan dan menghubungkan ke Google
Drive
from google.colab import drive
drive.mount('/content/drive')

import json
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Cell 2 - Load Hasil Evaluasi Semua Run dari Drive
# Memuat file hasil evaluasi dari setiap percobaan yang tersimpan
di Google Drive
# Load hasil CNN Custom (5 run)
with open('/content/drive/MyDrive/hasil_cnn_run1.json', 'r') as
f:
    cnn_run1 = json.load(f)
with open('/content/drive/MyDrive/hasil_cnn_run2.json', 'r') as
f:
    cnn_run2 = json.load(f)
with open('/content/drive/MyDrive/hasil_cnn_run3.json', 'r') as
f:
    cnn_run3 = json.load(f)
with open('/content/drive/MyDrive/hasil_cnn_run4.json', 'r') as
f:
    cnn_run4 = json.load(f)
with open('/content/drive/MyDrive/hasil_cnn_run5.json', 'r') as
f:
    cnn_run5 = json.load(f)

# Load hasil MobileNetV2 (5 run)
with open('/content/drive/MyDrive/hasil_mobilenet_run1.json',
'r') as f:
    mobilenet_run1 = json.load(f)
with open('/content/drive/MyDrive/hasil_mobilenet_run2.json',
'r') as f:
    mobilenet_run2 = json.load(f)
with open('/content/drive/MyDrive/hasil_mobilenet_run3.json',
'r') as f:
    mobilenet_run3 = json.load(f)
with open('/content/drive/MyDrive/hasil_mobilenet_run4.json',
'r') as f:
    mobilenet_run4 = json.load(f)
with open('/content/drive/MyDrive/hasil_mobilenet_run5.json',
'r') as f:
    mobilenet_run5 = json.load(f)
```

```

print('Semua data berhasil di-load.')

# Cell 3 - Tabel Hasil Per Run
# Menampilkan hasil evaluasi seluruh percobaan dalam bentuk tabel
cnn_runs = [cnn_run1, cnn_run2, cnn_run3, cnn_run4, cnn_run5]
mobilenet_runs = [mobilenet_run1, mobilenet_run2, mobilenet_run3,
mobilenet_run4, mobilenet_run5]

data_per_run = []
for i, run in enumerate(cnn_runs, 1):
    data_per_run.append({
        'Model': 'CNN Custom',
        'Run': f'Run {i}',
        'Accuracy (%)': round(run['accuracy'] * 100, 2),
        'Precision (%)': round(run['precision'] * 100, 2),
        'Recall (%)': round(run['recall'] * 100, 2),
        'F1-Score (%)': round(run['f1_score'] * 100, 2),
        'Inference Time (ms)': round(run['inference_time_ms'],
2),
        'Ukuran Model (MB)': round(run['model_size_mb'], 2)
    })

for i, run in enumerate(mobilenet_runs, 1):
    data_per_run.append({
        'Model': 'MobileNetV2',
        'Run': f'Run {i}',
        'Accuracy (%)': round(run['accuracy'] * 100, 2),
        'Precision (%)': round(run['precision'] * 100, 2),
        'Recall (%)': round(run['recall'] * 100, 2),
        'F1-Score (%)': round(run['f1_score'] * 100, 2),
        'Inference Time (ms)': round(run['inference_time_ms'],
2),
        'Ukuran Model (MB)': round(run['model_size_mb'], 2)
    })

df_per_run = pd.DataFrame(data_per_run)
df_per_run.index = df_per_run.index + 1

print('Tabel Hasil Per Run')
print('=' * 80)
display(df_per_run)

# Cell 4 - Hitung Mean dan Standar Deviasi
# Menghitung nilai rata-rata dan standar deviasi dari seluruh
percobaan

metrics = ['accuracy', 'precision', 'recall', 'f1_score',
'inference_time_ms', 'model_size_mb']

cnn_mean = {m: np.mean([r[m] for r in cnn_runs]) for m in metrics}
cnn_std = {m: np.std([r[m] for r in cnn_runs], ddof=1) for m in
metrics}

mobilenet_mean = {m: np.mean([r[m] for r in mobilenet_runs]) for
m in metrics}
mobilenet_std = {m: np.std([r[m] for r in mobilenet_runs],
ddof=1) for m in metrics}

```

```

rekap = {
    'Metrik': ['Accuracy (%)', 'Precision (%)', 'Recall (%)', 'F1-
Score (%)', 'Inference Time (ms)', 'Ukuran Model (MB)'],
    'CNN Mean': [
        round(cnn_mean['accuracy']*100, 2),
        round(cnn_mean['precision']*100, 2),
        round(cnn_mean['recall']*100, 2),
        round(cnn_mean['f1_score']*100, 2),
        round(cnn_mean['inference_time_ms'], 2),
        round(cnn_mean['model_size_mb'], 2)
    ],
    'CNN SD': [
        round(cnn_std['accuracy']*100, 2),
        round(cnn_std['precision']*100, 2),
        round(cnn_std['recall']*100, 2),
        round(cnn_std['f1_score']*100, 2),
        round(cnn_std['inference_time_ms'], 2),
        round(cnn_std['model_size_mb'], 2)
    ],
    'MobileNetV2 Mean': [
        round(mobilenet_mean['accuracy']*100, 2),
        round(mobilenet_mean['precision']*100, 2),
        round(mobilenet_mean['recall']*100, 2),
        round(mobilenet_mean['f1_score']*100, 2),
        round(mobilenet_mean['inference_time_ms'], 2),
        round(mobilenet_mean['model_size_mb'], 2)
    ],
    'MobileNetV2 SD': [
        round(mobilenet_std['accuracy']*100, 2),
        round(mobilenet_std['precision']*100, 2),
        round(mobilenet_std['recall']*100, 2),
        round(mobilenet_std['f1_score']*100, 2),
        round(mobilenet_std['inference_time_ms'], 2),
        round(mobilenet_std['model_size_mb'], 2)
    ]
}

df_rekap = pd.DataFrame(rekap)
df_rekap.index = df_rekap.index + 1

print('Tabel Rekapitulasi Mean dan Standar Deviasi')
print('=' * 80)
display(df_rekap)
# Cell 5 - Grafik Perbandingan Accuracy dan F1-Score per Percobaan
# Membandingkan nilai Accuracy dan F1-Score CNN Custom dan
MobileNetV2 pada setiap percobaan

runs = ['Run 1', 'Run 2', 'Run 3', 'Run 4', 'Run 5']
cnn_acc = [round(r['accuracy']*100, 2) for r in cnn_runs]
mobilenet_acc = [round(r['accuracy']*100, 2) for r in
mobilenet_runs]
cnn_f1 = [round(r['f1_score']*100, 2) for r in cnn_runs]
mobilenet_f1 = [round(r['f1_score']*100, 2) for r in
mobilenet_runs]

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(14, 5))
x = range(len(runs))
width = 0.35

```

```

ax1.bar([i - width/2 for i in x], cnn_acc, width, label='CNN
Custom', color='steelblue')
ax1.bar([i + width/2 for i in x], mobilenet_acc, width,
label='MobileNetV2', color='darkorange')
ax1.set_xlabel('Percobaan')
ax1.set_ylabel('Accuracy (%)')
ax1.set_title('Perbandingan Accuracy per Percobaan')
ax1.set_xticks(list(x))
ax1.set_xticklabels(runs)
ax1.set_ylim(min(cnn_acc + mobilenet_acc) - 5, 100)
ax1.legend()
ax1.grid(axis='y', alpha=0.3)

ax2.bar([i - width/2 for i in x], cnn_f1, width, label='CNN
Custom', color='steelblue')
ax2.bar([i + width/2 for i in x], mobilenet_f1, width,
label='MobileNetV2', color='darkorange')
ax2.set_xlabel('Percobaan')
ax2.set_ylabel('F1-Score (%)')
ax2.set_title('Perbandingan F1-Score per Percobaan')
ax2.set_xticks(list(x))
ax2.set_xticklabels(runs)
ax2.set_ylim(min(cnn_f1 + mobilenet_f1) - 5, 100)
ax2.legend()
ax2.grid(axis='y', alpha=0.3)

plt.tight_layout()
plt.savefig('/content/drive/MyDrive/grafik_perbandingan_per_run.
png', dpi=150, bbox_inches='tight')
plt.show()
print('Grafik tersimpan ke Drive.')

# Cell 6a - Grafik Perbandingan Rata-rata Metrik Klasifikasi
# Membandingkan rata-rata Accuracy, Precision, Recall, dan F1-
Score kedua model

klasifikasi = ['Accuracy', 'Precision', 'Recall', 'F1-Score']
cnn_scores = [
    round(cnn_mean['accuracy']*100, 2),
    round(cnn_mean['precision']*100, 2),
    round(cnn_mean['recall']*100, 2),
    round(cnn_mean['f1_score']*100, 2)
]
mobilenet_scores = [
    round(mobilenet_mean['accuracy']*100, 2),
    round(mobilenet_mean['precision']*100, 2),
    round(mobilenet_mean['recall']*100, 2),
    round(mobilenet_mean['f1_score']*100, 2)
]

x = range(len(klasifikasi))
width = 0.35

fig, ax = plt.subplots(figsize=(10, 5))
ax.bar([i - width/2 for i in x], cnn_scores, width, label='CNN
Custom', color='steelblue')
ax.bar([i + width/2 for i in x], mobilenet_scores, width,
label='MobileNetV2', color='darkorange')
ax.set_ylabel('Nilai (%)')

```

```

ax.set_title('Perbandingan Rata-rata Metrik Klasifikasi')
ax.set_xticks(list(x))
ax.set_xticklabels(klasifikasi)
ax.set_ylim(min(cnn_scores + mobilenet_scores) - 5, 100)
ax.legend()
ax.grid(axis='y', alpha=0.3)

plt.tight_layout()
plt.savefig('/content/drive/MyDrive/grafik_metrik_ratarata.png',
            dpi=150, bbox_inches='tight')
plt.show()
print('Grafik Metrik Klasifikasi tersimpan ke Drive.')
# Cell 6b - Grafik Perbandingan Rata-rata Efisiensi Model
# Membandingkan rata-rata Inference Time dan Ukuran Model kedua
model

efisiensi = ['Inference Time (ms)', 'Ukuran Model (MB)']
cnn_eff = [
    round(cnn_mean['inference_time_ms'], 2),
    round(cnn_mean['model_size_mb'], 2)
]
mobilenet_eff = [
    round(mobilenet_mean['inference_time_ms'], 2),
    round(mobilenet_mean['model_size_mb'], 2)
]

x2 = range(len(efisiensi))
width = 0.35

fig, ax = plt.subplots(figsize=(10, 5))
ax.bar([i - width/2 for i in x2], cnn_eff, width, label='CNN
Custom', color='steelblue')
ax.bar([i + width/2 for i in x2], mobilenet_eff, width,
label='MobileNetV2', color='darkorange')
ax.set_ylabel('Nilai')
ax.set_title('Perbandingan Rata-rata Efisiensi Model')
ax.set_xticks(list(x2))
ax.set_xticklabels(efisiensi)
ax.legend()
ax.grid(axis='y', alpha=0.3)

plt.tight_layout()
plt.savefig('/content/drive/MyDrive/grafik_efisiensi_ratarata.png',
            dpi=150, bbox_inches='tight')
plt.show()
print('Grafik Efisiensi Model tersimpan ke Drive.')
# Cell 7 - Pemenang Tiap Metrik dan Selisih
# Menentukan model dengan performa terbaik pada setiap metrik
evaluasi
print('PEMENANG TIAP METRIK')
print('=' * 60)

metrik_label = {
    'accuracy': 'Accuracy',
    'precision': 'Precision',
    'recall': 'Recall',
    'f1_score': 'F1-Score'
}

```

```

for key, label in metrik_label.items():
    cnn_val = round(cnn_mean[key]*100, 2)
    mob_val = round(mobilenet_mean[key]*100, 2)
    selisih = round(abs(cnn_val - mob_val), 2)
    pemenang = 'MobileNetV2' if mob_val >= cnn_val else 'CNN Custom'
    print(f'{label:15} → {pemenang} (selisih {selisih}%)')

# Inference time (lebih kecil lebih baik)
cnn_inf = round(cnn_mean['inference_time_ms'], 2)
mob_inf = round(mobilenet_mean['inference_time_ms'], 2)
selisih_inf = round(abs(cnn_inf - mob_inf), 2)
pemenang_inf = 'CNN Custom' if cnn_inf <= mob_inf else 'MobileNetV2'
print(f'{"Inference Time":15} → {pemenang_inf} (selisih {selisih_inf} ms)')

# Ukuran model (lebih kecil lebih baik)
cnn_size = round(cnn_mean['model_size_mb'], 2)
mob_size = round(mobilenet_mean['model_size_mb'], 2)
selisih_size = round(abs(cnn_size - mob_size), 2)
pemenang_size = 'CNN Custom' if cnn_size <= mob_size else 'MobileNetV2'
print(f'{"Ukuran Model":15} → {pemenang_size} (selisih {selisih_size} MB)')

# Cell 8 - Kesimpulan Analisis
# Merangkum hasil analisis perbandingan kedua model secara keseluruhan
print('KESIMPULAN ANALISIS')
print('=' * 60)
print(f'CNN Custom - Accuracy rata-rata : {round(cnn_mean["accuracy"]*100,2)}% (SD: {round(cnn_std["accuracy"]*100,2)}%)')
print(f'MobileNetV2 - Accuracy rata-rata : {round(mobilenet_mean["accuracy"]*100,2)}% (SD: {round(mobilenet_std["accuracy"]*100,2)}%)')
print()
print(f'CNN Custom - F1-Score rata-rata : {round(cnn_mean["f1_score"]*100,2)}% (SD: {round(cnn_std["f1_score"]*100,2)}%)')
print(f'MobileNetV2 - F1-Score rata-rata : {round(mobilenet_mean["f1_score"]*100,2)}% (SD: {round(mobilenet_std["f1_score"]*100,2)}%)')
print()
print(f'CNN Custom - Inference Time rata-rata : {round(cnn_mean["inference_time_ms"],2)} ms (SD: {round(cnn_std["inference_time_ms"],2)} ms)')
print(f'MobileNetV2 - Inference Time rata-rata : {round(mobilenet_mean["inference_time_ms"],2)} ms (SD: {round(mobilenet_std["inference_time_ms"],2)} ms)')
print()
print(f'CNN Custom - Ukuran Model : {round(cnn_mean["model_size_mb"],2)} MB')
print(f'MobileNetV2 - Ukuran Model : {round(mobilenet_mean["model_size_mb"],2)} MB')
print()
print('KESIMPULAN:')

```

```

print('MobileNetV2 unggul dalam accuracy, precision, recall, F1-
score, dan ukuran model.')
print('CNN Custom unggul dalam inference time.')

# Cell 9 - Menyusun tabel rekapitulasi akhir sebagai dasar
penulisan Bab IV

rekap_final = {
    'Model': ['CNN Custom', 'MobileNetV2'],
    'Accuracy (%)': [round(cnn_mean['accuracy']*100, 2),
round(mobilenet_mean['accuracy']*100, 2)],
    'SD Accuracy (%)': [round(cnn_std['accuracy']*100, 2),
round(mobilenet_std['accuracy']*100, 2)],
    'Precision (%)': [round(cnn_mean['precision']*100, 2),
round(mobilenet_mean['precision']*100, 2)],
    'SD Precision (%)': [round(cnn_std['precision']*100, 2),
round(mobilenet_std['precision']*100, 2)],
    'Recall (%)': [round(cnn_mean['recall']*100, 2),
round(mobilenet_mean['recall']*100, 2)],
    'SD Recall (%)': [round(cnn_std['recall']*100, 2),
round(mobilenet_std['recall']*100, 2)],
    'F1-Score (%)': [round(cnn_mean['f1_score']*100, 2),
round(mobilenet_mean['f1_score']*100, 2)],
    'SD F1-Score (%)': [round(cnn_std['f1_score']*100, 2),
round(mobilenet_std['f1_score']*100, 2)],
    'Inference Time (ms/gambar)':
[round(cnn_mean['inference_time_ms'], 2),
round(mobilenet_mean['inference_time_ms'], 2)],
    'Ukuran Model (MB)': [round(cnn_mean['model_size_mb'], 2),
round(mobilenet_mean['model_size_mb'], 2)]
}

df_rekap_final = pd.DataFrame(rekap_final)
df_rekap_final.index = df_rekap_final.index + 1

print('Tabel Rekapitulasi Final')
print('=' * 80)
display(df_rekap_final)

# Simpan ke Drive
df_rekap_final.to_csv('/content/drive/MyDrive/tabel_rekap_final.
csv', index=True)
df_per_run.to_csv('/content/drive/MyDrive/tabel_per_run.csv',
index=True)
print('Tabel tersimpan ke Drive.')

```