Klasifikasi Pengemudi Terganggu Berdasarkan Citra Menggunakan ResNet-50

Galih Hermawan
Program Studi Teknik Informatika, Fakultas Teknik dan Ilmu Komputer, Universitas Komputer Indonesia
e-mail: galih.hermawan@email.unikom.ac.id

Abstrak

Penelitian ini bertujuan untuk mengevaluasi dan membandingkan efektivitas algoritma ResNet-50 dalam mengklasifikasikan aktivitas pengemudi terganggu pada dua jenis dataset: setir kiri dan setir kanan, menggunakan dataset AUC Driver Distracted yang mencakup sepuluh kelas aktivitas seperti mengemudi normal, menelepon, dan menulis pesan. Model ResNet-50 diterapkan tanpa modifikasi arsitektur atau teknik augmentasi data, dengan pemilihan parameter menggunakan 5-fold cross-validation dan optimasi Stochastic Gradient Descent (SGD). Hasil penelitian menunjukkan perbedaan signifikan dalam kinerja model antara dataset setir kiri dan kanan, dengan akurasi mencapai 84% dan 96%, secara berturut-turut. Perbedaan ini mungkin disebabkan oleh variasi distribusi data dan visual antara kedua dataset. Kesalahan klasifikasi yang terjadi sebagian besar disebabkan oleh keterbatasan gambar statis yang tidak dapat menangkap konteks temporal dari aktivitas pengemudi. Temuan ini menunjukkan potensi aplikasi dan perlunya penelitian lebih lanjut untuk meningkatkan kemampuan deteksi aktivitas pengemudi terganggu dalam konteks dunia nyata.

Kata kunci*—deteksi pengemudi terganggu, ResNet-50, pembelajaran mendalam, pembelajaran mesin, klasifikasi citra*

1. PENDAHULUAN

Gangguan saat mengemudi adalah salah satu faktor utama yang berkontribusi terhadap kecelakaan lalu lintas di seluruh dunia. Gangguan dapat terjadi ketika pengemudi kehilangan fokus pada jalan karena aktivitas lain, seperti menggunakan ponsel, berbicara dengan penumpang, makan, atau merias wajah. Penelitian menunjukkan bahwa gangguan ini dapat sangat meningkatkan risiko kecelakaan, cedera, dan kematian di jalan raya [1].

Organisasi Kesehatan Dunia (WHO) mengatakan bahwa sekitar 1,35 juta orang meninggal setiap tahun akibat kecelakaan lalu lintas di seluruh dunia, dan sebagian besar dari kematian ini disebabkan oleh perilaku tidak aman pengemudi, termasuk pengemudi yang terganggu [2]. Berdasarkan informasi dari Badan Administrasi Keselamatan Lalu Lintas Jalan Raya Nasional atau *National Highway Traffic Safety Administration* (NHTSA) Amerika Serikat, pada tahun 2022 tercatat 3.308 kematian akibat kecelakaan yang melibatkan pengemudi yang terganggu, serta hampir 290.000 orang mengalami cedera. Di antara korban tewas tersebut, lebih dari 20 persen adalah pengguna jalan yang tidak berada di dalam kendaraan, termasuk pejalan kaki dan pesepeda [3]. Menurut laporan Korps Lalu Lintas (Korlantas) Polri, dari Januari hingga 13 September 2022 terjadi 94.600 kecelakaan lalu lintas di Indonesia, dengan 19.054 orang meninggal dunia, naik 3,7% dari periode yang sama tahun sebelumnya. Perhatian pengemudi yang teralihkan, seperti penggunaan ponsel saat berkendara atau berkendara dengan kecepatan tinggi, adalah penyebab utama kecelakaan, menurut Kombes Arman Achdiat, Kasubdit Dikmas Ditkamsel Korlantas Polri [4].

NHTSA mengklasifikasikan tiga jenis gangguan mengemudi [5]: gangguan visual, yang berarti pengemudi harus mengalihkan pandangan mereka dari jalan raya untuk mendapatkan informasi visual; gangguan manual, yang berarti pengemudi harus melepaskan tangannya dari kemudi dan memanipulasi perangkat; dan gangguan kognitif, yang berarti pengemudi harus memikirkan sesuatu selain mengemudi. Dampak gangguan pada pengemudi dipengaruhi oleh jenis gangguan, frekuensi, dan lama tugas. Pengemudi yang sering melakukan tugas yang sama

atau untuk waktu yang lama dapat meningkatkan risiko kecelakaan ke tingkat yang sebanding dengan tugas yang jauh lebih sulit yang dilakukan lebih jarang. Ini terjadi bahkan jika tugas tersebut tidak terlalu mengganggu.

Upaya penanganan dan pencegahan terhadap mengemudi terganggu, meliputi: strategi perilaku (penegakan peraturan, edukasi), strategi lingkungan (infrastruktur jalan raya, rambu-rambu lalu-lintas, area istirahat, elemen jalan), dan strategi kendaraan (antarmuka perangkat sistem dalam kendaraan, *Advanced Driver Assistance Systems* - ADAS). Pengamatan terhadap perilaku pengemudi secara alami atau *Naturalistic Driving Study* (NDS) merupakan salah satu cara yang dapat diimplementasikan dengan baik [6]. Salah satu perangkat yang dapat digunakan untuk pengamatan adalah kamera. Kamera digunakan untuk menganalisis dampak kegiatan sekunder dalam perilaku mengemudi, seperti: menggunakan ponsel ketika mengemudi, mengoperasikan perangkat di dashboard, berbicara dengan penumpang lain, dan lain-lain [7].

Dalam penelitian mereka, Abouelnaga dkk. [8, 9] mengumpulkan data mandiri dari kumpulan gambar pengemudi dengan berbagai aktivitas mengemudi yang dianggap aman dan tidak aman. Dataset AUC *Distracted Driver* yang tersedia umum digunakan dalam penelitian ini. Studi ini menggunakan algoritma genetik untuk memilih kumpulan (*ensemble*) CNN terbaik dan melakukan analisis visual seperti segmentasi kulit, lokalisasi tangan, dan wajah. Akurasi klasifikasi mencapai 90%.

Beberapa penelitan lain telah banyak menggunakan dataset terbukan khususnya setir kiri untuk mendeteksi gangguan saat mengemudi [10]. Studi oleh Nel dan Ngxande [11] serta Aljohani [12] menunjukkan bahwa teknologi seperti jaringan saraf konvolusional, termasuk arsitektur ResNet-50, efektif dalam mendeteksi gangguan berdasarkan postur pengemudi. Namun, penelitian ini cenderung kurang mengeksplorasi performa model pada dataset setir kanan, yang memiliki karakteristik dan tantangan unik, seperti sudut pandang kamera dan kebiasaan pengemudi yang berbeda.

ResNet-50 adalah salah satu arsitektur CNN yang banyak diterapkan dan terbukti efektif dalam berbagai tugas pengenalan gambar [13]. Namun, meskipun ResNet-50 telah digunakan secara luas, kurangnya penelitian yang membandingkan kinerjanya antara dataset setir kiri dan kanan menunjukkan adanya celah yang perlu dieksplorasi lebih lanjut. Penelitian ini bertujuan untuk mengisi celah tersebut dengan melakukan studi komparatif pada kinerja ResNet-50 dalam klasifikasi pengemudi terganggu menggunakan dataset setir kiri dan kanan. Dengan menganalisis performa model pada kedua dataset ini, diharapkan dapat diperoleh wawasan tambahan mengenai efektivitas ResNet-50 dalam mendeteksi gangguan pengemudi serta memberikan kontribusi terhadap pengembangan sistem deteksi berbasis citra yang lebih adaptif.

2. METODE PENELITIAN

Dalam bagian ini, dijelaskan secara rinci metode yang digunakan untuk mencapai tujuan penelitian. Proses penelitian ini melibatkan beberapa langkah penting mulai dari pemilihan dataset, pra-pemrosesan data, penerapan model arsitektur jaringan saraf konvolusional, hingga analisis hasil eksperimen. Masing-masing langkah ini akan dibahas secara sistematis untuk memberikan pemahaman yang jelas mengenai pendekatan dan teknik yang diterapkan dalam penelitian ini.

Dataset

Dalam penelitian ini, dataset yang digunakan adalah AUC *Distracted Driver* yang dipublikasikan oleh Abouelnaga dkk. [8, 9]. Dataset ini terdiri dari 14.478 bingkai gambar yang dikumpulkan dari 44 peserta pengemudi dari berbagai negara, menggunakan 5 jenis kendaraan yang berbeda. Dataset ini terbagi menjadi dua kategori berdasarkan posisi setir, yaitu setir kiri dan setir kanan. Data setir kiri memiliki resolusi 1920x1080 piksel, sedangkan data setir kanan memiliki resolusi 640x480 piksel. Dataset ini mencakup 10 kelas aktivitas mengemudi, meliputi c0 (mengemudi normal), c1 (menulis pesan dengan tangan kanan), c2 (menelepon dengan

tangan kanan), c3 (menulis pesan dengan tangan kiri), c4 (menelepon dengan tangan kiri), c5 (mengoperasikan perangkat di dashboard), c6 (minum dengan memegang gelas), c7 (menjangkau sesuatu di belakang kemudi hingga balik badan), c8 (membetulkan rambut atau merias wajah), dan c9 (berbicara kepada penumpang lain). Distribusi dataset dapat dilihat pada Tabel 1.

Tabel 1. Distribusi kelas dataset pengemudi terganggu

Kelas	Setir Kiri			Setir Kanan		
	Latiah	*Uji*	*Jumlah*	*Latiah*	*Uji*	*Jumlah*
c0	2440	266	2706	200	80	280
c1	1305	133	1438	200	80	280
c2	862	114	976	200	80	280
c3	744	100	844	200	80	280
c4	950	90	1040	200	80	280
c5	753	90	843	200	80	280
c6	733	63	796	200	80	280
c7	691	63	754	200	80	280
c8	698	66	764	200	80	280
c9	1379	138	1517	200	80	280
Total	10555	1123	11678	2000	800	2800

Tabel 1 menunjukkan distribusi kelas pada dataset pengemudi terganggu. Terlihat bahwa kelas "mengemudi normal" memiliki jumlah sampel paling banyak, mengindikasikan adanya ketidakseimbangan data. Perbedaan resolusi gambar antara data setir kiri dan kanan juga perlu diperhatikan, karena dapat memengaruhi kinerja model pengolahan citra.

Contoh gambar yang mewakili kesepuluh kelas aktivitas mengemudi dapat dilihat pada Gambar 1 (setir kiri) dan Gambar 2 (setir kanan). Gambar-gambar ini menunjukkan variasi pose dan objek yang terkait dengan setiap kelas aktivitas.

Gambar 1. Contoh variasi kelas aktivitas mengemudi (setir kiri)

Gambar 2. Contoh variasi kelas aktivitas mengemudi (setir kanan)

Pra-pemrosesan Data

Untuk mempersiapkan dataset gambar sebelum pelatihan model, dilakukan pra-pemrosesan yang meliputi *resizing* (mengubah ukuran gambar) dan normalisasi. Sebelum dilakukan pelatihan model, dataset gambar mengalami pra-pemrosesan. Tahap pra-pemrosesan meliputi resizing gambar menjadi ukuran 400x400 piksel dan normalisasi nilai piksel ke rentang 0-1. *Resizing* dilakukan untuk memastikan semua gambar memiliki dimensi yang sama, sedangkan normalisasi bertujuan untuk menstandarkan distribusi intensitas piksel. Meskipun teknik augmentasi data lainnya dapat meningkatkan kinerja model, dalam penelitian ini difokuskan pada efektivitas *resizing* dan normalisasi untuk mengevaluasi kinerja model dasar.

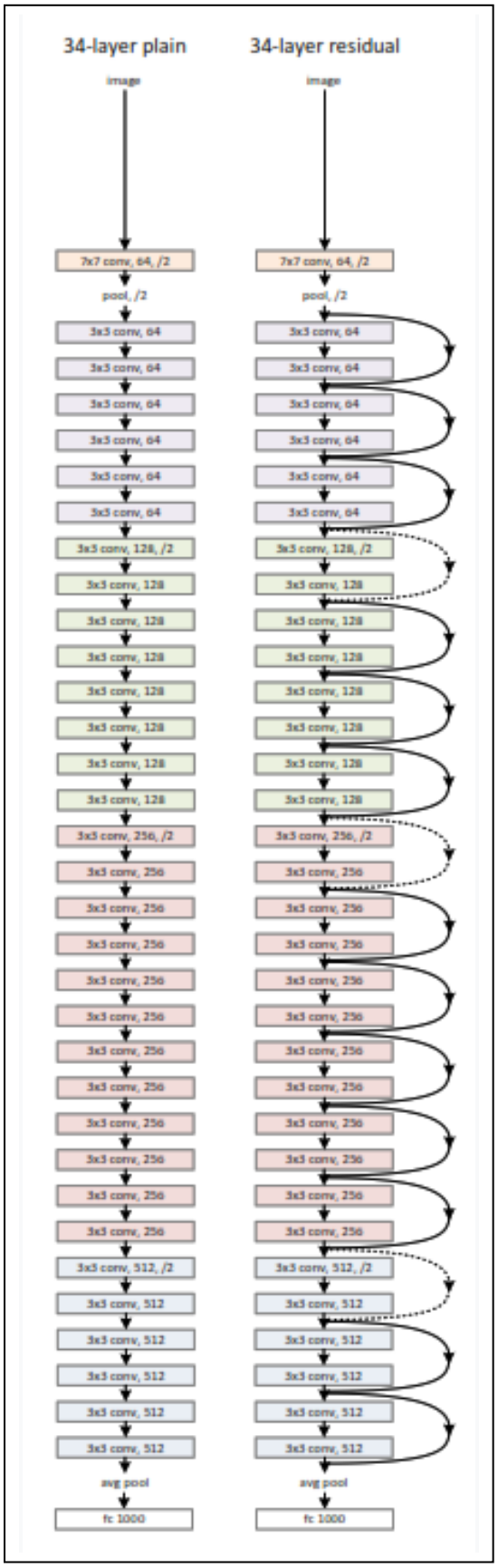
Arsitektur Model

Pada penelitian ini, arsitektur ResNet-50 digunakan sebagai tulang punggung model klasifikasi. ResNet-50, yang merupakan bagian dari keluarga ResNet, telah terbukti sangat efektif dalam berbagai tugas pengenalan gambar, termasuk klasifikasi gambar skala besar.

Meskipun makalah asli ResNet lebih sering menampilkan visualisasi ResNet-34, ResNet-50 memiliki struktur yang serupa namun dengan jumlah lapisan yang lebih dalam (50 lapisan). Jumlah lapisan yang lebih dalam ini memungkinkan ResNet-50 untuk menangkap fitur-fitur yang lebih kompleks dan hierarkis pada gambar, sehingga meningkatkan kemampuannya dalam tugas klasifikasi.

ResNet-50 dirancang untuk mengatasi masalah vanishing gradient yang sering terjadi pada jaringan saraf dalam atau *Deep Neural Network* (DNN). Hal ini dilakukan dengan mengadopsi konsep *residual learning*. Dalam setiap blok residual, identitas dari lapisan masukan akan ditambahkan ke lapisan luaran, sehingga memungkinkan informasi dari lapisan sebelumnya untuk mengalir secara langsung ke lapisan berikutnya [14].

Konsep blok residual diilustrasikan pada Gambar 3. Setiap blok residual terdiri dari beberapa lapisan konvolusi, *batch normalization*, dan fungsi aktivasi ReLU. Identitas dari lapisan masukan akan ditambahkan ke lapisan luaran melalui koneksi *shortcut*. Perbedaan utama antara ResNet-34 dan ResNet-50 terletak pada jumlah blok residualnya. Gambar 3 menunjukkan bahwa ResNet-34 memiliki jumlah blok yang lebih sedikit dibandingkan dengan ResNet-50. Jumlah blok yang lebih banyak pada ResNet-50 memungkinkan model untuk mengekstrak fitur yang lebih kompleks.



Gambar 3. Arsitektur ResNet-34 [14]

Untuk implementasi, pustaka PyTorch digunakan. Bobot parameter yang sudah dilatih atau *pre-trained weights* dari ResNet-50 dimuat menggunakan ResNet50_Weights.DEFAULT, yang telah dilatih pada dataset ImageNet yang besar dan beragam. *Pre-trained weights* ini memberikan inisialisasi yang baik pada model, sehingga proses pelatihan dipercepat dan akurasi meningkat.

Agar model dapat melakukan klasifikasi pada dataset yang memiliki 10 kelas, lapisan *fully-connected* (fc) pada model ResNet-50 dimodifikasi. Lapisan fc ini bertanggung jawab untuk menghasilkan luaran akhir berupa probabilitas untuk setiap kelas. Jumlah neuron pada lapisan fc diganti menjadi 10 sesuai dengan jumlah kelas dalam dataset.

Pengaturan Pelatihan

Proses pelatihan model klasifikasi gambar ini dilakukan menggunakan kerangka kerja deep learning PyTorch pada platform komputasi Kaggle. Dua unit GPU NVIDIA T4 digunakan dengan total memori 30GB untuk mempercepat proses komputasi. GPU memungkinkan paralelisme tingkat tinggi dalam komputasi matriks besar, yang sangat berguna dalam pelatihan jaringan saraf dalam. Konfigurasi perangkat keras ini memungkinkan untuk melatih model dengan ukuran batch yang lebih besar dan melakukan lebih banyak eksperimen dalam waktu yang lebih singkat. Penggunaan GPU juga mendukung pelaksanaan berbagai eksperimen dengan cepat, termasuk uji coba berbagai kombinasi *hyperparameter* dan ukuran *batch* yang berbeda. Ini penting untuk evaluasi yang lebih komprehensif dan optimasi model.

Penelitian ini mengoptimalkan *hyperparameter* model deteksi pengemudi terganggu untuk mencapai keseimbangan antara akurasi dan efisiensi pelatihan. Ukuran batch 16 dan 32 diuji dengan *K-Fold Cross Validation* untuk menilai dampaknya pada kinerja model. *Stochastic Gradient Descent* (SGD) dengan momentum 0,9 dipilih sebagai *optimizer* karena kesederhanaannya dan kemampuannya mempercepat konvergensi serta mengurangi *overfitting*. Dua nilai *learning rate*, 0,001 dan 0,01, diuji untuk mengontrol langkah pembaruan bobot, dengan 0,001 terpilih sebagai yang optimal. Regularisasi L2 dengan *weight decay* 0,0001 diterapkan untuk mencegah *overfitting,* dan *Cross-Entropy Loss* digunakan sebagai fungsi kerugian untuk klasifikasi multi-kelas. Pelatihan dilakukan dengan 5, 10, dan 15 *epoch* untuk menemukan keseimbangan terbaik antara *underfitting* dan *overfitting*.

Strategi Evaluasi

Untuk mengevaluasi performa model, diterapkan strategi 5-fold cross-validation. Dalam proses ini, dataset dibagi menjadi lima bagian (*folds*) yang sama besar. Setiap bagian berfungsi sebagai data validasi sementara empat bagian lainnya digunakan untuk pelatihan. Proses ini diulang sebanyak lima kali, dengan setiap bagian bergantian sebagai data validasi.

Dalam uji coba *hyperparameter*, dilakukan evaluasi berbagai kombinasi parameter, termasuk ukuran *batch*, *learning rate*, dan jumlah *epoch*. Setiap kombinasi diuji untuk menentukan parameter yang memberikan performa terbaik. Parameter dengan performa terbaik, berdasarkan hasil evaluasi, digunakan untuk melatih model pada dataset yang terdiri dari gambar setir kanan dan setir kiri.

Data pelatihan dibagi menjadi 80% untuk pelatihan dan 20% untuk validasi, memungkinkan untuk menilai akurasi model secara lebih akurat selama proses pelatihan. Evaluasi dilakukan menggunakan metrik seperti akurasi, matriks confusion, dan laporan klasifikasi untuk mendapatkan gambaran menyeluruh tentang performa model.

Laporan klasifikasi disajikan menggunakan berbagai metrik evaluasi, termasuk akurasi keseluruhan dan metrik lainnya seperti *precision*, *recall*, *F1-score*, dan *support*. *Precision* mengukur proporsi prediksi positif yang benar, di mana nilai yang lebih tinggi menunjukkan lebih sedikit *false positive*. *Recall* mengukur proporsi sampel positif yang diidentifikasi dengan benar oleh model, dengan nilai lebih tinggi menunjukkan lebih sedikit *false negative*. *F1-score* merupakan rata-rata harmonik dari *precision* dan *recall*, memberikan keseimbangan antara keduanya; nilai *F1-score* yang tinggi menunjukkan bahwa model memiliki *precision* dan *recall*

yang baik. *Support* menunjukkan jumlah sampel aktual untuk setiap kelas dalam dataset. Setelah pelatihan, model diuji pada dataset terpisah untuk menilai kemampuannya dalam menggeneralisasi pada data yang belum pernah dilihat sebelumnya.

3. HASIL DAN PEMBAHASAN

Bab ini menyajikan hasil eksperimen dan analisis mendalam terhadap kinerja model klasifikasi dalam mendeteksi pengemudi terganggu pada kondisi setir kiri dan kanan. Model klasifikasi yang digunakan adalah ResNet-50 yang telah dilatih dengan berbagai parameter hyperparameter yang diujicobakan.

Hasil Eksperimen

Pada bagian ini, disajikan hasil eksperimen yang dilakukan untuk mengevaluasi performa model klasifikasi gambar dalam mendeteksi pengemudi terganggu. Berikut adalah ringkasan pengaturan eksperimen yang digunakan dalam studi ini:

1. Pengaturan Eksperimen

Dalam penelitian ini, berbagai *hyperparameter* diuji untuk mengevaluasi pengaruhnya terhadap performa model. Ukuran *batch* yang diuji adalah 16 dan 32 gambar per iterasi, sedangkan learning rate diuji pada 0,001 dan 0,01 untuk menentukan kecepatan pembelajaran optimal. Eksperimen juga dilakukan dengan epoch sebanyak 5, 10, dan 15 untuk mengamati konvergensi model. Model dilatih menggunakan kerangka kerja PyTorch di platform Kaggle dengan dua unit GPU NVIDIA T4, yang memungkinkan pelatihan dengan ukuran *batch* lebih besar dan eksperimen lebih efisien. Evaluasi model dilakukan menggunakan metode *k-fold cross-validation* dengan 5 bagian, di mana dataset dibagi menjadi 80% data latih dan 20% data validasi untuk memastikan hasil yang andal. Kinerja terbaik dicatat berdasarkan akurasi dan nilai kerugian (*loss*) validasi, dan konfigurasi parameter terbaik digunakan untuk pelatihan lebih lanjut. Hasil eksperimen divisualisasikan melalui grafik akurasi dan *loss* validasi per *epoch*, serta matriks confusion dan laporan klasifikasi untuk dataset setir kanan dan kiri. Pendekatan ini memungkinkan analisis mendalam terhadap dampak *hyperparameter* dan memastikan bahwa hasil penelitian dapat diandalkan dan representatif dalam klasifikasi pengemudi terganggu.

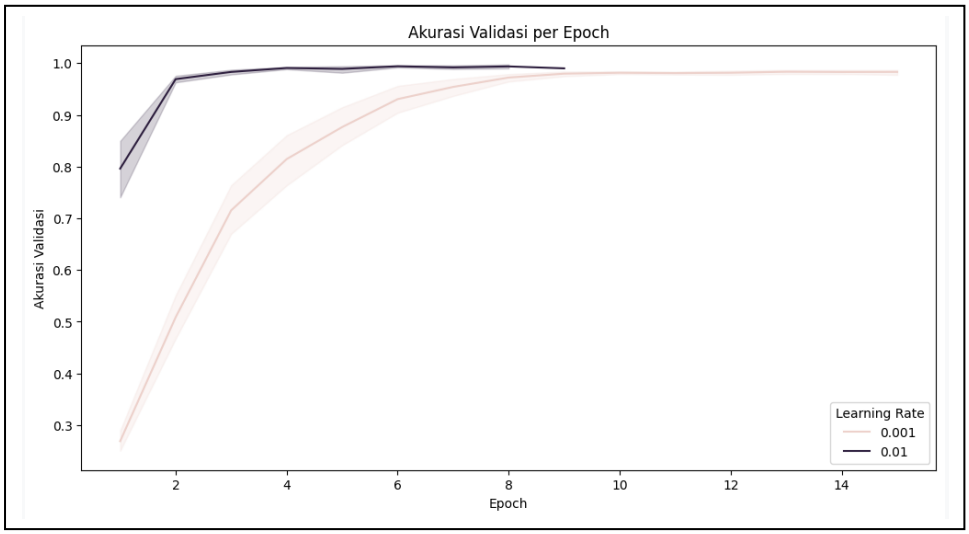
2. Top 3 Performa Berdasarkan Akurasi Validasi

Tabel 2 menyajikan tiga konfigurasi parameter dengan performa tertinggi berdasarkan akurasi validasi. Hasil ini menunjukkan parameter yang memberikan akurasi tertinggi selama proses *k-fold cross-validation*.

Tabel 2. Top 3 performa berdasarkan akurasi validasi

Indeks	Fold	Learning Rate	Bactch	Epoch	Akurasi
149	2	0,01	16	15	0,9975
150	2	0,01	16	15	0,9975
154	2	0,01	32	5	0,9975

Grafik pada Gambar 4 menunjukkan perubahan akurasi validasi pada setiap *epoch* untuk berbagai konfigurasi parameter, termasuk *learning rate* dan ukuran *batch*. Grafik ini memberikan gambaran visual tentang bagaimana akurasi validasi berkembang selama proses pelatihan untuk masing-masing kombinasi parameter yang diuji.



Gambar 4. Grafik nilai akurasi per *epoch*

Dalam Gambar 4 dapat dilihat hasil akurasi tertinggi berdasarkan validasi menunjukkan bahwa konfigurasi dengan learning rate 0,01 dan ukuran *batch* 16 memberikan hasil yang konsisten dengan akurasi tertinggi, mencapai 0,9975 pada beberapa bagian (*fold*). Konfigurasi ini menunjukkan bahwa ukuran *batch* 16, yang lebih kecil, mampu mencapai akurasi tinggi dengan stabil, sementara ukuran *batch* 32 juga menunjukkan performa yang baik namun pada *epoch* yang berbeda. Hasil ini menunjukkan bahwa parameter *learning rate* 0,01 sangat efektif dalam memaksimalkan akurasi model, dan ukuran *batch* 16 memberikan keseimbangan yang baik antara kecepatan pelatihan dan akurasi. Ini mungkin disebabkan oleh fakta bahwa ukuran *batch* yang lebih kecil memberikan pembaruan bobot yang lebih sering, memungkinkan model untuk belajar dengan lebih detail dari data.

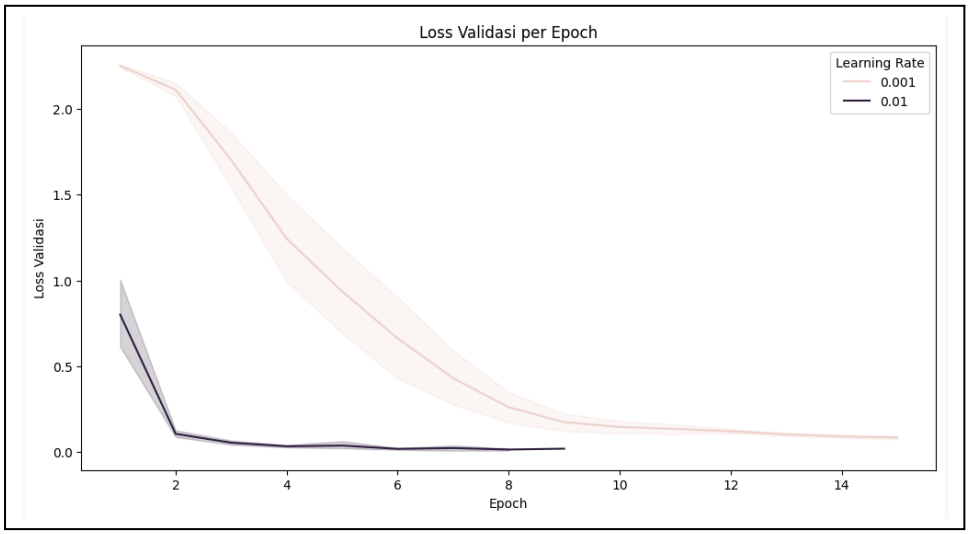
3. Top 3 Performa Berdasarkan Nilai Kerugian (Loss) Validasi

Tabel 3 berikut menyajikan tiga konfigurasi parameter dengan nilai kerugian (*loss*) validasi terendah. Hasil ini menunjukkan parameter yang memberikan nilai *loss* validasi paling rendah selama proses *k-fold cross-validation*.

Tabel 3. Top 3 performa berdasarkan *loss* validasi

Indeks	Fold	Learning Rate	Bactch	Epoch	Akurasi
150	2	0,01	16	15	0,00233
149	2	0,01	16	15	0,00240
227	3	0,01	16	10	0,00777

Grafik pada Gambar 5 menunjukkan perubahan nilai kerugian (*loss*) validasi pada setiap *epoch* untuk berbagai konfigurasi parameter. Grafik ini memberikan gambaran visual tentang bagaimana *loss* validasi berkembang selama proses pelatihan untuk masing-masing kombinasi parameter yang diuji.



Gambar 5. Grafik nilai kerugian (*loss*) per *epoch*

Gambar 5 menunjukkan bahwa konfigurasi dengan *learning rate* 0,01 dan ukuran *batch* 16 menghasilkan nilai *loss* validasi terendah, yaitu 0,00233 pada salah satu bagian (*fold*). Konfigurasi ini terbukti efektif dalam mengurangi kerugian model selama pelatihan dibandingkan dengan ukuran *batch* 32 dan berbagai *epoch*. Hasil ini menegaskan bahwa parameter *learning rate* 0,01 dan ukuran *batch* 16 tidak hanya meningkatkan akurasi, tetapi juga meminimalkan *loss* model. Konfigurasi ini juga menunjukkan kestabilan yang baik dalam *loss* validasi, yang merupakan indikator penting kemampuan model untuk melakukan generalisasi pada data baru.

4. Parameter Terbaik Berdasarkan Nilai Rata-Rata

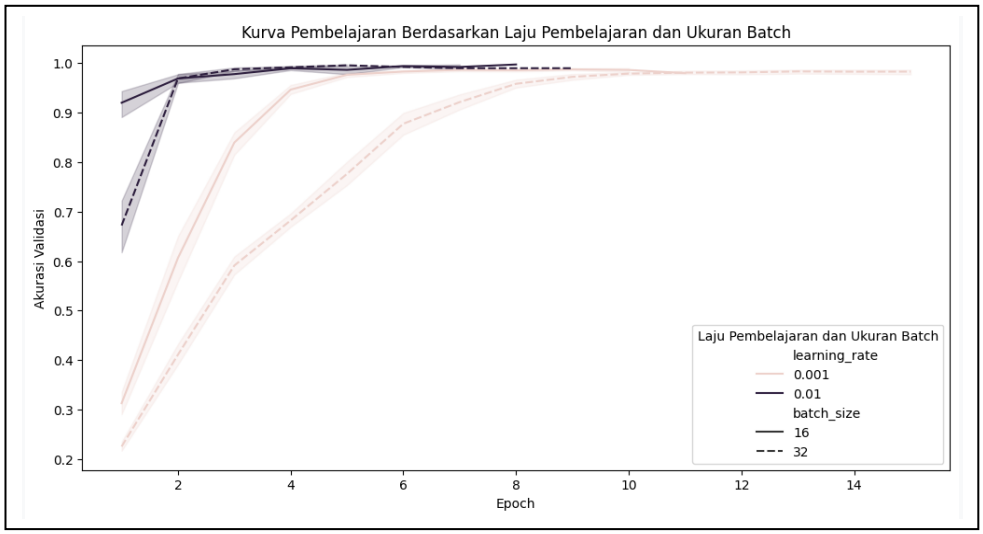
Tabel 4 berikut menyajikan parameter dengan performa rata-rata terbaik berdasarkan nilai rata-rata *loss* dan akurasi selama proses *k-fold cross-validation*.

Tabel 4. Parameter terbaik rata-rata berdasarkan nilai *loss* dan akurasi validasi

Kategori	Learning Rate	Bactch	Epoch	Akurasi	Loss
Akurasi	0,01	16	10	0,97194	0,09646
Loss	0,01	16	15	0,97150	0,09640

Berdasarkan Tabel 4, kombinasi parameter terbaik untuk model diperoleh dengan learning rate 0,01 dan ukuran *batch* 16, namun jumlah *epoch* yang optimal bervariasi bergantung pada metrik yang dioptimalkan. Untuk akurasi tertinggi, konfigurasi dengan 10 *epoch* menghasilkan rata-rata akurasi validasi sebesar 0,97194 dan nilai loss 0,09646. Sementara itu, untuk nilai *loss* terendah, konfigurasi dengan 15 *epoch* memberikan akurasi rata-rata 0,97150 dan *loss* 0,09640. Ini menunjukkan bahwa meskipun kedua konfigurasi memiliki performa yang hampir serupa, penggunaan 10 *epoch* lebih optimal jika prioritasnya adalah akurasi, sedangkan 15 *epoch* lebih sesuai untuk meminimalkan loss.

Grafik pada Gambar 6 menunjukkan kurva pembelajaran yang mengilustrasikan perubahan akurasi validasi berdasarkan jumlah *epoch*, laju pembelajaran (*learning rate*), dan ukuran *batch* (*batch size*). Grafik tersebut memberikan wawasan visual tentang bagaimana akurasi validasi berfluktuasi seiring dengan perubahan parameter, serta perbandingan antara konfigurasi parameter yang berbeda.

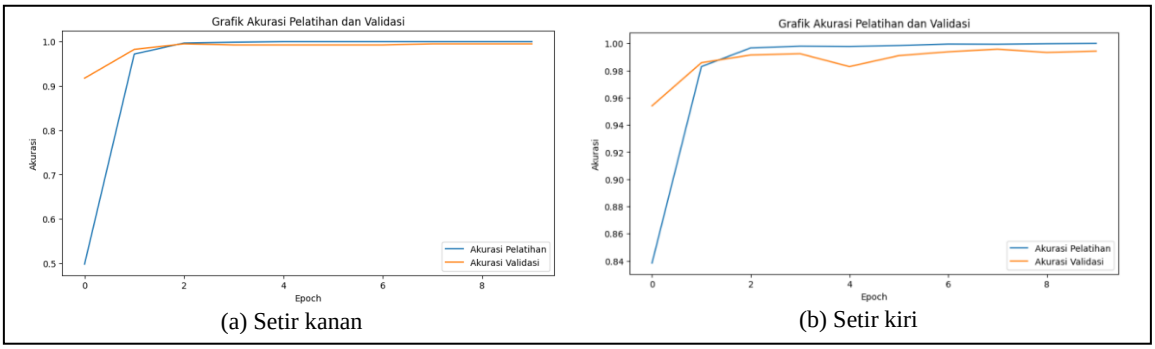


Gambar 6. Kurva pembelajaran berdasarkan laju pembelajaran dan ukuran *batch*

Berdasarkan grafik di Gambar 6, konfigurasi dengan *learning rate* 0,01 dan ukuran *batch* 16 menunjukkan performa rata-rata terbaik untuk kedua metrik, yaitu akurasi dan *loss*. Parameter ini lebih stabil dan efektif dibandingkan konfigurasi lainnya, baik dalam hal akurasi validasi maupun nilai *loss*. Hasil ini mengindikasikan bahwa parameter optimal untuk model adalah *learning rate* 0,01 dan ukuran *batch* 16, dengan *epoch* bervariasi antara 10 dan 15 tergantung pada metrik yang dioptimalkan. Kurva pembelajaran menunjukkan bahwa kombinasi ini konsisten memberikan hasil yang baik, sehingga mendukung pemilihan parameter ini untuk pelatihan akhir model.

5. Hasil Pelatihan dan Validasi pada Dataset Setir Kanan dan Kiri

Pada bagian ini, disajikan hasil pelatihan dan validasi model menggunakan parameter terbaik yang telah ditentukan melalui eksperimen sebelumnya. Parameter terbaik yang digunakan adalah *learning rate* 0,01, ukuran *batch* 16, dan *epoch* 10. Model dilatih dan divalidasi pada dua dataset yang berbeda, yaitu dataset setir kanan dan setir kiri. Setiap dataset dilatih menggunakan parameter yang sama untuk memastikan perbandingan yang adil dan untuk mengevaluasi performa model secara konsisten pada berbagai jenis data.

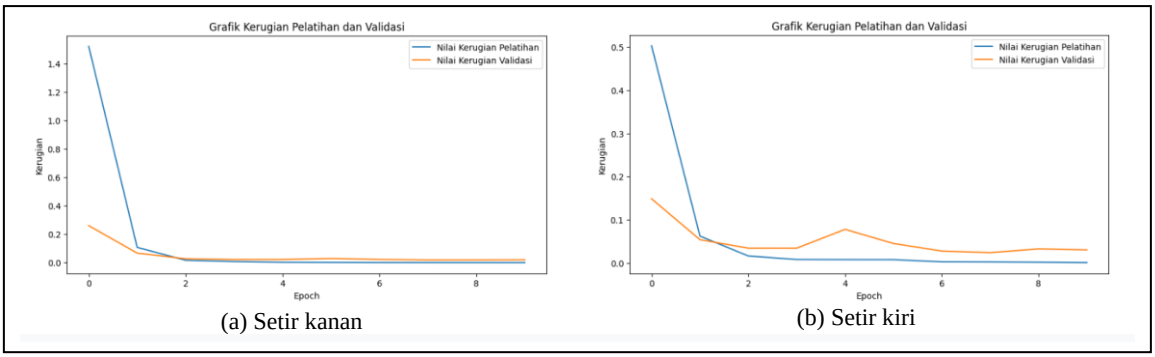


(a) Setir kanan (b) Setir kiri

Gambar 7. Grafik akurasi pelatihan dan validasi pada dataset setir kanan dan setir kiri

Gambar 7 menunjukkan perbandingan akurasi pelatihan dan validasi antara model yang dilatih pada dataset setir kanan dan kiri. Kedua model mencapai akurasi validasi di atas 90%, menunjukkan kinerja yang baik. Model untuk dataset setir kanan mencapai akurasi maksimum 99% pada epoch ke-2, sementara model untuk dataset setir kiri mencapai 98% pada epoch ke-7. Perbedaan kecil ini mungkin disebabkan oleh variasi jumlah data pelatihan atau kompleksitas fitur yang perlu dipelajari oleh model.

Grafik pada Gambar 8 menunjukkan kerugian atau *loss* validasi model pada dataset setir kanan dan kiri selama proses pelatihan. Setiap kurva mewakili perubahan loss validasi sepanjang epoch untuk kedua dataset.



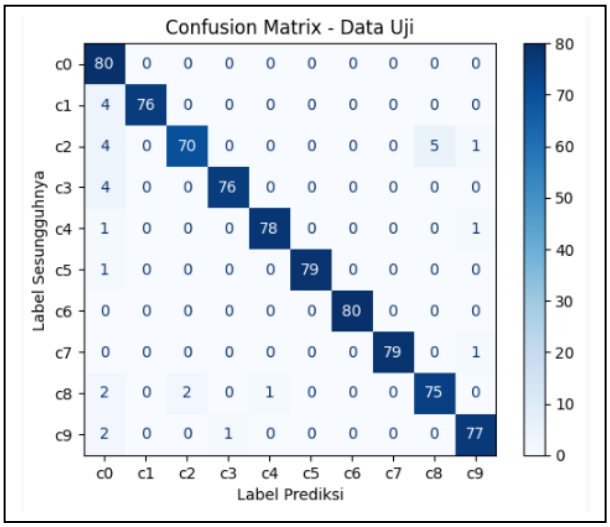
(a) Setir kanan (b) Setir kiri

Gambar 8. Grafik loss pelatihan dan validasi pada dataset setir kanan dan setir kiri

Berdasarkan Gambar 8, model untuk dataset setir kiri menunjukkan penurunan tajam pada nilai kerugian pelatihan di *epoch* pertama, lalu mendatar, sedangkan nilai kerugian validasi menurun lebih lambat namun stabil. Ini menunjukkan bahwa model belajar cepat dari data pelatihan dan meningkatkan performanya pada data validasi, meskipun dengan laju yang lebih lambat. Model untuk dataset setir kanan juga menunjukkan penurunan tajam pada nilai kerugian pelatihan di beberapa *epoch* pertama sebelum mendatar. Nilai kerugian validasi model ini juga menurun lebih lambat namun tetap meningkat. Secara keseluruhan, kedua grafik menunjukkan bahwa model dapat belajar dengan baik dari data pelatihan dan meningkatkan performa pada data validasi tanpa tanda-tanda *overfitting* atau *underfitting* yang signifikan.

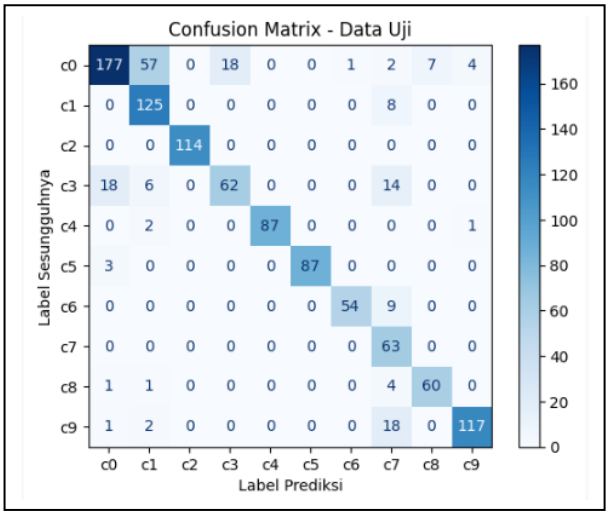
6. Hasil Pengujian pada Dataset Setir Kanan dan Kiri

Bagian ini menyajikan hasil dari proses pengujian akhir model pada dataset setir kanan dan setir kiri menggunakan parameter terbaik yang telah ditentukan. Evaluasi dilakukan untuk mengukur kemampuan model dalam mengklasifikasikan gambar berdasarkan gangguan yang ada. Matriks Confusion memberikan gambaran visual tentang jumlah prediksi yang benar dan salah untuk setiap kelas dalam dataset. Ini membantu dalam memahami bagaimana model melakukan klasifikasi di setiap kategori dan di mana terjadi kesalahan. Berikut ini gambar matriks confusion untuk hasil uji dataset setir kanan.



Gambar 9. Matriks confusion hasil uji dataset setir kanan

Matriks confusion di Gambar 9 menunjukkan performa model klasifikasi menggunakan dataset AUC distracted driver pada kendaraan setir kanan dengan 10 kelas aktivitas pengemudi. Dari 10 kelas, model memiliki performa yang baik dengan prediksi yang akurat di sebagian besar kelas. Kelas c0 hingga c9 memiliki tingkat akurasi prediksi yang tinggi, terutama kelas c0, c6, dan c9, dengan lebih dari 75 sampel yang berhasil diklasifikasikan dengan benar. Terdapat beberapa kesalahan klasifikasi pada kelas c1, c2, dan c8, di mana prediksi salah tersebar ke beberapa kelas lain, seperti kelas c8 yang memiliki 5 kesalahan klasifikasi ke kelas c2. Secara keseluruhan, model mampu memprediksi dengan baik sebagian besar kelas, meskipun terdapat beberapa kesalahan minor pada beberapa kelas tertentu.



Gambar 10. Matriks confusion hasil uji dataset setir kiri

Berdasarkan matriks pada Gambar 10, performa klasifikasi model bervariasi di setiap kelas. Kelas c0 dan c9 menunjukkan hasil yang baik dengan masing-masing 177 dan 117 prediksi benar dari total 266 dan 137 sampel, meskipun terdapat beberapa kesalahan. Kesalahan signifikan terjadi pada kelas c0, di mana 57 sampel diklasifikasikan salah sebagai c1 dan 18 sampel sebagai c2, serta kelas c3 dan c7 yang juga menunjukkan kesalahan serupa. Kesalahan cenderung terkonsentrasi di kelas-kelas tertentu seperti c0, c1, c2, serta c3, c5, dan c7, yang mungkin disebabkan oleh kemiripan fitur visual di antara kelas-kelas tersebut.

7. Laporan Klasifikasi untuk Dataset Setir Kanan dan Kiri

Adapun laporan klasifikasi pengujian dapat dilihat pada Gambar 11 untuk dataset setir kanan dan Gambar 12 untuk dataset setir kiri.

	precision	recall	f1-score	support
c0	0.82	1.00	0.90	80
c1	1.00	0.95	0.97	80
c2	0.97	0.88	0.92	80
c3	0.99	0.95	0.97	80
c4	0.99	0.97	0.98	80
c5	1.00	0.99	0.99	80
c6	1.00	1.00	1.00	80
c7	1.00	0.99	0.99	80
c8	0.94	0.94	0.94	80
c9	0.96	0.96	0.96	80
accuracy			0.96	800
macro avg	0.97	0.96	0.96	800
weighted avg	0.97	0.96	0.96	800

Gambar 11. Laporan klasifikasi hasil uji dataset setir kanan

Hasil pengujian model ResNet-50 pada dataset setir kanan di Gambar 11 menunjukkan performa yang sangat baik, dengan *precision* sempurna (1,00) di kelas c1, c5, c6, dan c7, serta *recall* di atas 0,90 untuk semua kelas kecuali c2 (0,88). *F1-Score* konsisten tinggi di semua kelas, minimal 0,90. Akurasi keseluruhan mencapai 0,96, dengan rata-rata makro dan tertimbang 0,97, mengindikasikan performa yang konsisten dan optimal di seluruh kelas.

	precision	recall	f1-score	support
c0	0.89	0.67	0.76	266
c1	0.65	0.94	0.77	133
c2	1.00	1.00	1.00	114
c3	0.78	0.62	0.69	100
c4	1.00	0.97	0.98	90
c5	1.00	0.97	0.98	90
c6	0.98	0.86	0.92	63
c7	0.53	1.00	0.70	63
c8	0.90	0.91	0.90	66
c9	0.96	0.85	0.90	138
accuracy			0.84	1123
macro avg	0.87	0.88	0.86	1123
weighted avg	0.87	0.84	0.85	1123

Gambar 12. Laporan klasifikasi hasil uji dataset setir kiri

Berdasarkan uji dataset setir kiri di Gambar 12, performa model ResNet-50 cukup tinggi, dengan *precision* tertinggi (1,00) di kelas c2, c4, dan c5, yang menunjukkan semua prediksi positif di kelas tersebut benar. *Recall* juga tinggi di kelas c1, c2, c4, c5, dan c7, menandakan model mampu mendeteksi sebagian besar contoh positif. *F1-Score* berkisar antara 0,69 (kelas c3 dan c7) hingga 1.00 (kelas c2), mencerminkan keseimbangan antara *precision* dan *recall*. Akurasi keseluruhan mencapai 0,84 dengan rata-rata makro dan tertimbang 0,87, menunjukkan performa model yang baik meski ada beberapa kelas dengan kinerja lebih rendah.

Pembahasan

Dalam analisis performa model, ukuran batch 16 dan learning rate 0,01 terbukti memberikan hasil yang konsisten dalam hal akurasi dan *loss*. Ukuran *batch* yang lebih kecil cenderung menghasilkan hasil yang lebih stabil namun dengan lebih banyak *noise*, sementara *learning rate* 0,01 menawarkan keseimbangan yang baik antara kecepatan konvergensi dan stabilitas model. Selain itu, model menunjukkan performa lebih baik pada dataset setir kanan, yang bisa disebabkan oleh keseragaman data atau perbedaan distribusi kelas. Kelas dengan pose yang lebih statis, seperti menelepon dengan tangan kanan dan mengoperasikan perangkat di *dashboard*, lebih mudah dikenali dibandingkan dengan pose dinamis seperti menjangkau sesuatu di belakang kemudi.

Matriks confusion dan laporan klasifikasi mengungkapkan tantangan dalam membedakan beberapa kelas pada dataset setir kiri, seperti menulis pesan dengan tangan kanan dan kiri yang memiliki kemiripan pose. Kelas menjangkau sesuatu di belakang kemudi menunjukkan *recall* tinggi tetapi *precision* rendah, mungkin karena variasi postur dan gerakan yang sering tumpang tindih dengan aktivitas lain. Sebaliknya, kelas seperti menelepon dengan tangan kanan dan mengoperasikan perangkat di *dashboard* memperoleh nilai *precision* dan *recall* tinggi karena pose yang lebih statis dan konsisten. Pada dataset setir kanan, model lebih akurat dalam mendeteksi aktivitas tangan kanan dan postur yang stabil, kemungkinan disebabkan oleh perspektif visual yang lebih baik dari posisi kamera atau orientasi setir kanan.

Perbedaan kinerja antara penelitian ini dan studi lain dapat diatributkan pada beberapa faktor. Penelitian Eraqi dkk. [9] mencapai akurasi 90% dengan menggunakan *ensemble* CNN dan algoritma genetik, sedangkan penelitian ini menunjukkan akurasi 84% pada dataset setir kiri dan 96% pada dataset setir kanan. Kinerja yang lebih tinggi pada dataset setir kanan dapat dikaitkan dengan kualitas dataset yang lebih baik dan keberagaman data yang lebih besar. Di sisi lain, hasil penelitian Nel dan Ngxande [11] menunjukkan akurasi 92.9% untuk dataset State Farm (Kaggle), yang juga mendemonstrasikan kinerja yang solid dengan model ResNet-200 dan dataset yang lebih besar.

Pada dataset setir kiri, kelas c7 (menjangkau sesuatu di belakang kemudi hingga balik badan) menunjukkan performa yang lebih rendah dalam hal recall dan *F1-Score*, mencerminkan tantangan dalam mendeteksi distraksi yang melibatkan perubahan postur yang lebih kompleks atau pergerakan tangan yang tidak terduga. Ini serupa dengan temuan Eraqi dkk. [9] yang mencatat kesulitan dalam menangani variasi postur pengemudi.

Penelitian ini tidak menggunakan teknik augmentasi data, yang bisa menjadi faktor penting dalam meningkatkan kinerja model, terutama pada dataset yang terbatas. Studi Eraqi dkk. [9] dan Nel dan Ngxande [11] tidak mencantumkan teknik augmentasi dalam laporan mereka, tetapi secara umum, augmentasi data dapat membantu model menjadi lebih andal terhadap variasi dalam data dan meningkatkan generalisasi model. Oleh karena itu, penerapan teknik augmentasi dapat berpotensi memperbaiki kinerja deteksi, terutama dalam menghadapi data yang tidak seimbang atau terbatas.

Keterbatasan utama dalam penelitian ini meliputi keterbatasan dataset, terutama pada dataset setir kiri, yang dapat membatasi kemampuan model untuk menggeneralisasi dengan baik pada berbagai jenis gangguan. Selain itu, keterbatasan dalam deteksi postur tertentu seperti c7 menunjukkan bahwa model masih perlu dikembangkan untuk menangani variasi yang lebih kompleks. Terbatasnya teknik yang diterapkan juga menjadi kendala, di mana teknik augmentasi data yang tidak digunakan mungkin dapat meningkatkan kinerja model.

4. KESIMPULAN

Penelitian ini telah mengevaluasi kinerja model ResNet-50 dalam mendeteksi gangguan pengemudi menggunakan dataset setir kanan dan kiri. Hasil eksperimen menunjukkan bahwa model ini mampu mencapai akurasi 96% pada dataset setir kanan, mengindikasikan kemampuannya dalam menggeneralisasi pada data dengan kualitas yang lebih baik. Perbandingan dengan dataset setir kiri yang memiliki akurasi 84% menunjukkan pengaruh kualitas data terhadap performa model.

Studi ini memperlihatkan bahwa jenis gangguan yang melibatkan perubahan postur yang kompleks, seperti menjangkau sesuatu di belakang kemudi, merupakan tantangan utama dalam deteksi. Keterbatasan dataset yang digunakan berdampak pada kinerja model, menunjukkan bahwa data yang lebih beragam dan teknik augmentasi data dapat memberikan peningkatan signifikan dalam akurasi.

Meskipun sistem ini menunjukkan akurasi yang memadai, penelitian ini juga mengidentifikasi beberapa keterbatasan, termasuk ketergantungan pada dataset yang terbatas dan kemampuan model dalam menangani variasi postur. Penerapan teknik augmentasi data dan peningkatan dalam diversifikasi dataset dapat menjadi arah yang bermanfaat untuk penelitian di masa depan.

Secara keseluruhan, hasil penelitian ini memberikan kontribusi penting dalam pengembangan sistem deteksi distraksi pengemudi dan menyoroti kebutuhan untuk penelitian lebih lanjut guna meningkatkan akurasi dan daya generalisasi model dalam lingkungan nyata.

DAFTAR PUSTAKA

[1] "Distracted Driving Event; Traffic Fatality Data Release | NHTSA." Diakses: 2 September 2024. [Daring]. Tersedia pada: https://www.nhtsa.gov/speeches-presentations/distracted-driving-event-put-phone-away-or-pay-campaign

[2] G. Hermawan dan E. Husni, "Acquisition, Modeling, and Evaluating Method of Driving Behavior Based on OBD-II: A Literature Survey," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 879, no. 1, hlm. 012030, Jul 2020, doi: 10.1088/1757-899X/879/1/012030.

[3] "Distracted Driving Dangers and Statistics | NHTSA." Diakses: 12 Januari 2023. [Daring]. Tersedia pada: https://www.nhtsa.gov/risky-driving/distracted-driving

[4] "Kecelakaan Lalu Lintas Meningkat, Capai 94 Ribu Kasus sampai September 2022." Diakses: 13 Januari 2023. [Daring]. Tersedia pada: https://databoks.katadata.co.id/datapublish/2022/11/22/kecelakaan-lalu-lintas-meningkat-capai-94-ribu-kasus-sampai-september-2022

[5] "Distracted Driving | Transportation Safety | Injury Center | CDC." Diakses: 13 Januari 2023. [Daring]. Tersedia pada: https://www.cdc.gov/transportationsafety/distracted_driving/index.html

[6] J. Bärgman, "Methods for Analysis of Naturalistic Driving Data in Driver Behavior Research," Doctoral dissertation, Chalmers University of Technology, Gothenburg, Sweden, 2016.

[7] H. Singh dan A. Kathuria, "Analyzing driver behavior under naturalistic driving conditions: A review," *Accid. Anal. Prev.*, vol. 150, hlm. 105908, Feb 2021, doi: 10.1016/j.aap.2020.105908.

[8] Y. Abouelnaga, H. M. Eraqi, dan M. N. Moustafa, "Real-time Distracted Driver Posture Classification," *ArXiv170609498 Cs*, Nov 2018, Diakses: 15 Juli 2021. [Daring]. Tersedia pada: http://arxiv.org/abs/1706.09498

[9] H. M. Eraqi, Y. Abouelnaga, M. H. Saad, dan M. N. Moustafa, "Driver Distraction Identification with an Ensemble of Convolutional Neural Networks," *J. Adv. Transp.*, vol. 2019, hlm. 1–12, Feb 2019, doi: 10.1155/2019/4125865.

[10] S. Fu, Z. Yang, Y. Ma, Z. Li, L. Xu, dan H. Zhou, "Advancements in the Intelligent Detection of Driver Fatigue and Distraction: A Comprehensive Review," *Appl. Sci.*, vol. 14, no. 7, Art. no. 7, Jan 2024, doi: 10.3390/app14073016.

[11] F. Nel dan M. Ngxande, "Driver Activity Recognition Through Deep Learning," dalam *2021 Southern African Universities Power Engineering Conference/Robotics and Mechatronics/Pattern Recognition Association of South Africa (SAUPEC/RobMech/PRASA)*, Potchefstroom, South Africa: IEEE, Jan 2021, hlm. 1–6. doi: 10.1109/SAUPEC/RobMech/PRASA52254.2021.9377022.

[12] Abeer. A. Aljohani, "Real-time driver distraction recognition: A hybrid genetic deep network based approach," *Alex. Eng. J.*, vol. 66, hlm. 377–389, Mar 2023, doi: 10.1016/j.aej.2022.12.009.

[13] T. Khan, G. Choi, dan S. Lee, "EFFNet-CA: An Efficient Driver Distraction Detection Based on Multiscale Features Extractions and Channel Attention Mechanism," *Sensors*, vol. 23, no. 8, Art. no. 8, Jan 2023, doi: 10.3390/s23083835.

[14] K. He, X. Zhang, S. Ren, dan J. Sun, "Deep Residual Learning for Image Recognition," dipresentasikan pada Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, hlm. 770–778. Diakses: 10 Mei 2022. [Daring]. Tersedia pada: https://openaccess.thecvf.com/content_cvpr_2016/html/He_Deep_Residual_Learning_CVPR_2016_paper.html