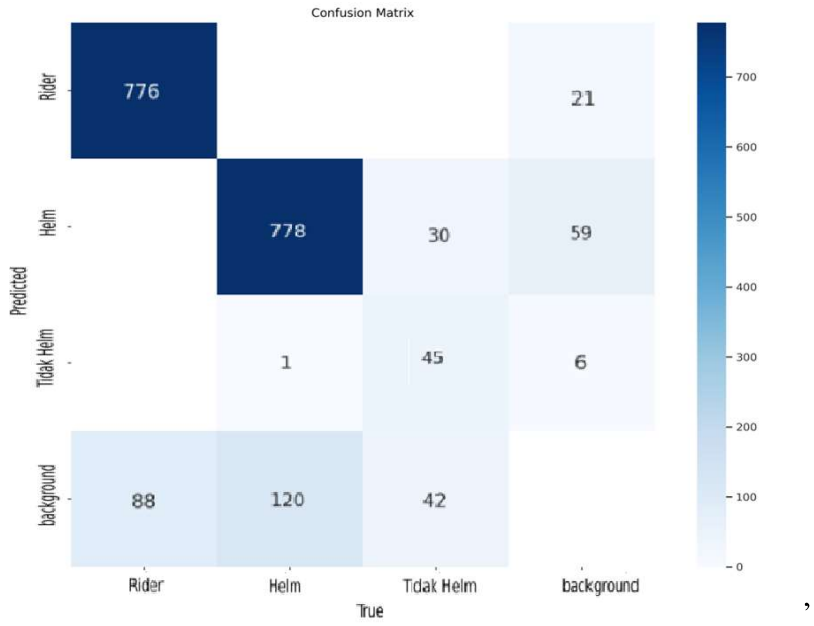
BAB IV
ANALISIS DAN PEMBAHASAN

4.1 Analisis Model *Training* YOLOv8

Proses eksperimental dalam pengembangan model YOLOv8 yang disesuaikan dilakukan berulang kali pada dataset yang telah diolah sebelumnya. Upaya ini diarahkan untuk mendapatkan konfigurasi yang menghasilkan kinerja optimal. Pemantauan eksperimen dijalankan dengan menggunakan ClearML, yang memfasilitasi *tracking* yang rinci dan terstruktur. Dihasilkan metrik evaluasi dan *confusion matrix* dari model *training* yang diterapkan pada data validasi, berikut *confusion matrix* yang dihasilkan seperti pada Gambar 4.1.



Gambar 4.1 *Confusion Matrix* Model *Training*

Pada Gambar 4.1, *confusion matrix* mengungkapkan bahwa model dengan tepat mengklasifikasikan ‘Rider’ sebanyak 776 kali dari 864 anotasi dan ‘Helm’ sebanyak 778 kali dari 899 anotasi, menunjukkan tingkat *true positives* yang tinggi untuk kedua kelas tersebut. Di sisi lain, ‘Tidak Helm’ diidentifikasi benar hanya 45 kali dari 117 anotasi, yang lebih rendah dibandingkan dengan dua kelas lainnya. Kelas ‘*background*’ merupakan area yang dihasilkan oleh model YOLOv8, kelas ini merupakan area non-objek, yang akan berkorelasi dengan

prediksi *false negative* dan *false positive* agar lebih komprehensif. Kesalahan deteksi terjadi saat:

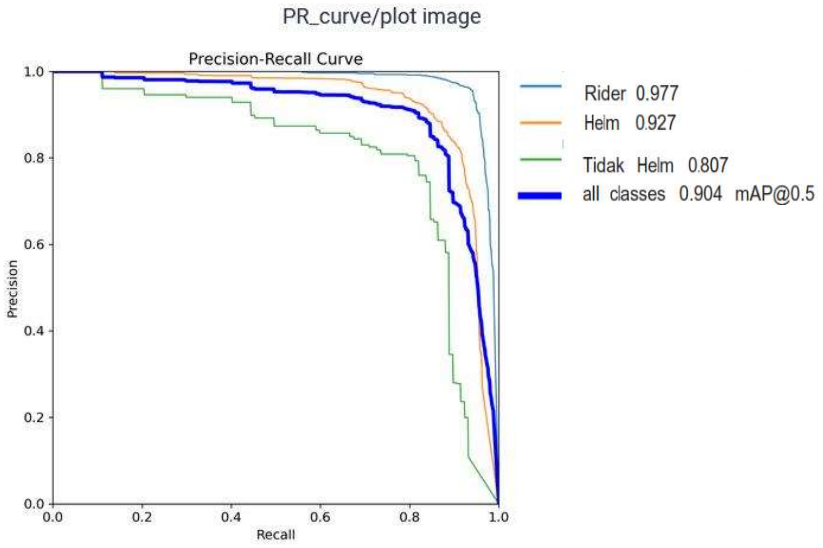
a. Kelas 'Rider' diprediksi sebagai '*background*' sebanyak 88 kali.
b. Kelas 'Helm' diprediksi sebagai kelas 'Tidak Helm' 1 kali, dan '*background*' 120 kali.
c. Kelas 'Tidak Helm' diprediksi sebagai kelas 'Helm' sebanyak 30 kali, dan '*background*' 42 kali.

Model ini juga sering mengklasifikasikan '*background*' sebagai kelas objek, dengan sebanyak 6 untuk kelas 'Rider', 59 untuk kelas 'Helm', 21 untuk kelas 'Tidak Helm'. Dari hasil analisis *confusion matrix*, dapat dilihat korelasinya dengan nilai metrik evaluasi seperti pada Tabel 4.1.

Tabel 4.1 Hasil Metrik Evaluasi Model *Training*

Kelas	*Precision*	*Recall*	mAP50	mAP50-95	F_1-*score*
Semua Kelas	85.9%	87.9%	90.4%	50.5%	86.8%
Rider	95.6%	93.8%	97.7%	68.0%	94.5%
Helm	81.3%	91.5%	92.7%	46.2%	85.9%
Tidak Helm	80.6%	78.3%	80.7%	37.3%	79.4%

Pada Tabel 4.2, hasil evaluasi YOLOv8 menunjukkan *precision* 85.9% dan *recall* 87.9%. Namun, mAP50-95 rendah menunjukkan kesulitan dalam mendeteksi objek pada IoU tinggi. Deteksi 'Rider' unggul dengan nilai *precision*, *recall*, mAP50, mAP50-95, dan f_1-*score* yang tinggi. Deteksi 'Helm' memiliki *recall* tinggi namun *precision* rendah, cenderung menghasilkan *false positive* pada IoU tinggi. Deteksi 'Tidak Helm' menunjukkan keseimbangan, meskipun kesulitan pada IoU tinggi. Kesulitan pada IoU tinggi disebabkan oleh bentuk objek yang rumit, variasi pencahayaan, dan gangguan lingkungan, serta dataset *training* yang kurang representatif atau ketidakseimbangan sampel kelas. Selain itu, kurva *precision-recall* seperti pada Gambar 4.2 memberikan pemahaman lebih rinci tentang performa model.



Gambar 4.2 Kurva *Precision-Recall* Model *Training*

Pada Gambar 4.2, kurva *precision-recall* menunjukkan keseimbangan antara *precision* dan recall untuk berbagai nilai threshold pada tiga kelas dan keseluruhan kelas. Kelas 'Rider' memiliki *Average Precision* (AP) tertinggi sebesar 0.977, diikuti oleh 'Helm' pada 0.927, dan 'Tidak Helm' pada 0.807, dengan mAP untuk semua kelas sebesar 0.904 pada IoU 0.5. Kurva menunjukkan *precision* tinggi pada recall rendah, menandakan keakuratan model dalam mengidentifikasi kasus positif yang mudah, namun *precision* menurun dengan peningkatan recall. Penurunan *precision* pada recall tinggi menunjukkan kesulitan model dalam mengidentifikasi semua kasus positif. *Area under curve* (AUC) mengukur seberapa baik model membedakan antara kelas, dan kurva yang mendekati sudut kanan atas menunjukkan model yang akurat.

4.2 Analisis *Testing* Model Pada Video

Dalam tahap ini, model pelatihan yang telah jadi akan melanjutkan proses *testing* dengan mengintegrasikan model ke dalam algoritma *tracking* Bytetrack dan menganalisis kinerjanya pada video, dengan parameter evaluasi yang sama dengan sebelumnya. Pada kasus pertama yaitu kasus pencahayaan pagi hari, memiliki hasil deteksi serta evaluasi seperti Tabel 4.2.

Tabel 4.2 Hasil Prediksi Dan Evaluasi Model Pada Video *Testing* Pagi Hari

Kelas	Aktual	Hasil Prediksi			*Precision*	*Recall*	F_1-*score*
		TP	FP	FN			
Rider	350	350	12	0	96.7%	100.0%	98.3%
Helm	443	443	48	0	90.2%	100.0%	94.9%
Tidak Helm	30	24	10	6	70.6%	80.0%	75.0%

Pada Tabel 4.2, hasil prediksi dan evaluasi model untuk kondisi pencahayaan pagi hari menunjukkan bahwa kelas 'Rider' memiliki *precision* tertinggi sebesar 96.7%. Kelas 'Rider' dan 'Helm' mencapai nilai recall sempurna sebesar 100.0%, sedangkan kelas 'Tidak Helm' memiliki recall sebesar 80.0%. Mean harmonik antara *precision* dan *recall* tertinggi ditemukan pada kelas 'Rider', yaitu 98.3%. Meskipun model akurat untuk kelas 'Rider' dan 'Helm', terdapat kasus *false positive* pada objek non-aktual dan latar belakang, serta prediksi salah untuk kelas lain. Kelas 'Tidak Helm' mengalami *false positive* dan *false negative* yang mempengaruhi nilai evaluasi. Pada kasus kedua yaitu kasus pencahayaan siang hari, memiliki hasil deteksi serta evaluasi seperti Tabel 4.3.

Tabel 4.3 Hasil Prediksi Dan Evaluasi Model Pada Video *Testing* Siang Hari

Kelas	Aktual	Hasil Prediksi			*Precision*	*Recall*	F_1-*score*
		TP	FP	FN			
Rider	381	381	24	0	94.1%	100.0%	96.9%
Helm	481	480	44	1	91.6%	99.8%	95.5%
Tidak Helm	19	13	25	6	34.2%	68.4%	45.6%

Pada Tabel 4.3, hasil prediksi dan evaluasi model pada kondisi pencahayaan siang hari menunjukkan bahwa kelas 'Rider' dan 'Helm' memiliki *precision* tinggi, sementara kelas 'Tidak Helm' memiliki precision rendah yaitu 34.2%. Nilai *recall* sempurna dicapai pada kelas 'Rider' dan hampir sempurna pada kelas 'Helm', tetapi rendah pada kelas 'Tidak Helm' yaitu 68.4%. Mean harmonik antara *precision* dan *recall* tertinggi ditemukan pada kelas 'Rider' dengan 96.9%, sedangkan kelas 'Tidak Helm' hanya 45.6%. Kesalahan prediksi pada kelas 'Tidak Helm' disebabkan oleh *false positive* pada objek non-aktual dan latar

belakang serta kesalahan prediksi pada kelas lain. Beberapa pelanggaran 'Tidak Helm' juga terlewat atau *false negative*, mempengaruhi nilai evaluasi kelas tersebut. Pada kasus terakhir yaitu kasus pencahayaan sore hari, memiliki hasil deteksi serta evaluasi seperti Tabel 4.4.

Tabel 4.4 Hasil Prediksi Dan Evaluasi Model Pada Video *Testing* Siang Hari

Kelas	**Aktual**	**Hasil Prediksi**			***Precision***	***Recall***	***F_1-score***
		TP	**FP**	**FN**			
Rider	610	607	5	3	99.2%	99.5%	99.4%
Helm	728	721	53	7	93.2%	99.0%	96.0%
Tidak Helm	16	11	23	5	32.4%	68.8%	44.0%

Pada Tabel 4.4, hasil evaluasi model pada kondisi pencahayaan siang hari menunjukkan kelas 'Rider' dan 'Helm' memiliki *precision* tinggi, masing-masing 99.2% dan 93.2%, sedangkan *precision* kelas 'Tidak Helm' rendah, hanya 32.4%. *Recall* kelas 'Rider' dan 'Helm' baik, tetapi rendah untuk kelas 'Tidak Helm' yaitu 68.8%. Mean harmonik antara *precision* dan *recall* tertinggi pada kelas 'Rider' mencapai 99.5%, tetapi rendah pada kelas 'Tidak Helm' yaitu 44.0%. Kesalahan prediksi pada kelas 'Tidak Helm' disebabkan oleh *false positive* pada objek non-aktual dan latar belakang serta prediksi yang salah pada kelas lain. Beberapa kasus pada ketiga kelas terlewat dalam prediksi atau terdeteksi sebagai *false negative*, memengaruhi evaluasi performa ketiga kelas tersebut.

4.3 Analisis Pengujian Sistem *Live Feed* menggunakan Jetson Nano

Pada tahap ini, model dan sistem yang sudah diuji dan dikembangkan sebelumnya akan langsung diuji dalam lingkungan operasional menggunakan perangkat Jetson Nano. Pengujian dilakukan untuk mengevaluasi kinerja sistem dalam mendeteksi objek secara *real-time*. Pengujian dilaksanakan pada ketiga kondisi pencahayaan yang berbeda, yakni pagi, siang, dan sore, dan berlangsung selama 25 menit. Analisis dilakukan terhadap metrik evaluasi seperti *precision*, *recall*, dan *f_1-score*, serupa dengan tahap sebelumnya. Selain itu, evaluasi juga mempertimbangkan performa mikrokomputer Jetson Nano dalam menjalankan sistem, dengan memperhatikan penggunaan sumber daya. Analisis ini bertujuan untuk

mendapatkan pemahaman yang lebih mendalam tentang kinerja sistem secara keseluruhan dalam konteks pengujian yang dilakukan. Pada Gambar 4.3, merupakan tampilan dari sistem deteksi helm dengan penggunaan Jetson Nano.

Gambar 4.3 Tampilan Sistem Deteksi Helm *Real-time* menggunakan Jetson Nano

Pada Gambar 4.3, merupakan *Graphical User Interface* (GUI) untuk sistem deteksi helm yang dioperasikan pada laptop melalui aplikasi *remote access*, yaitu PuTTY. Jendela di sebelah kanan menampilkan *jtop* yang digunakan untuk mengevaluasi kinerja dan menganalisis sumber daya Jetson Nano ketika sistem deteksi helm sedang berjalan. Evaluasi mempertimbangkan performa mikrokomputer Jetson Nano dalam menjalankan sistem, dengan memperhatikan penggunaan sumber daya seperti CPU1 hingga CPU4, RAM, dan GPU, suhu komponen, daya termal serta daya total saat operasi sistem.

4.3.2 Hasil Analisis Pengujian Sistem pada Pagi Hari

Pada kasus pertama, yang merujuk pada kondisi pencahayaan pada pagi hari, pengujian dilakukan pada pukul 07.18 pagi, dimulai dengan periode pengoperasian sistem *live feed* selama 25 menit. Hasil pengujian yang mencakup kinerja model dalam mengidentifikasi objek disajikan pada Tabel 4.5.

Tabel 4.5 Hasil Prediksi Dan Evaluasi Model Pada Video *Testing* Pagi Hari

Kelas	**Aktual**	**Hasil Prediksi**			***Precision***	***Recall***	***F_1-score***
		TP	**FP**	**FN**			

Rider	1294	1196	6	98	99.5%	92.4%	95.8%
Helm	1533	1473	13	60	99.1%	96.1%	97.6%
Tidak Helm	17	12	47	5	20.3%	70.6%	31.6%
Semua Kelas $\bar{x}$					73.0%	86.3%	75.0%

Pada Tabel 4.5, model menunjukkan *precision* dan *recall* yang sangat baik untuk kelas 'Rider' dan 'Helm' dengan kedua metrik tersebut lebih dari 90% dan nilai mean harmonik atau f_1*-score* masing-masing sebesar 95.8% dan 97.6%. Ini menandakan kemampuan yang sangat baik untuk mengidentifikasi pengendara dan helm dengan benar. Namun, kinerja model menurun secara signifikan untuk kelas 'Tidak Helm' dengan *precision* hanya 20.3%, meskipun *recall*-nya relatif lebih tinggi pada 70.6%. Ini menghasilkan f_1*-score* yang rendah sebesar 31.6%, menunjukkan bahwa model sering salah mengklasifikasikan objek lain atau latar belakang sebagai 'Tidak Helm'. Performa yang dihasilkan bisa dipengaruhi oleh beberapa faktor, termasuk performa penggunaan Jetson Nano secara *real-time*, kualitas *webcam* yang digunakan, serta kondisi pencahayaan di pagi hari. Secara keseluruhan, sementara model sangat efektif untuk dua kelas, model memerlukan peningkatan dalam membedakan kelas "Tidak Helm" untuk mengurangi *false positives* dan meningkatkan kinerja secara keseluruhan. Berikut adalah hasil kecepatan rata-rata saat menjalankan sistem untuk kasus pagi hari yang disajikan pada Tabel 4.6.

Tabel 4.6 Hasil Kecepatan Pemrosesan Deteksi Per-*frame* Pada Pagi Hari

Parameter	**$\bar{x}$ Waktu**
Pre-processing Speed	7.8 ms
Inference Time	72.4 ms
Post-processing Speed	6.6 ms

Pada Tabel 4.6, terdapat hasil rata-rata *output log* kecepatan pemrosesan per-*frame* dari pengujian sistem deteksi secara *real-time* menggunakan Jetson Nano. Rata-rata waktu pra-pemrosesan *frame* sekitar 7.8 ms menunjukkan kemampuan sistem dalam menyiapkan gambar untuk inferensi dengan cepat. Selama proses inferensi frame, waktu sekitar 72.4 ms dianggap wajar untuk sistem deteksi objek *real-time* pada perangkat *edge* seperti Jetson Nano. Meskipun ada sistem yang

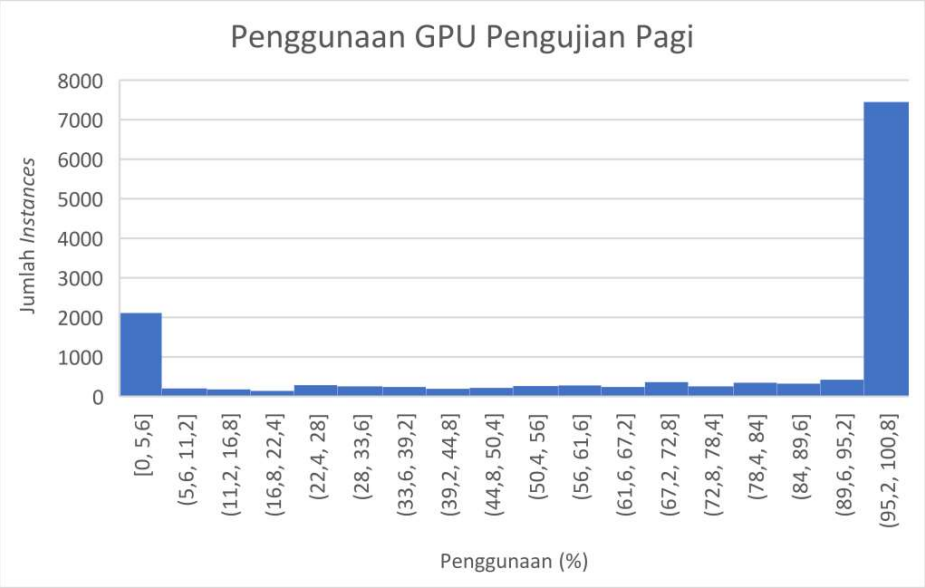
lebih cepat, waktu ini masih cukup baik untuk banyak aplikasi *real-time*. Waktu pasca-pemrosesan sekitar 6.6 ms menunjukkan kemampuan sistem dalam menyelesaikan tugas seperti pelacakan objek dan klasifikasi tambahan dengan cepat setelah deteksi. Dari analisis waktu pemrosesan yang dilakukan, diperoleh hasil kalkulasi untuk *frame per second* (FPS) sekitar 11.7. Meskipun angka FPS ini masih dianggap cukup untuk sebagian besar kasus, untuk mendeteksi objek kendaraan yang bergerak secara *real-time*, nilai FPS ini masih dianggap belum optimal. Faktor-faktor seperti sumber daya yang tersedia pada Jetson Nano serta kompleksitas model dan sistem dapat memengaruhi nilai FPS. Dari pengujian ini, didapat hasil kinerja komputasi Jetson Nano seperti pada Tabel 4.7.

Tabel 4.7 Hasil Performa Jetson Nano pada Pagi Hari

Metrik Performa	Penggunaan
CPU1	73.9%
CPU2	79.4%
CPU3	80.1%
CPU4	80.9%
RAM	76.3%
Temperatur CPU	34°C -70°C
Temperatur GPU	33.5°C -67.5°C
Daya Total	6.7 W

Pada Tabel 4.7, merupakan hasil performa komputasi Jetson Nano pada kasus pencahayaan di pagi hari. Perangkat ini menunjukkan kinerja yang intensif dengan penggunaan CPU mencapai 73.9% hingga 80.9% untuk keempat *core*-nya. Ini menandakan kemampuan Jetson Nano dalam menangani beban kerja dari algoritma deteksi objek. Meskipun demikian, masih ada ruang untuk pemrosesan data tambahan, sebagaimana ditunjukkan oleh penggunaan RAM yang mencapai 76.3%. Terjadi fluktuasi suhu yang signifikan, namun kemudian suhu kembali turun hingga mencapai kisaran 65°C untuk kedua komponen, menandakan adanya mekanisme pendinginan yang efektif untuk menjaga keandalan sistem dalam jangka panjang. Selain itu, terlihat efisiensi energi yang baik dari konsumsi daya perangkat sekitar 6.7 W selama operasi, berada di bawah batas operasional standar 10 W. Hal ini menunjukkan bahwa Jetson Nano mampu menjaga keseimbangan

beban operasional secara optimal, terutama untuk tugas-tugas yang memerlukan pemrosesan intensif seperti deteksi objek. Untuk penggunaan GPU, berikut disajikan grafik histogram pada Gambar 4.4 untuk persentase penggunaan GPU.



Gambar 4.4 Grafik Penggunaan GPU Pengujian Pagi Hari

Pada Gambar 4.4, merupakan grafik histogram untuk persentase penggunaan GPU selama operasi pada Jetson Nano untuk kasus pencahayaan pagi hari. Penggunaan GPU pada sistem deteksi helm menunjukkan pola fluktuasi yang signifikan. Fluktuasi ini mencapai puncak hingga 99.7%, yang menandakan periode di mana GPU bekerja hampir pada kapasitas maksimumnya. Setelah mencapai puncak, penggunaan GPU kemudian turun drastis mendekati 0%, menunjukkan bahwa ada interval di mana beban kerja GPU sangat ringan atau tidak ada sama sekali. Fluktuasi ini terjadi beberapa kali dan secara acak, yang bisa menandakan bahwa sistem melakukan pemrosesan data dalam *batch* pada interval yang tidak teratur, menunggu hingga akumulasi data tertentu terkumpul sebelum memulai proses inferensi, dan setelah *batch* data diproses, penggunaan GPU turun kembali karena sistem menunggu kumpulan data berikutnya.

4.3.3 Hasil Analisis Pengujian Sistem pada Siang Hari

Pada kasus kedua, yang merujuk pada kondisi pencahayaan pada siang hari, pengujian dilakukan pada pukul 14.25 siang, dimulai dengan periode

pengoperasian sistem *live feed* selama 25 menit. Hasil pengujian yang mencakup kinerja model dalam mengidentifikasi objek disajikan pada Tabel 4.8.

Tabel 4.8 Hasil Prediksi Dan Evaluasi Model Pada Pengujian *Real-time* Sistem Siang Hari

Kelas	**Aktual**	**Hasil Prediksi**			***Precision***	***Recall***	***F_1-score***
		TP	**FP**	**FN**			
Rider	768	264	0	504	100.0%	34.4%	51.2%
Helm	949	605	2	344	99.7%	63.8%	77.8%
Tidak Helm	34	16	42	18	27.6%	47.1%	34.8%
Semua Kelas x̄					75.8%	48.4%	54.6%

Pada Tabel 4.8, model menunjukkan tingkat presisi yang tinggi untuk kelas 'Rider' dan 'Helm', melebihi 90%. Namun, *recall* untuk kedua kelas tersebut relatif rendah, menyebabkan penurunan *f_1-score*. Hal ini menandakan kemampuan model dalam mengenali pengendara dan helm dengan baik, namun seringkali melewatkan objek yang sebenarnya ada. Kinerja model juga rendah untuk kelas 'Tidak Helm', dengan presisi hanya sekitar 27.6% dan *recall* sekitar 47.1%, menghasilkan *f_1-score* rendah sekitar 34.8%. Kemungkinan performa yang rendah dipengaruhi oleh beberapa faktor, termasuk kinerja Jetson Nano, kualitas *webcam*, dan kondisi pencahayaan. Secara keseluruhan, model kurang efektif dalam mengklasifikasikan ketiga kelas, memerlukan peningkatan untuk membedakan kelas 'Tidak Helm' dan meningkatkan kinerja secara keseluruhan. Berikut adalah hasil kecepatan rata-rata saat menjalankan sistem untuk kasus siang hari yang disajikan pada Tabel 4.9.

Tabel 4.9 Hasil Kecepatan Pemrosesan Deteksi Per-*frame* Pada Siang Hari

Parameter	**x̄ Waktu**
Pre-processing Speed	7.9 ms
Inference Time	78.3 ms
Post-processing Speed	4.9 ms

Pada Tabel 4.9, terdapat hasil rata-rata *output log* kecepatan pemrosesan per-*frame* dari pengujian sistem deteksi secara *real-time* menggunakan Jetson Nano.

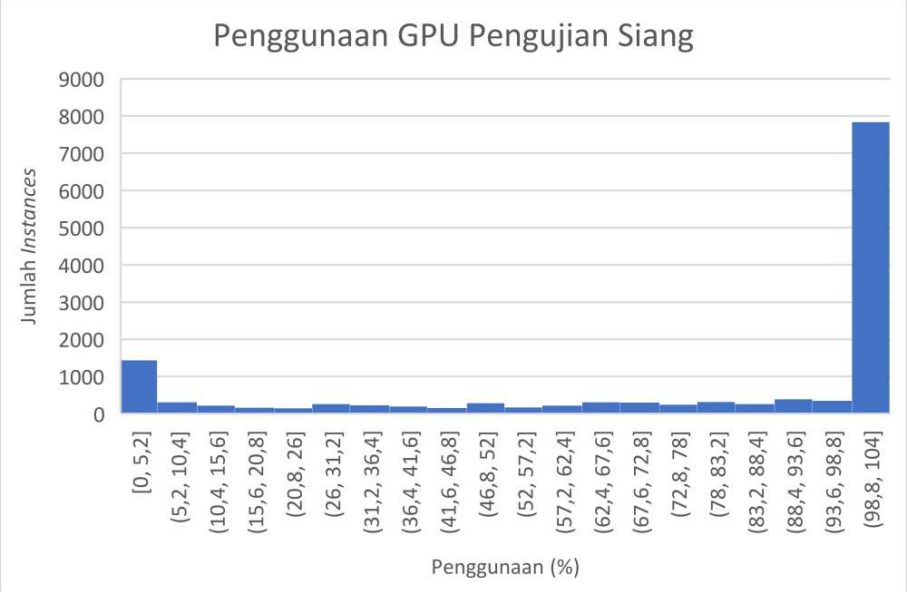
Rata-rata waktu pra-pemrosesan *frame* sekitar 7.9 ms menunjukkan kemampuan sistem dalam menyiapkan gambar untuk inferensi dengan cepat. Selama proses inferensi frame, waktu sekitar 78.3 ms dianggap wajar untuk sistem deteksi objek *real-time* pada perangkat *edge* seperti Jetson Nano. Meskipun ada sistem yang lebih cepat, waktu ini masih cukup baik untuk banyak aplikasi *real-time*. Waktu pasca-pemrosesan sekitar 4.9 ms menunjukkan kemampuan sistem dalam menyelesaikan tugas seperti pelacakan objek dan klasifikasi tambahan dengan lebih cepat dari pengujian sebelumnya setelah deteksi. Dari analisis waktu pemrosesan yang dilakukan, diperoleh hasil kalkulasi untuk *frame per second* (FPS) sekitar 11.1. Meskipun angka FPS ini masih dianggap cukup untuk sebagian besar kasus, untuk mendeteksi objek kendaraan yang bergerak secara *real-time*, nilai FPS ini masih dianggap belum optimal. Faktor-faktor seperti sumber daya yang tersedia pada Jetson Nano serta kompleksitas model dan sistem dapat memengaruhi nilai FPS. Dari pengujian ini, didapat hasil kinerja komputasi Jetson Nano seperti pada Tabel 4.10.

Tabel 4.10 . Hasil Performa Jetson Nano pada Siang Hari

Metrik Performa	Penggunaan
CPU1	77.1%
CPU2	81.1%
CPU3	82.6%
CPU4	83.3%
RAM	79.6%
Temperatur CPU	34°C -50.5°C
Temperatur GPU	34°C -49°C
Daya Total	6.2 W

Pada Tabel 4.10 ditunjukkan hasil performa komputasi Jetson Nano pada kasus pencahayaan di siang hari. Perangkat ini menunjukkan penggunaan CPU antara 77.1% hingga 83.3% untuk keempat *core*-nya, menandakan kemampuannya dalam menangani beban kerja dari algoritma deteksi objek. Penggunaan RAM mencapai 79.6%, menunjukkan masih ada ruang untuk pemrosesan data tambahan. Terjadi peningkatan suhu yang cukup signifikan, namun suhu kembali stabil di kisaran 45°C untuk kedua komponen, menunjukkan mekanisme pendinginan yang efektif. Konsumsi daya perangkat sekitar 6.2 W, berada di

bawah batas operasional standar 10 W, menunjukkan efisiensi energi yang baik. Untuk penggunaan GPU, disajikan grafik histogram pada Gambar 4.5.



Gambar 4.5 Grafik Penggunaan GPU Pengujian Siang Hari

Pada Gambar 4.5, merupakan grafik histogram untuk persentase penggunaan GPU selama operasi pada Jetson Nano untuk kasus pencahayaan pagi hari. Penggunaan GPU pada sistem deteksi helm menunjukkan pola fluktuasi yang signifikan. Fluktuasi ini mencapai puncak hingga 99.9%, yang menandakan periode di mana GPU bekerja hampir pada kapasitas maksimumnya. Setelah mencapai puncak, penggunaan GPU kemudian turun drastis mendekati 0%, menunjukkan bahwa ada interval di mana beban kerja GPU sangat ringan atau tidak ada sama sekali. Fluktuasi ini terjadi beberapa kali dan secara acak, yang bisa menandakan bahwa sistem melakukan pemrosesan data dalam *batch* pada interval yang tidak teratur, menunggu hingga akumulasi data tertentu terkumpul sebelum memulai proses inferensi, dan setelah *batch* data diproses, penggunaan GPU turun kembali karena sistem menunggu kumpulan data berikutnya.

4.3.4 Hasil Analisis Pengujian Sistem pada Sore Hari

Pada kasus terakhir, yang merujuk pada kondisi pencahayaan pada siang hari, pengujian dilakukan pada pukul 18.02 sore, dimulai dengan periode

pengoperasian sistem *live feed* selama 25 menit. Hasil pengujian yang mencakup kinerja model dalam mengidentifikasi objek disajikan pada Tabel 4.11.

Tabel 4.11 Hasil Prediksi Dan Evaluasi Model Pada Pengujian *Real-time* Sistem Sore Hari

Kelas	**Aktual**	**Hasil Prediksi**			***Precision***	***Recall***	***F_1-score***
		TP	**FP**	**FN**			
Rider	1336	1291	7	45	99.5%	96.6%	98.0%
Helm	1567	1533	33	34	97.9%	97.8%	97.9%
Tidak Helm	35	9	11	26	45.0%	25.7%	32.7%
Semua Kelas x̄					80.8%	73.4%	76.2%

Pada Tabel 4.11, model menunjukkan *precision* dan *recall* yang sangat baik untuk kelas ‘Rider’ dan ‘Helm’ dengan kedua metrik tersebut lebih dari 90% dan nilai mean harmonik atau *f_1-score* masing-masing sebesar 98.0% dan 97.9%. Ini menandakan kemampuan yang sangat baik untuk mengidentifikasi pengendara dan helm dengan benar. Namun, kembali kinerja model lebih rendah untuk kelas ‘Tidak Helm’ dengan *precision* hanya 45.0% dan *recall* sebesar 25.7%. Ini menghasilkan *f_1-score* yang rendah sebesar 32.7%, menunjukkan bahwa model sering salah mengklasifikasikan objek 'Tidak Helm' atau melewatkan deteksinya sebagai latar belakang. Performa yang dihasilkan bisa dipengaruhi oleh beberapa faktor, termasuk performa penggunaan Jetson Nano secara *real-time*, kualitas *webcam* yang digunakan, serta kondisi pencahayaan di sore hari yang cenderung lebih gelap. Secara keseluruhan, sementara model sangat efektif untuk dua kelas, model memerlukan peningkatan dalam membedakan kelas “Tidak Helm” untuk mengurangi *false positives* dan meningkatkan kinerja secara keseluruhan. Berikut adalah hasil kecepatan rata-rata saat menjalankan sistem untuk kasus sore hari yang disajikan pada Tabel 4.12.

Tabel 4.12 Hasil Kecepatan Pemrosesan Deteksi Per-*frame* Pada Sore Hari

Parameter	**x̄ Waktu**
Pre-processing Speed	8.0 ms

Inference Time	85.6 ms
Post-processing Speed	5.0 ms

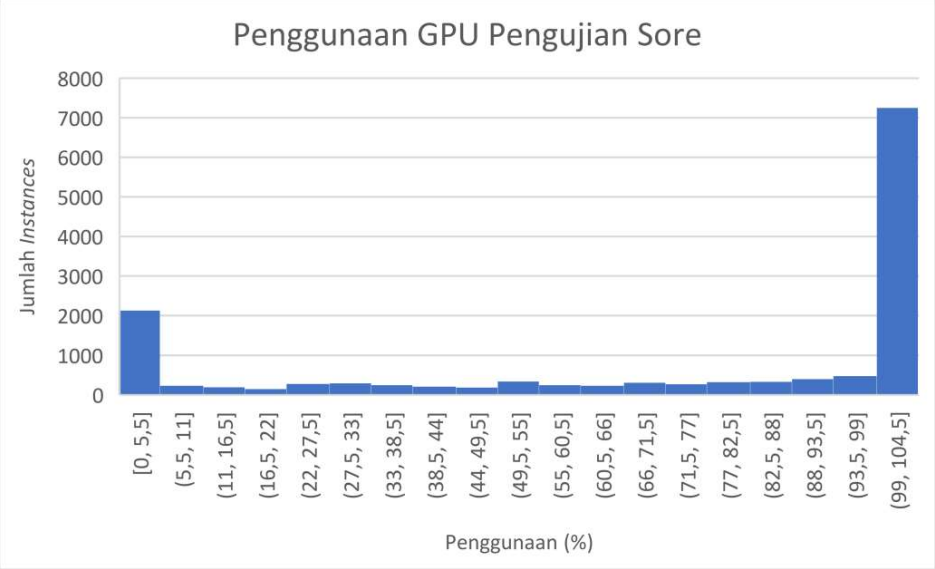
Pada Tabel 4.12, hasil rata-rata *output log* kecepatan pemrosesan per-*frame* dari pengujian sistem deteksi secara *real-time* menggunakan Jetson Nano ditampilkan. Waktu *pre-processing frame* sekitar 8.0 ms, menunjukkan kemampuan sistem dalam menyiapkan gambar untuk inferensi dengan cepat. Waktu inferensi *frame* sekitar 85.6 ms, sedikit lebih lama dari pengujian sebelumnya, namun masih wajar untuk perangkat *edge* seperti Jetson Nano. Waktu *post-processing* sekitar 5.0 ms, menunjukkan kemampuan sistem dalam menyelesaikan tugas tambahan dengan cepat setelah deteksi. Hasil analisis menunjukkan *frame per second* (FPS) sekitar 10.2, meskipun sedikit lebih rendah dari pengujian sebelumnya. Meskipun FPS ini cukup untuk banyak kasus, untuk deteksi objek kendaraan yang bergerak secara real-time, nilai FPS masih dianggap belum optimal. Faktor-faktor seperti sumber daya yang tersedia pada Jetson Nano dan kompleksitas model dan sistem dapat memengaruhi nilai FPS. Dari pengujian ini, didapat hasil kinerja komputasi Jetson Nano seperti pada Tabel 4.13.

Tabel 4.13 Hasil Performa Jetson Nano pada Sore Hari

Metrik Performa	**Penggunaan**
CPU1	70.8%
CPU2	75.8%
CPU3	76.9%
CPU4	77.3%
RAM	76.4%
Temperatur CPU	33.3°C -72°C
Temperatur GPU	35°C -69°C
Daya Total	6.5 W

Pada Tabel 4.13, merupakan hasil performa komputasi Jetson Nano pada kasus pencahayaan di siang hari. Perangkat ini menunjukkan kinerja yang intensif dengan penggunaan CPU mencapai 70.8% hingga 77.3% untuk keempat *core*-nya, namun tidak se-intensif kedua pengujian sebelumnya. Ini menandakan kemampuan Jetson Nano dalam menangani beban kerja dari algoritma deteksi objek. Meskipun demikian, masih ada ruang untuk pemrosesan data tambahan,

sebagaimana ditunjukkan oleh penggunaan RAM yang mencapai 76.4%. Terjadi peningkatan suhu yang signifikan, namun kemudian suhu kembali stabil di kisaran 50°C untuk kedua komponen, menandakan adanya mekanisme pendinginan yang efektif untuk menjaga keandalan sistem dalam jangka panjang. Selain itu, terlihat efisiensi energi yang baik dari konsumsi daya perangkat sekitar 6.5 W selama operasi, berada di bawah batas operasional standar 10 W. Hal ini menunjukkan bahwa Jetson Nano mampu menjaga keseimbangan beban operasional secara optimal, terutama untuk tugas-tugas yang memerlukan pemrosesan intensif seperti deteksi objek. Untuk penggunaan GPU, berikut disajikan grafik histogram pada Gambar 4.6 untuk persentase penggunaan GPU.



Gambar 4.6 Grafik Penggunaan GPU Pengujian Sore Hari

Pada Gambar 4.6, merupakan grafik histogram untuk persentase penggunaan GPU selama operasi pada Jetson Nano untuk kasus pencahayaan sore hari. Penggunaan GPU pada sistem deteksi helm menunjukkan pola fluktuasi yang signifikan. Sama seperti pengujian di kondisi sebelumnya, fluktuasi ini mencapai puncak hingga 99.9%, yang menandakan periode di mana GPU bekerja hampir pada kapasitas maksimumnya. Setelah mencapai puncak, penggunaan GPU kemudian turun drastis mendekati 0%, menunjukkan bahwa ada interval di mana beban kerja GPU sangat ringan atau tidak ada sama sekali. Fluktuasi ini terjadi beberapa kali dan secara acak, yang bisa menandakan bahwa sistem

melakukan pemrosesan data dalam *batch* pada interval yang tidak teratur, menunggu hingga akumulasi data tertentu terkumpul sebelum memulai proses inferensi, dan setelah *batch* data diproses, penggunaan GPU turun kembali karena sistem menunggu kumpulan data berikutnya.

4.4 Analisa Performa Sistem Deteksi Helm

Secara keseluruhan, faktor yang sangat berpengaruh pada performa sistem deteksi helm secara real-time adalah model yang digunakan dan beban komputasi itu sendiri. Dari model yang diperoleh untuk berbagai kondisi pencahayaan, diperoleh rata-rata untuk setiap kelas, serta rata-rata keseluruhan sistem di semua pengujian pencahayaan seperti yang disajikan pada Tabel 4.14 berikut.

Tabel 4.14 Hasil Pengujian Secara Keseluruhan

Kelas	***Precision*** $\bar{x}$	***Recall*** $\bar{x}$	F_1***-score*** $\bar{x}$
Rider	99.7%	74.5%	81.7%
Helm	98.9%	85.9%	91.1%
Tidak Helm	31.0%	47.8%	33.0%
Semua Kelas $\bar{x}$	76.5%	69.4%	68.6%

Pada Tabel 4.14, terlihat bahwa secara keseluruhan, pada kasus pengujian sistem deteksi, kelas ‘Helm’ memiliki nilai f_1*-score* yang lebih tinggi dibandingkan dengan kelas lainnya. Kelas dengan kinerja terendah ada pada kasus ‘Tidak Helm’. Hal ini menunjukkan bahwa model kesulitan membedakan kelas ‘Tidak Helm’ dari kelas lainnya, terutama dari kelas ‘Helm’. Rendahnya kinerja kelas ‘Tidak Helm’ memengaruhi nilai evaluasi model secara keseluruhan, menghasilkan nilai f_1*-score* yang tidak terlalu tinggi, yaitu 68.6%. Penyebabnya adalah banyaknya kasus *false positive* dan *false negative*. Kasus *false positive* dapat digambarkan pada sampel gambar pada pengujian langsung yang disajikan pada Gambar 4.7.

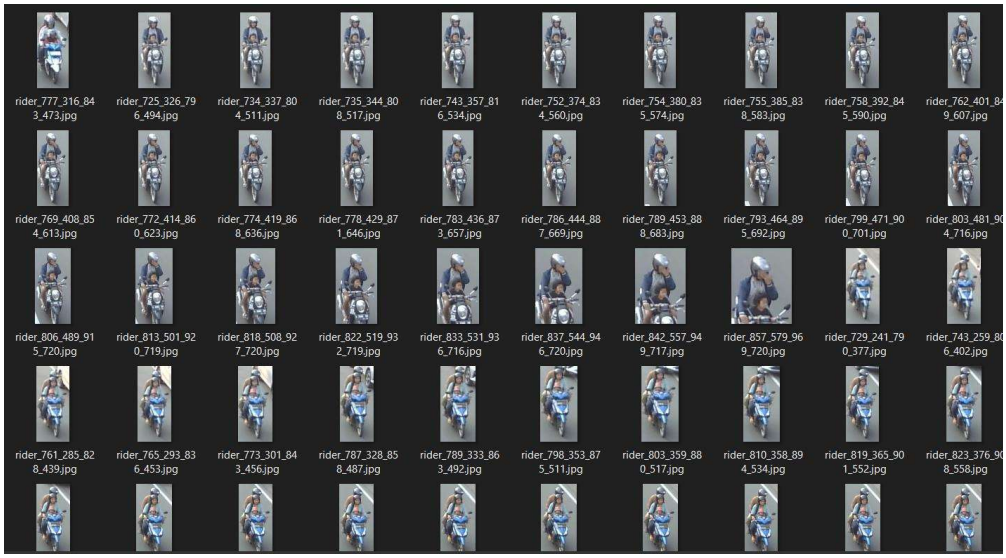
Gambar 4.7 Kasus *False Positives* pada Pengujian Langsung

Pada Gambar 4.7, terlihat beberapa contoh kasus *false positive* yang terjadi selama pengujian langsung. Untuk pengendara motor yang tidak menggunakan helm, sering kali prediksi yang muncul adalah kelas ‘Helm’, terutama saat objek mendekati kamera hingga keluar dari pandangan, namun dalam beberapa *frame* saat objek bergerak keluar dari pandangan, terkadang terdeteksi sebagai kelas ‘Tidak Helm’, menyebabkan deteksi yang tidak konsisten. Kasus *false positive* pada kelas ‘Tidak Helm’ sering terjadi pada pejalan kaki, yang bukan merupakan target penelitian sistem deteksi helm ini. Hal serupa juga terjadi pada kelas ‘Rider’, di mana pengendara motor yang melawan arah atau berada di jalur berbeda terkadang terdeteksi, meskipun kasus ini jarang. Ini disebabkan oleh sudut pandang kamera yang terlalu luas sehingga menampilkan trotoar, yang lebih efektif jika fokus pada jalan raya saja. Kasus lainnya pada kelas ‘Helm’ mencakup kesulitan membedakan antara aksesoris kepala seperti topi, kerudung, dan helm yang tidak khusus untuk pengendara motor. Pada kelas ‘Rider’, kesulitan muncul dalam membedakan pengendara sepeda, gerobak, dan kendaraan non-motor serupa lainnya. Selain itu, artefak lain di area jalan raya, seperti benda bulat, memengaruhi deteksi pada kelas ‘Helm’ dan ‘Tidak Helm’. Masalah ini bisa disebabkan oleh pencahayaan dan dataset yang kurang representatif. Kasus *false negative* dapat digambarkan pada sampel gambar pada pengujian langsung yang disajikan pada Gambar 4.7.

Gambar 4.8 Kasus *False Negatives* pada Pengujian Langsung

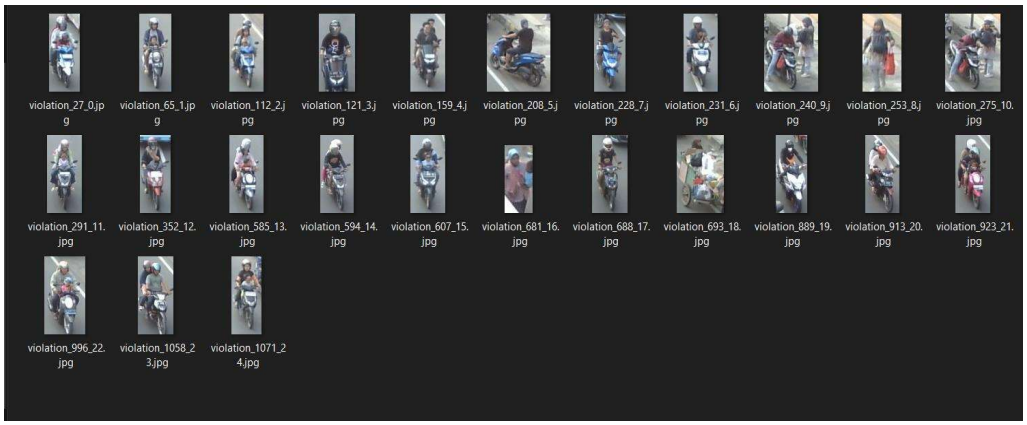
Pada Gambar 4.8, terlihat beberapa contoh kasus *false negative* yang terjadi selama pengujian langsung. Kasus *false negative* ini lebih sering melewatkan deteksi kelas ‘Rider’ dan ‘Tidak Helm’ dibandingkan dengan kelas ‘Helm’. Pada pengujian siang dan sore, kurangnya deteksi *bounding box* terjadi pada ketiga kelas saat siang hari, dan lebih sering melewatkan kelas ‘Rider’ saat cahaya redup di sore hari. Hal ini disebabkan oleh intensitas cahaya yang terlalu tinggi pada siang hari, sehingga *webcam* kesulitan mendeteksi objek dengan resolusi yang ada, serta cahaya yang redup pada sore hari yang menyebabkan lampu motor menyala, mengakibatkan *blur* ketika objek bergerak cepat. Kasus *false negative* juga muncul ketika pengendara bergerak terlalu cepat atau mengebut.

Penggunaan ByteTrack untuk mengurangi pengulangan atau duplikasi berlebih dalam hasil *cropping* pengendara motor terbukti cukup efisien dan efektif, meskipun terdapat sedikit *error* yang mungkin disebabkan oleh model itu sendiri. Tanpa sistem *tracking*, isi dari *folder* 'violation' terlihat seperti pada Gambar 4.9 berikut.



Gambar 4.9 Hasil *Cropping* tanpa Sistem *Tracking*

Pada Gambar 4.9, ditampilkan *folder* 'violation' yang berisi hasil *cropping* tanpa sistem tracking. Terlihat banyak duplikasi gambar dari objek yang sama, terutama ketika kelas 'Tidak Helm' bertumpang tindih dengan kelas 'Rider', menghasilkan sebanyak 218 gambar dalam satu *folder*. Berikut adalah tampilan folder 'violation' dengan sistem tracking ByteTrack pada Gambar 4.10.



Gambar 4.10 Hasil *Cropping* dengan menggunakan Sistem *Tracking*

Pada Gambar 4.10, ditampilkan *folder* 'violation' yang berisi hasil *cropping* dengan sistem *tracking* yang diintegrasikan. Terlihat bahwa duplikasi gambar dari objek yang sama berkurang secara signifikan ketika kelas 'Tidak Helm' bertumpang tindih dengan kelas 'Rider', sehingga hanya menghasilkan 25 gambar saja dalam satu folder.

Performa sistem juga bisa dipengaruhi oleh komputasi sistem, terutama saat menggunakan *edge computing* dengan mikrokomputer Jetson Nano. Saat melakukan pengujian langsung, terkadang FPS yang dihitung berdasarkan waktu inferensi tidak selalu sesuai dengan apa yang terlihat secara *real-time* di layar monitor. Hal ini mungkin disebabkan oleh beberapa faktor, termasuk proses pengujian yang dilakukan melalui *remote access*. Penggunaan *remote access* menyebabkan GUI Jetson Nano ditampilkan pada laptop, yang seringkali mengalami latensi dan membebani *bandwidth*, terutama ketika data harus dikirim bolak-balik antara Jetson Nano dan perangkat *remote*. Selain itu, pemrosesan *live feed* dan penggunaan model AI juga dapat memengaruhi komputasi secara keseluruhan, menyebabkan pemrosesan yang intensif pada komponen Jetson Nano. Akibatnya, suhu komponen dapat meningkat, dan ini dapat memicu *thermal throttling*. Di lingkungan luar ruangan, di mana suhu udara mungkin lebih tinggi, risiko *thermal throttling* bisa lebih besar. Notifikasi "*System throttled due to over-current*" muncul ketika Jetson Nano mendeteksi aliran listrik yang melampaui kapasitas maksimum yang dapat ditangani oleh sistem. Hal ini dapat terjadi jika Jetson Nano menggunakan perangkat tambahan seperti perangkat USB, terutama jika perangakt USB tersebut memiliki konsumsi daya yang tinggi, sehingga dapat menyebabkan tidak konsistennya total daya yang dikonsumsi oleh sistem secara keseluruhan.