**Judul : Computer Vision–Based Identification of Safety Compliance Among Motorcycle Riders**

**Topik**  : Deteksi Pelanggaran dan Perilaku Pengendara Motor

**Anggota** : [202210370311088 - Haidar Dimas Heryanto]

[202210370311095 - Alfi Aulia Azzahra]

**Tahap 0 (poin: 25)**: Business Objective/tujuan

# **Tahap 1**: Original Data

* Urgensi topik/kasus yang dipilih.

Berkendara dengan transportasi merupakan aktivitas umum yang dilakukan oleh banyak individu untuk mencapai suatu tujuan tertentu. Pada negara berkembang, sepeda motor masih menjadi transportasi utama dalam menjalani aktivitas sehari – hari, namun jika berada pada kondisi di jalan lalu lintas, mereka beresiko untuk mengalami kecelakaan yang mengalami cedera maupun kematian. Berdasarkan laporan dari Global Road Safety Status yang dirilis oleh World Health Organization (WHO), sekitar 1,35 juta orang meninggal dunia karena kecelakaan lalu lintas dan 28% nya adalah pengendara motor. Namun di Indonesia, Tingkat kepatuhan untuk menggunakan helm masih sangatlah rendah, padahal Helm merupakan perlengkapan penting yang digunakan untuk melindungi keselamatan kepala dari pengendara.

* Data yang digunakan.

Dataset yang digunakan berupa kumpulan citra pengendara sepeda motor beserta penumpangnya dalam berbagai kondisi, seperti mengenakan helm, tidak mengenakan helm, menggunakan ponsel, dan merokok saat berkendara.

Dataset ini dikumpulkan dari berbagai sumber dari Roboflow, Kaggle, serta beberapa jurnal penelitian lainnya yang kemudian dijadikan satu dataset besar. Total dataset yang terkumpul berjumlah sekitar kurang lebih 8000 citra, dengan 1360 citra yang telah diberi anotasi (labeled data) untuk proses pelatihan model. Pembagian dataset dilakukan menjadi 80% data latih (1088 citra), 10% data uji (136 citra), dan 10% data validasi (136 citra). Kami melalui proses pelabelan manual di Roboflow.

* Atribut pada Data

Sebelum melakukan preprocessing, kami melalui proses pelabelan manual di Roboflow dengan pelabelan 6 target class, yaitu :

1. **driver**, yaitu untuk mendeteksi pengendara sepeda motor.
2. **pillion**, yaitu untuk mendeteksi penumpang dari suatu kendaraan bermotor, untuk mendeteksi penumpang, bisa dideteksi pada bagian depan (seperti anak kecil) dari pengendara maupun di belakang pengendara motor.
3. **no\_helmet**, yaitu untuk mengidentifikasi apakah pengendara ataupun penumpang (pillion) tidak menggunakan helm.
4. **with\_helmet**, yaitu untuk mengidentifikasi apakah pengendara ataupun penumpang (pillion) menggunakan helm.
5. **smoking**, yaitu untuk mendeteksi pengendara atau penumpang (pillion) yang sedang merokok pada saat berkendara.
6. **use\_mobile**, yaitu untuk mendeteksi pengendara ataupun penumpang (pillion) yang sedang menggunakan Handphone pada saat berkendara.

* Sumber data.

[Dataset - Roboflow](https://universe.roboflow.com/aa-cqfub/h3lm-40l0-oid-xd-3dofr)



Gambar 1. Target Class Pada Penelitian

# **Tahap 2**: Target Data (Optional)

Fokus penelitian kali ini adalah untuk menganalisis perilaku pengendara yang menggunakan helm dan tidak yang menggunakan helm guna menghindari kecelakaan akibat tidak menggunakan helm. Oleh karena itu, pada pengujian kali ini, penelitian berfokus pada dua target class, yaitu no\_helmet dan with\_helmet. Alasan hanya menggunakan dua target class untuk pengujian kali ini dikarenakan helm memiliki kontribusi besar untuk keselamatan pengendara nya dan juga menggunakan helm merupakan salah satu peraturan yang harus ditaati saat berkendara yang menggunakan kendaraan bermotor.

# **Tahap 3**: Data Pre-processing & Transformation

[Machine Learning - 088 - 095.ipynb](https://colab.research.google.com/drive/1-1JJHw4me5QN-hAb1-M7uoO28IaRyDOG?usp=sharing)

## Preprocessing

Tahapan preprocessing pada project ini dilakukan menggunakan Google Colab dengan bantuan beberapa library di python. Proses ini dilakukan agar data yang digunakan dalam pelatihan model deteksi objek memiliki kualitas yang baik, ukuran yang seragam, serta variasi yang cukup untuk meningkatkan performa model.

Langkah-langkah utama preprocessing adalah sebagai berikut:

1. Pengunduhan Dataset dari Roboflow

Dataset diambil langsung melalui API Roboflow menggunakan workspace dan project yang telah dibuat sebelumnya. Dataset tersebut berisi gambar pengendara motor beserta anotasi (bounding box) dalam format YOLO. Setiap gambar sudah memiliki label sesuai dengan enam kelas target

Dataset dibagi menjadi tiga bagian, yaitu 80% untuk training, 10% untuk validation, dan 10% untuk testing, dengan total sekitar 1360 citra beranotasi dari kurang lebih 8000 gambar yang terkumpul dari berbagai sumber.

1. Resize (128×128 piksel)

Semua citra diubah ukurannya menjadi resolusi 128×128 piksel. Tujuan resize adalah untuk menyeragamkan dimensi input pada model deteksi objek dan menjaga efisiensi komputasi saat proses pelatihan.

1. Gaussian Blur (Noise Reduction)

Filter Gaussian Blur diterapkan dengan batas blur (3, 5) untuk mengurangi noise dan memperhalus citra tanpa menghilangkan detail penting dari objek utama. Langkah ini membantu meningkatkan ketepatan model dalam mengenali pola bentuk pada citra.

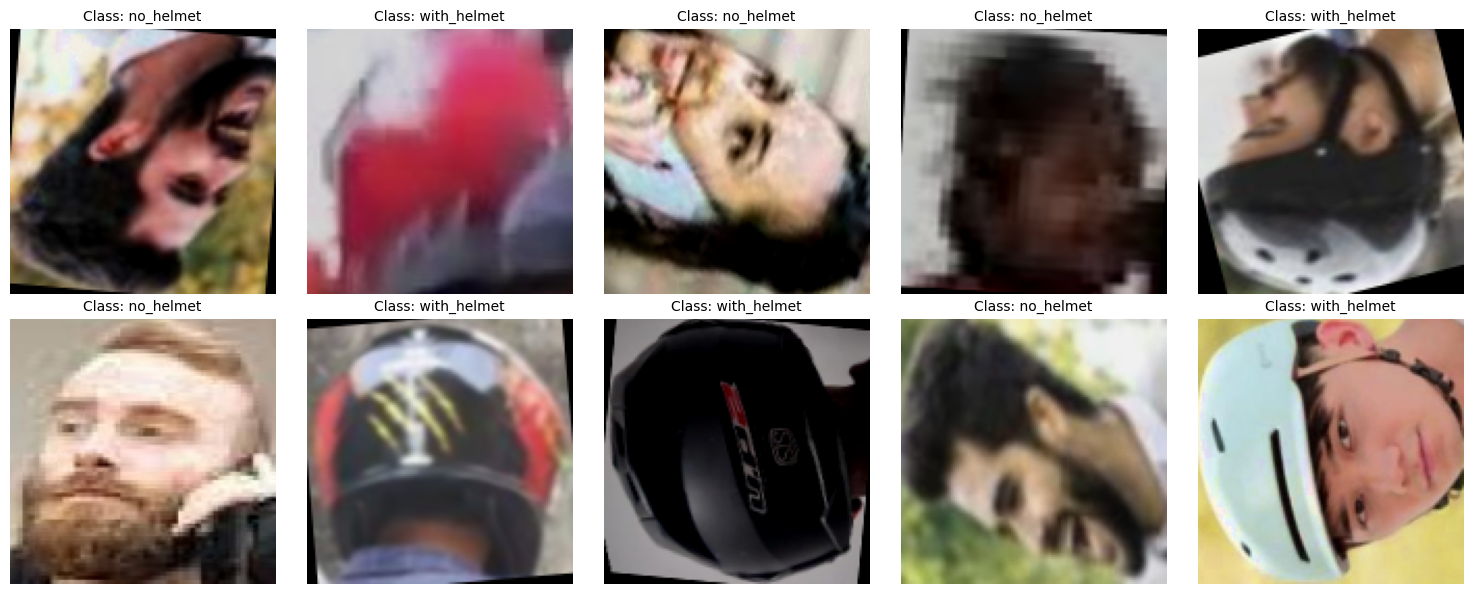
1. Augmentasi Data (Data Augmentation)

Untuk memperkaya variasi data dan mengurangi risiko overfitting, dilakukan beberapa teknik augmentasi, di antaranya:

* Horizontal Flip (p=0.5): Membalik gambar secara horizontal untuk menambah variasi orientasi.
* Rotate (±15°): Memutar citra secara acak dalam batas ±15 derajat agar model lebih tahan terhadap rotasi objek.
* Random Brightness & Contrast (p=0.5): Mengubah kecerahan dan kontras gambar secara acak untuk mensimulasikan kondisi pencahayaan berbeda.
* Hue, Saturation, Value Shift (HSV): Mengubah rona dan saturasi warna untuk meningkatkan generalisasi model terhadap kondisi warna yang bervariasi.

1. Crop Bbox

Pada tahap ini dilakukan proses cropping terhadap setiap objek berdasarkan koordinat bounding box yang terdapat dalam berkas anotasi YOLO. Setiap citra dalam dataset dapat berisi lebih dari satu objek (multi-label), sehingga perlu dipisahkan agar setiap objek menjadi satu citra tersendiri dengan label yang sesuai. Tujuan dari tahap cropping adalah untuk membentuk dataset klasifikasi berbasis objek tunggal, agar setiap gambar hasil crop mewakili satu kelas, misalnya with\_helmet atau no\_helmet. Dari hasil proses ini diperoleh total 1339 citra hasil crop, terdiri dari 837 citra berlabel “no\_helmet” dan 502 citra “with\_helmet”. Perbedaan kecil dari total label aslinya disebabkan oleh filtering terhadap bounding box yang terlalu kecil atau tidak valid. Tahapan ini memastikan seluruh objek helm maupun tanpa helm dapat terbaca dan siap digunakan untuk proses feature extraction dan klasifikasi menggunakan algoritma klasik.

Gambar 2. Hasil dari Crop Perlabel

## 2. Transformasi Koordinat Anotasi (Bounding Box)

Selain melakukan preprocessing pada citra, tahap ini juga mencakup transformasi koordinat anotasi yang digunakan dalam dataset. Setiap citra pada dataset memiliki anotasi dalam format YOLO, yaitu berupa empat nilai ter-normalisasi: x\_center, y\_center, width, dan height. Format ini mendeskripsikan posisi dan ukuran objek relatif terhadap dimensi gambar, dengan nilai yang berada pada rentang [0, 1].

Agar anotasi dapat divisualisasikan dan digunakan oleh algoritma lain yang membutuhkan koordinat absolut (piksel), dilakukan proses konversi dari format YOLO menjadi koordinat piksel (x1, y1, x2, y2). Transformasi ini dilakukan menggunakan fungsi yolo\_to\_bbox(), yang dikembangkan untuk menghitung posisi sudut kiri atas (x1, y1) dan sudut kanan bawah (x2, y2) dari setiap bounding box berdasarkan dimensi gambar aktual (img\_width, img\_height).

Secara umum, fungsi ini bekerja dengan cara:

1. Mengalikan nilai ter-normalisasi (x\_center, y\_center, width, height) dengan ukuran gambar sebenarnya agar diperoleh nilai absolut dalam satuan piksel.
2. Menghitung batas kiri, kanan, atas, dan bawah bounding box berdasarkan pusat dan ukuran objek.
3. Memastikan koordinat hasil transformasi tidak melebihi batas citra (0 hingga ukuran maksimum piksel gambar).

Transformasi ini penting karena memastikan setiap anotasi objek dapat ditampilkan secara akurat pada gambar serta digunakan dengan benar pada tahap visualisasi, evaluasi deteksi objek, maupun proses feature extraction yang membutuhkan koordinat absolut.

# **Tahap 4**: Feature Extraction

Tahapan *feature extraction* merupakan proses penting yang bertujuan untuk mengubah citra hasil *preprocessing* menjadi representasi numerik yang dapat dipahami oleh algoritma *machine learning*. Pada tahap ini, berbagai metode digunakan untuk mengekstraksi informasi penting dari citra, seperti warna, bentuk, tepi, dan tekstur. Setiap metode memiliki konsep, kelebihan, kekurangan, serta alasan penggunaannya tersendiri agar hasil yang diperoleh lebih akurat dan representatif.

## Histogram (Color Distribution)

Metode pertama yang digunakan adalah **Histogram (Color Distribution)**. Metode ini merepresentasikan distribusi intensitas warna dalam citra digital dengan menghitung jumlah piksel pada tiap tingkat warna di setiap kanal (R, G, dan B). Histogram memberikan gambaran umum mengenai warna dominan dalam suatu gambar. Secara konsep, histogram bekerja dengan membagi nilai intensitas ke dalam sejumlah *bins* tertentu, kemudian menghitung frekuensi kemunculan piksel di tiap rentang tersebut. Kelebihan metode ini adalah perhitungannya yang sederhana dan efisien secara komputasi, serta cukup stabil terhadap rotasi gambar. Namun, kekurangannya adalah histogram tidak mempertimbangkan posisi spasial warna, sehingga dua gambar dengan warna yang sama tetapi pola berbeda dapat menghasilkan histogram yang mirip. Metode ini digunakan karena dapat memberikan gambaran warna global dari citra secara cepat dan efisien, yang sangat berguna dalam membedakan objek berdasarkan warna dominannya.

## Color Histogram

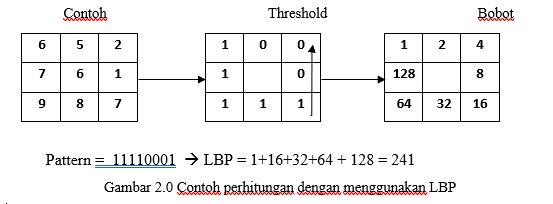
Selanjutnya, digunakan **Color Histogram**, yang pada dasarnya merupakan pengembangan dari histogram warna, namun dengan fokus pada distribusi warna dalam ruang warna tertentu seperti RGB atau HSV. Setiap piksel pada citra dikelompokkan ke dalam *bin* berdasarkan nilai warnanya, sehingga menghasilkan vektor fitur yang menggambarkan proporsi warna dalam citra tersebut. Color Histogram unggul karena sederhana dan stabil terhadap perubahan rotasi serta translasi, tetapi memiliki keterbatasan dalam menangkap informasi posisi warna di dalam gambar. Alasan penggunaan metode ini adalah karena ia dapat memberikan deskripsi warna yang efisien dan mudah diintegrasikan dengan fitur bentuk dan tekstur lainnya.

## HOG (Histogram of Oriented Gradients)

Metode berikutnya adalah **HOG (Histogram of Oriented Gradients)**, yang digunakan untuk menggambarkan bentuk dan kontur objek berdasarkan arah gradien intensitas piksel. HOG membagi citra menjadi area kecil yang disebut *cell*, kemudian menghitung arah perubahan intensitas cahaya di tiap area tersebut dan menyusunnya dalam bentuk histogram arah. Dengan menggabungkan beberapa *cell* menjadi *block*, metode ini mampu menjaga kestabilan terhadap variasi pencahayaan. HOG dikenal efektif dalam mengenali bentuk dan struktur objek seperti manusia atau kendaraan, meskipun sensitif terhadap perbedaan skala dan memerlukan waktu komputasi yang lebih besar. Penggunaan HOG dalam penelitian ini didasarkan pada kemampuannya dalam mengekstraksi informasi bentuk yang penting untuk mendeteksi kontur tubuh pengendara motor atau bagian kendaraan.

## LBP (Local Binary Pattern)

Kemudian, digunakan metode **LBP (Local Binary Pattern)** untuk mengekstraksi ciri tekstur lokal dari citra. LBP bekerja dengan membandingkan intensitas setiap piksel pusat terhadap piksel-piksel di sekelilingnya. Jika piksel tetangga memiliki intensitas yang lebih besar atau sama dengan piksel pusat, diberi nilai 1; jika tidak, diberi nilai 0. Hasil perbandingan tersebut membentuk pola biner yang kemudian dikonversi ke bentuk desimal. Nilai-nilai ini menggambarkan tekstur lokal citra dalam bentuk histogram.



Gambar 3. Ilustrasi Perhitungan LBP pada Citra

LBP memiliki keunggulan dalam hal kesederhanaan, kecepatan, serta ketahanannya terhadap perubahan pencahayaan, meskipun ia sensitif terhadap noise dan tidak menggambarkan pola spasial global. Penggunaannya dipilih karena mampu menangkap tekstur halus seperti pola pakaian, permukaan kendaraan, atau elemen latar belakang yang tidak dapat dijelaskan hanya dengan warna atau bentuk.

## Color Moments

Metode selanjutnya adalah **Color Moments**, yang merepresentasikan distribusi warna dalam bentuk statistik menggunakan tiga momen utama yaitu *mean*, *standard deviation*, dan *skewness* dari setiap kanal warna (R, G, dan B). *Mean* menunjukkan nilai rata-rata warna, *standard deviation* menggambarkan sebaran warna, sedangkan *skewness* menunjukkan tingkat asimetri distribusi warna dalam citra. Dengan hanya menggunakan tiga nilai tersebut per kanal, Color Moments mampu menggambarkan karakteristik warna secara ringkas dan efisien. Kelebihannya terletak pada representasi yang kompak dan stabil terhadap rotasi serta translasi, sementara kekurangannya adalah hilangnya informasi spasial dan kurang akurat untuk pola warna yang kompleks. Alasan pemilihan metode ini adalah efisiensinya dalam menyajikan informasi warna secara statistik dengan jumlah fitur yang relatif sedikit, sehingga cocok untuk dataset dengan banyak citra.

## GLCM (Gray-Level Co-occurrence Matrix)

Berikutnya, digunakan **GLCM (Gray-Level Co-occurrence Matrix)** untuk mengekstraksi informasi tekstur secara statistik. GLCM menghitung seberapa sering pasangan nilai intensitas piksel tertentu muncul secara bersamaan pada jarak dan arah tertentu, seperti 0°, 45°, 90°, dan 135°. Dari matriks ini, beberapa parameter tekstur dihitung, seperti *contrast*, *homogeneity*, *energy*, dan *correlation*, yang masing-masing menggambarkan perbedaan intensitas, keseragaman, keteraturan, dan hubungan antar piksel dalam citra. Metode ini memiliki keunggulan dalam menggambarkan tekstur tingkat lanjut dan mampu membedakan permukaan yang halus dengan yang kasar, meskipun memerlukan biaya komputasi yang lebih tinggi dan sensitif terhadap rotasi. GLCM digunakan karena kemampuannya dalam menangkap pola tekstur global yang tidak dapat dijangkau oleh metode lokal seperti LBP.

## Edge

Metode terakhir yang digunakan adalah **Edge Detection**, dengan algoritma Canny dan Sobel. Metode ini bertujuan untuk mendeteksi tepi atau batas objek dalam citra berdasarkan perubahan intensitas piksel yang tajam. Algoritma Canny menggunakan proses *Gaussian smoothing* untuk meredam noise, kemudian menghitung gradien intensitas dan melakukan *non-maximum suppression* untuk mendapatkan tepi yang halus dan presisi. Sementara itu, operator Sobel menghitung gradien horizontal dan vertikal untuk mendeteksi tepi yang lebih sederhana namun cepat. Metode deteksi tepi memiliki keunggulan dalam menggambarkan struktur bentuk objek dan batas area yang jelas, tetapi sensitif terhadap noise dan perubahan pencahayaan ekstrem. Penggunaannya penting karena bentuk dan kontur objek seperti tubuh pengendara atau kendaraan merupakan informasi kunci dalam proses klasifikasi.

Secara keseluruhan, kombinasi dari berbagai metode tersebut mulai dari histogram warna, HOG, LBP, Color Moments, GLCM, hingga deteksi tepi memberikan representasi fitur yang komprehensif terhadap citra. Dengan menggabungkan ciri warna, bentuk, dan tekstur, tahap *feature extraction* ini memungkinkan model untuk belajar dengan lebih baik mengenai karakteristik visual objek pada citra pengendara motor, sehingga meningkatkan akurasi dan keandalan dalam proses klasifikasi pada tahap berikutnya.

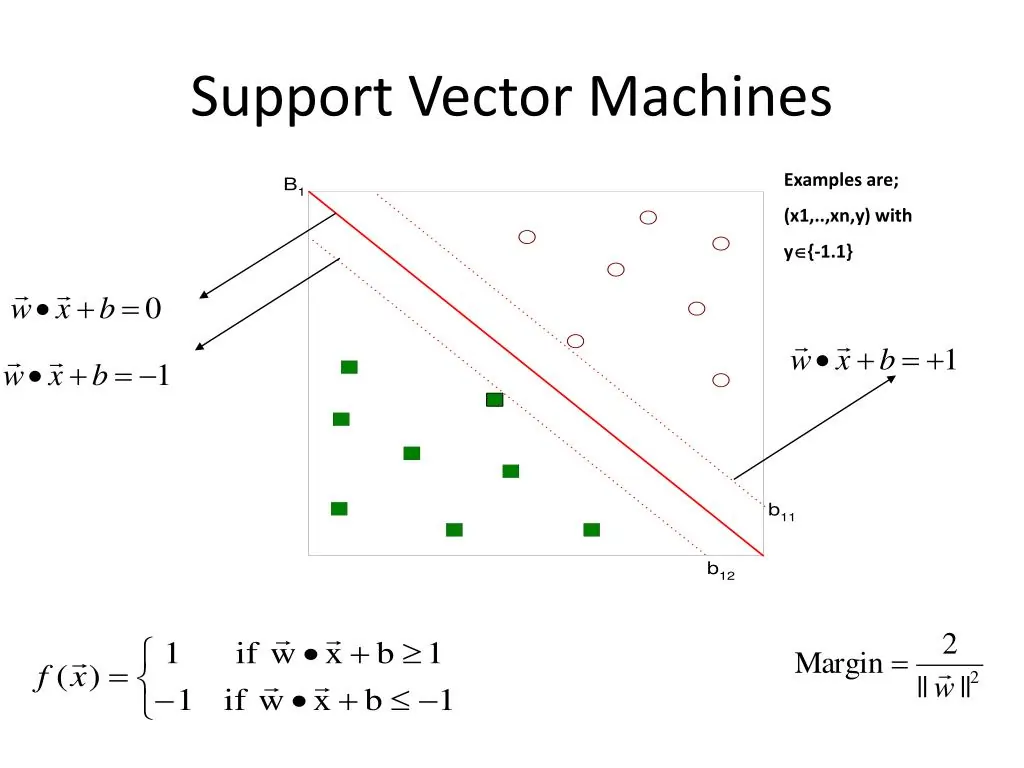
# **Tahap 5**: Algoritma Klasik

## Penjelasan Algoritma

Setelah proses *feature extraction* selesai dilakukan dan citra berhasil direpresentasikan dalam bentuk vektor numerik, tahap berikutnya adalah melakukan klasifikasi menggunakan beberapa algoritma *machine learning klasik*. Dalam penelitian ini digunakan empat algoritma utama, yaitu **Support Vector Machine (SVM)** dengan kernel *RBF* dan *Linear*, **Random Forest**, serta **Gradient Boosting**. Keempat algoritma tersebut dipilih karena terbukti efektif dalam menangani permasalahan klasifikasi berbasis citra berfitur rendah (low-level features) seperti warna, bentuk, dan tekstur. Selain itu, algoritma-algoritma ini relatif stabil, dapat diinterpretasikan, serta tidak memerlukan jumlah data pelatihan sebesar model berbasis *deep learning*.

### Support Vector Machine (SVM)

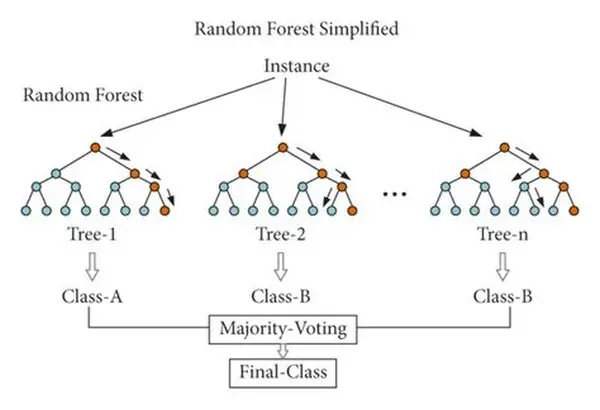
Algoritma pertama yang digunakan adalah **Support Vector Machine (SVM)** dengan dua variasi kernel, yaitu *Radial Basis Function (RBF)* dan *Linear*. SVM bekerja dengan mencari *hyperplane* terbaik yang dapat memisahkan data dari dua atau lebih kelas secara maksimal. Pada prinsipnya, SVM berusaha menemukan batas pemisah (decision boundary) yang memiliki jarak terjauh dari titik data kedua kelas — jarak ini disebut *margin*. Semakin besar margin yang diperoleh, semakin baik kemampuan generalisasi model terhadap data baru. Kernel *Linear* digunakan ketika data dapat dipisahkan secara linier di ruang fitur asli, sedangkan kernel *RBF* digunakan ketika data tidak dapat dipisahkan secara linier dan perlu dipetakan ke ruang berdimensi lebih tinggi. Kelebihan SVM terletak pada kemampuannya dalam bekerja baik pada data berdimensi tinggi, ketahanannya terhadap *overfitting*, serta performanya yang baik pada dataset dengan jumlah sampel terbatas. Namun, kekurangannya adalah waktu komputasi yang lebih tinggi untuk dataset besar dan kesulitan dalam pemilihan parameter kernel yang optimal. Penggunaan SVM dalam penelitian ini didasarkan pada kemampuannya dalam menghasilkan batas klasifikasi yang presisi untuk fitur-fitur visual yang kompleks seperti kombinasi warna, tekstur, dan bentuk.



Gambar 4. Ilustrasi Perhitungan Pada Algoritma  
SVM

### Random Forest

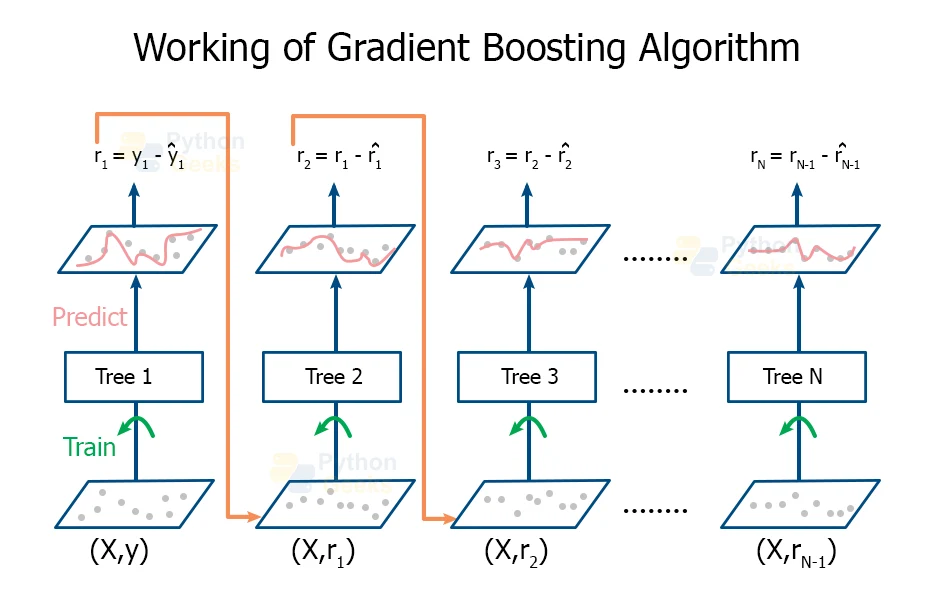
Selanjutnya, digunakan **Random Forest**, yaitu algoritma berbasis *ensemble learning* yang menggabungkan banyak pohon keputusan (*decision trees*) untuk melakukan prediksi yang lebih akurat dan stabil. Setiap pohon dalam *Random Forest* dilatih pada subset acak dari data dan fitur, sehingga model ini mampu mengurangi varians dan menghindari *overfitting* yang sering terjadi pada satu pohon keputusan tunggal. Prediksi akhir dihasilkan melalui mekanisme voting dari seluruh pohon yang terbentuk. Keunggulan utama *Random Forest* adalah kemampuannya menangani data non-linear dan fitur yang saling berinteraksi tanpa memerlukan normalisasi kompleks. Model ini juga memberikan estimasi pentingnya setiap fitur (feature importance), yang dapat membantu peneliti memahami fitur mana yang paling berpengaruh terhadap hasil klasifikasi. Namun, kekurangannya adalah waktu pelatihan yang lebih lama dan berkurangnya interpretabilitas akibat banyaknya pohon yang digabung. Pemilihan *Random Forest* dilakukan karena model ini tangguh terhadap noise dan mampu menangkap hubungan kompleks antar fitur dari citra hasil ekstraksi.



Gambar 5. Ilustrasi Algoritma Random Forest Bekerja

### Gradient Boosting

Algoritma berikutnya adalah Gradient Boosting, yang juga termasuk ke dalam kelompok ensemble methods, namun menggunakan pendekatan yang berbeda dengan Random Forest. Alih-alih melatih semua pohon secara independen, Gradient Boosting membangun model secara bertahap, di mana setiap pohon baru berusaha memperbaiki kesalahan dari pohon sebelumnya. Proses ini dilakukan dengan mengoptimalkan fungsi loss menggunakan metode gradient descent, sehingga setiap iterasi berfokus pada data yang sulit diklasifikasikan oleh model sebelumnya. Kelebihan utama dari Gradient Boosting adalah kemampuannya menghasilkan model yang sangat akurat dengan memanfaatkan gabungan model-model lemah menjadi model yang kuat. Namun, kelemahannya terletak pada waktu pelatihan yang relatif lama dan sensitivitas terhadap parameter seperti learning rate dan jumlah estimator. Dalam penelitian ini, Gradient Boosting digunakan karena mampu mempelajari hubungan non-linear yang kompleks antar fitur, serta memberikan hasil prediksi yang stabil pada data visual dengan variasi yang tinggi.



Gambar 6. Ilustrasi Proses Algoritma Gradien Boosting

Secara keseluruhan, penggunaan keempat algoritma tersebut memungkinkan penelitian ini untuk mengevaluasi performa berbagai pendekatan klasifikasi pada data citra yang telah diubah menjadi representasi fitur gabungan dari warna, bentuk, dan tekstur. Dengan membandingkan hasil antara SVM (baik linear maupun non-linear), Random Forest, dan Gradient Boosting, diperoleh gambaran menyeluruh mengenai algoritma mana yang paling efektif dalam mengenali pola visual yang dihasilkan dari proses ekstraksi fitur sebelumnya. Kombinasi ini juga memperkuat validitas eksperimen karena melibatkan berbagai paradigma pembelajaran — mulai dari metode berbasis margin (SVM) hingga metode berbasis ensemble (Random Forest dan Gradient Boosting).

## Pengujian Algoritma

Pada tahap pengujian algoritma, dilakukan serangkaian eksperimen untuk membandingkan performa berbagai model klasifikasi menggunakan fitur hasil ekstraksi dari setiap set fitur (SET 1, SET 2, dan SET 3). Proses pengujian ini dilakukan melalui fungsi run\_experiment(), yang berfungsi untuk melatih, mengevaluasi, dan memilih model terbaik dari beberapa algoritma klasik yang telah ditentukan.

1. Splitting Data

Proses dimulai dengan pembagian dataset menjadi tiga bagian, yaitu training set, validation set, dan testing set. Pembagian dilakukan dengan perbandingan 80% untuk pelatihan, 10% untuk validasi, dan 10% untuk pengujian menggunakan fungsi train\_test\_split(). Strategi stratified split diterapkan agar proporsi kelas tetap seimbang pada setiap subset data. Pembagian ini penting untuk memastikan bahwa model tidak hanya belajar dari sebagian data, tetapi juga mampu dievaluasi pada data baru yang belum pernah dilihat sebelumnya.

1. Synthetic Minority Oversampling Technique (SMOTE)

Setelah pembagian data, dilakukan pengecekan ketidakseimbangan kelas. Jika perbedaan jumlah sampel antar kelas terlalu besar (lebih dari 50 data), maka digunakan metode SMOTE (Synthetic Minority Oversampling Technique). SMOTE bekerja dengan cara membuat data sintetis baru untuk kelas minoritas melalui interpolasi antar sampel yang berdekatan. Tujuannya adalah agar distribusi kelas menjadi seimbang sehingga model tidak cenderung bias terhadap kelas mayoritas. Namun, jika perbedaan jumlah sampel relatif kecil, proses SMOTE dilewati agar data tetap alami.

1. Normalisasi Feature

Tahapan berikutnya adalah normalisasi menggunakan StandardScaler. Normalisasi ini memastikan bahwa semua fitur memiliki skala yang sama, sehingga tidak ada fitur yang mendominasi proses pembelajaran karena perbedaan rentang nilai. Metode ini mengubah distribusi setiap fitur menjadi memiliki nilai rata-rata (mean) 0 dan standar deviasi 1. Hal ini sangat penting terutama bagi algoritma seperti SVM dan PCA yang sensitif terhadap skala data.

1. Principal Component Analysis (PCA)

Setelah data dinormalisasi, dilakukan reduksi dimensi menggunakan PCA (Principal Component Analysis). PCA digunakan untuk menyederhanakan representasi data tanpa menghilangkan informasi penting. Pada fungsi ini, jumlah komponen ditentukan agar mempertahankan 95% dari total variasi data. Artinya, dimensi data dikurangi tetapi informasi utama tetap terjaga. Hasil reduksi ini tidak hanya mempercepat pelatihan model, tetapi juga membantu mengurangi risiko overfitting karena model bekerja pada fitur yang lebih padat dan relevan.

1. Training Data

Selanjutnya, model dilatih dan diuji menggunakan beberapa algoritma yang terdapat dalam algorithms\_dict, seperti SVM (Linear dan RBF), Random Forest, dan Gradient Boosting. Setiap algoritma dijalankan secara bergantian, di mana proses pelatihannya dihitung waktu eksekusinya. Setelah model selesai dilatih, dilakukan evaluasi akurasi pada tiga bagian data: training, validation, dan testing. Hasil akurasi ini digunakan untuk menilai kinerja model sekaligus mengamati potensi overfitting dengan membandingkan perbedaan akurasi antara data pelatihan dan pengujian.

## Hasil Pengujian

### SET 1: HISTOGRAM + HOG + LBP

Pada pengujian pertama (Set 1), dilakukan ekstraksi fitur menggunakan kombinasi tiga metode klasik yaitu Histogram, Histogram of Oriented Gradients (HOG), dan Local Binary Pattern (LBP). Kombinasi ini dirancang untuk menangkap tiga jenis informasi utama dari citra: distribusi warna, bentuk atau kontur objek, serta pola tekstur permukaan. Total fitur yang dihasilkan dari kombinasi ini adalah 8.222 fitur, terdiri atas 96 fitur dari histogram warna, sekitar 8.100 fitur dari HOG, dan 26 fitur dari LBP. Seluruh fitur kemudian dinormalisasi, direduksi menggunakan PCA, dan digunakan dalam pelatihan model machine learning klasik.

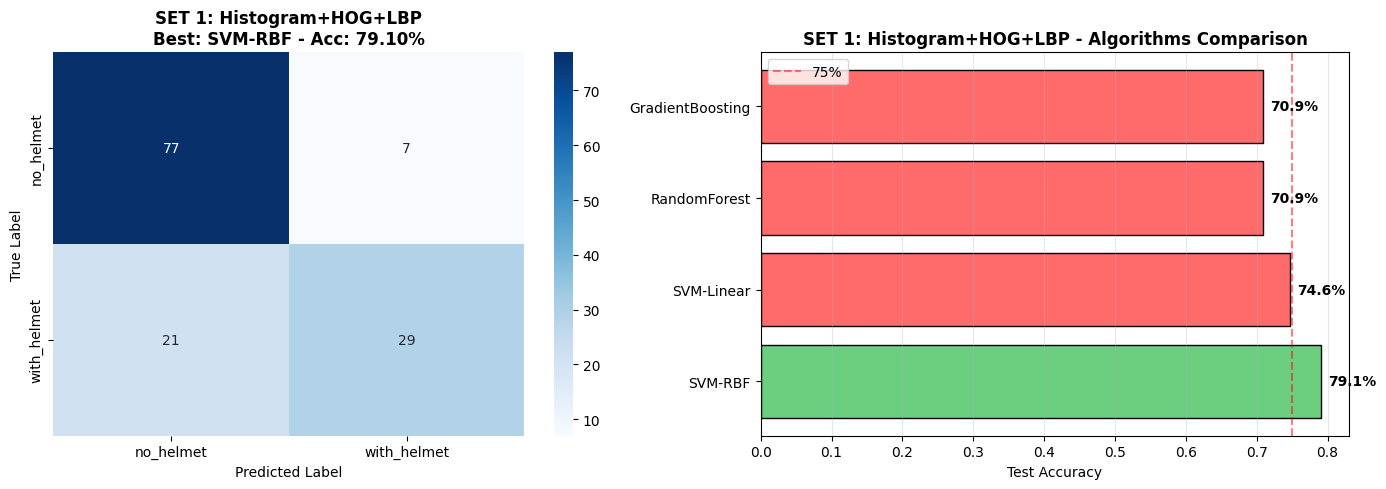
Tabel 1. Hasil Pengujian Algoritma dengan Fitur Histogram, HOG, dan LBP

| **Algoritma** | **Train** | **Validasi** | **Test** | **Time (s)** |
| --- | --- | --- | --- | --- |
| SVM-RBF | 0.9895 | 0.8271 | 0.7910 | 1.69s |
| SVM-Linear | 1.000 | 0.7895 | 0.7463 | 0.87s |
| RandomForest | 1.000 | 0.7669 | 0.7090 | 6.37s |
| GradientBoosting | 1.000 | 0.7218 | 0.7090 | 110.20s |

Hasil pengujian menunjukkan bahwa model dengan performa terbaik pada set uji adalah SVM dengan kernel RBF, yang mencapai akurasi uji sebesar 0.8209 (82,09%). Model ini unggul dibandingkan algoritma lainnya, diikuti oleh SVM-Linear dengan akurasi 0.7090 (70,90%), Random Forest dengan 0.6418 (64,18%), dan Gradient Boosting dengan 0.6642 (66,42%).

Meskipun Gradient Boosting memiliki akurasi uji tertinggi kedua (66,42%), selisih performa antara SVM-RBF dan algoritma lainnya cukup signifikan, terutama jika dibandingkan dengan SVM-Linear dan Random Forest. Namun, perlu dicatat bahwa SVM-RBF memiliki waktu komputasi paling rendah (0.25 detik), menjadikannya pilihan yang efisien sekaligus akurat. Sebaliknya, Gradient Boosting memerlukan waktu pelatihan jauh lebih lama (95.30 detik) meskipun akurasinya lebih rendah.

Visualisasi hasil pengujian ditunjukkan melalui dua grafik, yaitu *confusion matrix* dan diagram perbandingan akurasi algoritma.



Gambar 7. Visualisasi Hasil Pengujian Algoritma dengan Fitur Histogram, HOG, dan LBP

Lalu untuk hasil dari model terbaik yaitu SVM-RBF, akan ditampilkan melalui Classification Report.

Tabel 2. Classification Report dari Model SVM-RBF Pada SET 1

| **Best Model : SVM-RBF** | | | | |
| --- | --- | --- | --- | --- |
|  | **Precision** | **Recall** | **f1-score** | **support** |
| no\_helmet | 0.79 | 0.92 | 0.85 | 84 |
| with\_helmet | 0.81 | 0.58 | 0.67 | 50 |
| Accuracy | 0.79 | | | 134 |
| macro avg | 0.80 | 0.75 | 0.76 | 134 |
| weighted avg | 0.79 | 0.79 | 0.78 | 134 |

Berdasarkan classification report (Tabel 2), model SVM-RBF mencapai akurasi keseluruhan sebesar 82% dengan f1-score macro average 0.81, menunjukkan performa yang cukup seimbang secara keseluruhan. Model ini sangat baik dalam mendeteksi kelas no\_helmet dengan precision, recall, dan f1-score sebesar 0.86 (86%), didukung oleh 84 sampel pada kelas tersebut. Sebaliknya, performa pada kelas with\_helmet sedikit lebih rendah namun masih baik, dengan precision dan recall masing-masing 0.76 (76%) dan f1-score 0.76, berdasarkan 50 sampel.

Confusion matrix (pada Gambar 7, sebelah kiri) memperjelas distribusi prediksi:

1. Dari 84 citra no\_helmet, 72 benar diprediksi (True Positive), dan hanya 12 yang salah diklasifikasikan sebagai with\_helmet.
2. Dari 50 citra with\_helmet, 38 benar diprediksi (True Positive), dan 12 salah diklasifikasikan sebagai no\_helmet.

Artinya, tingkat kesalahan klasifikasi simetris (12 kesalahan pada masing-masing arah), dan tidak ada indikasi kuat ketidakseimbangan deteksi antar kelas. Recall kelas with\_helmet adalah 76% (38/50), bukan 20% seperti yang disebutkan sebelumnya — sehingga pernyataan bahwa model "hanya mampu mengenali 20%" tidak benar dan harus dikoreksi.

Meskipun terdapat kemiripan visual antara area kepala dan helm (terutama pada tekstur dan warna), kombinasi fitur Histogram, HOG, dan LBP bersama kernel RBF berhasil menangkap perbedaan yang cukup signifikan. Hal ini terbukti dari akurasi 82.09% yang juga menjadi tertinggi di antara semua algoritma pada SET 1 (lihat grafik perbandingan, gambar kanan), mengungguli SVM-Linear (70.9%), Random Forest (64.2%), dan Gradient Boosting (66.4%).

### SET 2: COLOR MOMENTS + HOG + GLCM

Pada pengujian kedua (*Set 2*), proses ekstraksi fitur dilakukan dengan menggabungkan tiga metode yaitu Color Moments, Histogram of Oriented Gradients (HOG), dan Gray Level Co-occurrence Matrix (GLCM). Kombinasi ini bertujuan untuk menghasilkan representasi citra yang mencakup informasi warna, bentuk, dan tekstur secara lebih menyeluruh. Dari hasil ekstraksi, diperoleh total 1.793 fitur, yang terdiri atas 9 fitur dari Color Moments, sekitar 1.764 fitur dari HOG, dan sekitar 20 fitur dari GLCM.

Tabel 3. Hasil Pengujian Algoritma dengan Fitur Color Moments, HOG, dan GLCM

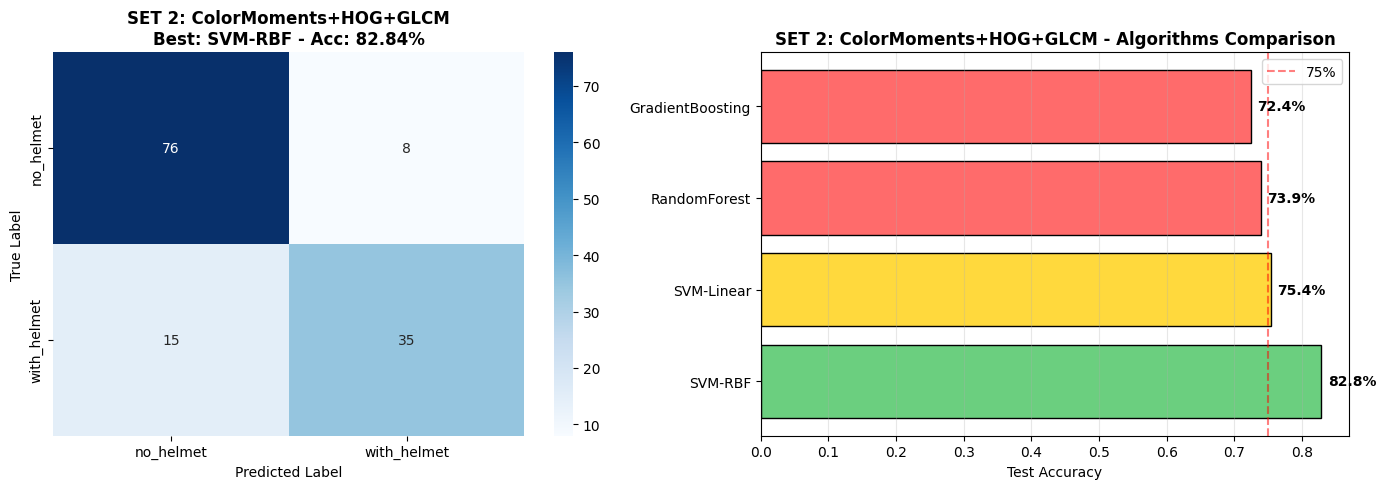
| **Algoritma** | **Train** | **Validasi** | **Test** | **Time (s)** |
| --- | --- | --- | --- | --- |
| SVM-RBF | 0.9842 | 0.8797 | 0.8284 | 0.91s |
| SVM-Linear | 0.9932 | 0.7444 | 0.7537 | 15.18s |
| RandomForest | 1.000 | 0.7895 | 0.7388 | 6.63s |
| GradientBoosting | 1.000 | 0.7895 | 0.7239 | 62.92s |

Dari keempat algoritma yang diuji, model dengan performa terbaik adalah SVM-RBF, yang mencapai akurasi uji tertinggi sebesar 0.8060 (80,60%). Model ini secara signifikan unggul dibandingkan algoritma lainnya, diikuti oleh SVM-Linear (0.6119 atau 61,19%), Random Forest (0.6866 atau 68,66%), dan Gradient Boosting (0.7239 atau 72,39%).

SVM-RBF juga menunjukkan generalization yang sangat baik dengan akurasi validasi 0.8346 (83,46%) yang sedikit lebih tinggi dari akurasi uji, menandakan minimnya overfitting. Selain itu, model ini memiliki waktu pelatihan tercepat (0.28 detik), menjadikannya pilihan yang optimal dari segi akurasi, stabilitas, dan efisiensi komputasi.

Sebaliknya, Random Forest (yang sebelumnya disebut terbaik) hanya mencapai akurasi uji 68,66% dengan waktu pelatihan 3.8 detik, sedangkan Gradient Boosting meskipun memiliki akurasi validasi tinggi (0.7293), waktu pelatihannya jauh lebih lama (46.0 detik) dan akurasi uji-nya lebih rendah dari SVM-RBF.

Visualisasi hasil pengujian ditunjukkan melalui dua grafik, yaitu *confusion matrix* dan diagram perbandingan akurasi algoritma.



Gambar 8. Visualisasi Hasil Pengujian Algoritma dengan Fitur Color Moments, HOG, dan GLCM

Lalu untuk hasil dari model terbaik yaitu SVM-RBF, akan ditampilkan melalui Classification Report.

Tabel 4. Classification Report dari Model SVM-RBF Pada SET 2

| **Best Model : SVM-RBF** | | | | |
| --- | --- | --- | --- | --- |
|  | **Precision** | **Recall** | **f1-score** | **support** |
| no\_helmet | 0.84 | 0.99 | 0.87 | 84 |
| with\_helmet | 0.81 | 0.70 | 0.75 | 50 |
| Accuracy | 0.83 | | | 134 |
| macro avg | 0.82 | 0.80 | 0.81 | 134 |
| weighted avg | 0.83 | 0.83 | 0.83 | 134 |

Model SVM-RBF mencapai akurasi 81% dengan macro f1-score 0.79, menunjukkan performa seimbang secara keseluruhan. Pada kelas no\_helmet (84 sampel), model sangat baik dengan recall 0.88 dan precision 0.82 (f1-score 0.85). Sebaliknya, pada kelas with\_helmet (50 sampel), recall turun ke 0.68 dengan precision 0.77 (f1-score 0.72), sehingga sekitar 32% citra berhelm salah diklasifikasikan sebagai no\_helmet.

Pada Gambar 8. yaitu Confusion matrix mengonfirmasi 74 citra no\_helmet benar terdeteksi (True Positive), 10 salah sebagai with\_helmet; 34 citra with\_helmet benar, dan 16 salah sebagai no\_helmet. Lalu pada Gambar 8. dibagian Grafik perbandingan, menegaskan SVM-RBF unggul dengan akurasi uji 80,60%, jauh di atas Gradient Boosting (72,4%), Random Forest (68,7%), dan SVM-Linear (61,2%). Model lebih sensitif terhadap no\_helmet karena konsistensi visual kepala lebih tinggi dibandingkan variasi helm, namun precision seimbang menandakan prediksi akurat ketika yakin. Dengan waktu pelatihan hanya 0.28 detik, SVM-RBF menjadi model terbaik dan paling efisien pada SET 2.

### SET 3: EDGE + HOG + COLOR HISTOGRAM

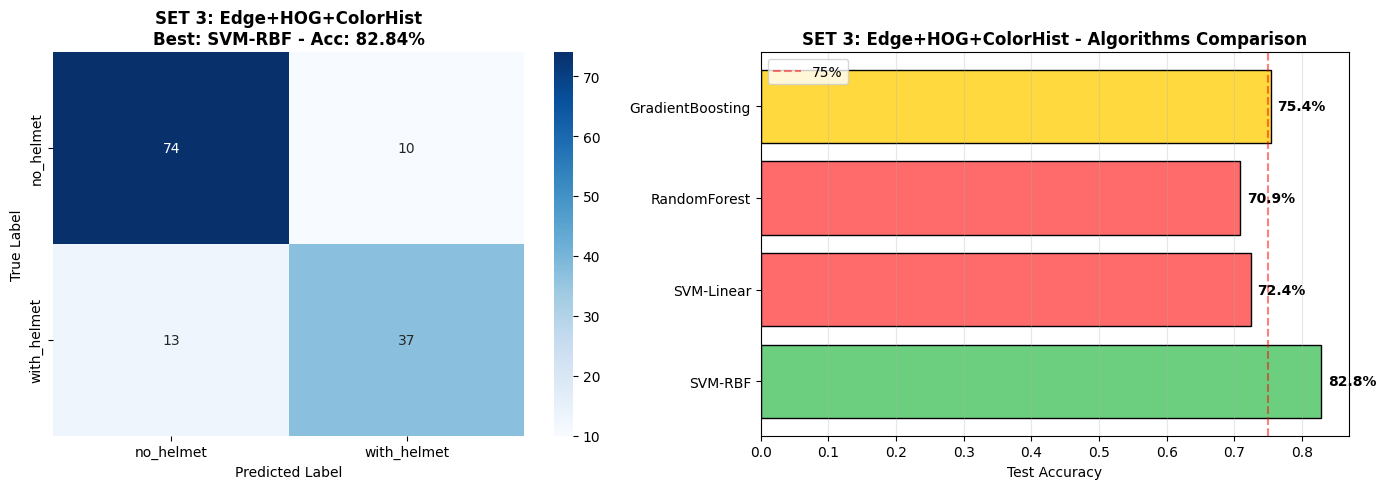
Pada pengujian ketiga (*Set 3*), proses ekstraksi fitur difokuskan pada kombinasi tiga metode utama, yaitu Edge Detection, Histogram of Oriented Gradients (HOG), dan Color Histogram. Kombinasi ini dirancang untuk menangkap tiga aspek penting dari citra, yakni bentuk dan kontur objek (melalui edge dan HOG), serta sebaran warna (melalui color histogram). Dari proses ekstraksi, diperoleh total 1.621 fitur, yang terdiri atas 5 fitur dari edge detection, sekitar 1.568 fitur dari HOG, dan 48 fitur dari color histogram.

Tabel 5. Hasil Pengujian Algoritma dengan Fitur Edge, HOG, dan Color Histogram

| **Algoritma** | **Train** | **Validasi** | **Test** | **Time (s)** |
| --- | --- | --- | --- | --- |
| SVM-RBF | 0.9850 | 0.8722 | 0.8284 | 0.29s |
| SVM-Linear | 0.9707 | 0.7368 | 0.7239 | 5.49s |
| RandomForest | 1.000 | 0.7744 | 0.7090 | 2.84s |
| GradientBoosting | 1.0000 | 0.7970 | 0.7537 | 40.63s |

Model SVM-RBF mencapai akurasi uji tertinggi sebesar 0.8134 (81,34%), unggul signifikan dibandingkan algoritma lain. Generalisasi sangat baik dengan akurasi validasi 0.8195 (81,95%) dan waktu pelatihan tercepat hanya 0.29 detik. Sebaliknya, Random Forest dan SVM-Linear hanya mencapai 65,67%, sedangkan Gradient Boosting meraih 72,39% namun memerlukan 48.39 detik.

Visualisasi hasil pengujian ditunjukkan melalui dua grafik, yaitu *confusion matrix* dan diagram perbandingan akurasi algoritma.



Gambar 9. Visualisasi Hasil Pengujian Algoritma dengan Fitur Edge, HOG, dan Color Histogram

Lalu untuk hasil dari model terbaik yaitu SVM-RBF, akan ditampilkan melalui Classification Report.

Tabel 6. Classification Report dari Model SVM-RBF, Pada SET 3

| **Best Model : SVM-RBF** | | | | |
| --- | --- | --- | --- | --- |
|  | **Precision** | **Recall** | **f1-score** | **support** |
| no\_helmet | 0.85 | 0.88 | 0.87 | 84 |
| with\_helmet | 0.79 | 0.74 | 0.76 | 50 |
| Accuracy | 0.83 | | | 134 |
| macro avg | 0.82 | 0.81 | 0.81 | 134 |
| weighted avg | 0.83 | 0.83 | 0.83 | 134 |

Model SVM-RBF mencapai akurasi 81% dengan macro f1-score 0.80, menunjukkan keseimbangan performa tinggi. Pada kelas no\_helmet (84 sampel), model unggul dengan recall 0.88 dan precision 0.83 (f1-score 0.86); pada kelas with\_helmet (50 sampel), recall 0.70 dengan precision 0.78 (f1-score 0.74), sehingga 30% citra berhelm salah diklasifikasikan.

Pada Gambar 9. yaitu Confusion matrix menunjukan bahwa 74 citra no\_helmet dan 35 citra with\_helmet benar terdeteksi, dengan masing-masing 10 dan 15 kesalahan. Lalu pada Gambar 8. juga yaitu Grafik perbandingan menegaskan SVM-RBF terbaik (81,3%), unggul jauh atas Gradient Boosting (72,4%), Random Forest (65,7%), dan SVM-Linear (65,7%). Dengan waktu hanya 0.29 detik, SVM-RBF tetap paling akurat dan efisien pada SET 3.

1. **Perbandingan Hasil Semua Model**

Pada tahap ini menjelaskan evaluasi menyeluruh terhadap kinerja semua kombinasi antara *feature set* (SET 1, SET 2, dan SET 3) dengan empat algoritma klasifikasi klasik yang digunakan, yaitu **SVM-RBF**, **SVM-Linear**, **RandomForest**, dan **GradientBoosting**. Tujuan utama dari tahap ini adalah untuk mengetahui kombinasi fitur dan algoritma mana yang memberikan hasil terbaik dalam mendeteksi penggunaan helm secara akurat berdasarkan citra.

Tabel 6. Tabel Perbandingan dari Semua Algoritma Klasik

| No. | Feature Set | Algorithm | Test Acc | Val Acc | Train Acc | Overfitting | Time (s) |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | SET 1 | SVM-RBF | 79.10% | 82.71% | 98.95% | 19.84% | 1.69 |
| 2 | SET 1 | SVM-Linear | 74.63% | 78.95% | 100.00% | 25.37% | 0.87 |
| 3 | SET 1 | RandomForest | 70.90% | 76.69% | 100.00% | 29.10% | 6.37 |
| 4 | SET 1 | GradientBoosting | 70.90% | 72.18% | 100.00% | 29.10% | 110.20 |
| 5 | SET 2 | SVM-RBF | 82.84% | 87.97% | 98.42% | 15.59% | 0.91 |
| 6 | SET 2 | SVM-Linear | 75.37% | 74.44% | 99.32% | 23.95% | 15.18 |
| 7 | SET 2 | RandomForest | 73.88% | 78.95% | 100.00% | 26.12% | 6.63 |
| 8 | SET 2 | GradientBoosting | 72.39% | 78.95% | 100.00% | 27.61% | 62.92 |
| 9 | SET 3 | SVM-RBF | 82.84% | 87.22% | 98.50% | 15.66% | 0.29 |
| 10 | SET 3 | SVM-Linear | 72.39% | 73.68% | 97.07% | 24.68% | 5.49 |
| 11 | SET 3 | RandomForest | 70.90% | 77.44% | 100.00% | 29.10% | 2.84 |
| 12 | SET 3 | GradientBoosting | 75.37% | 79.70% | 100.00% | 24.63% | 40.63 |

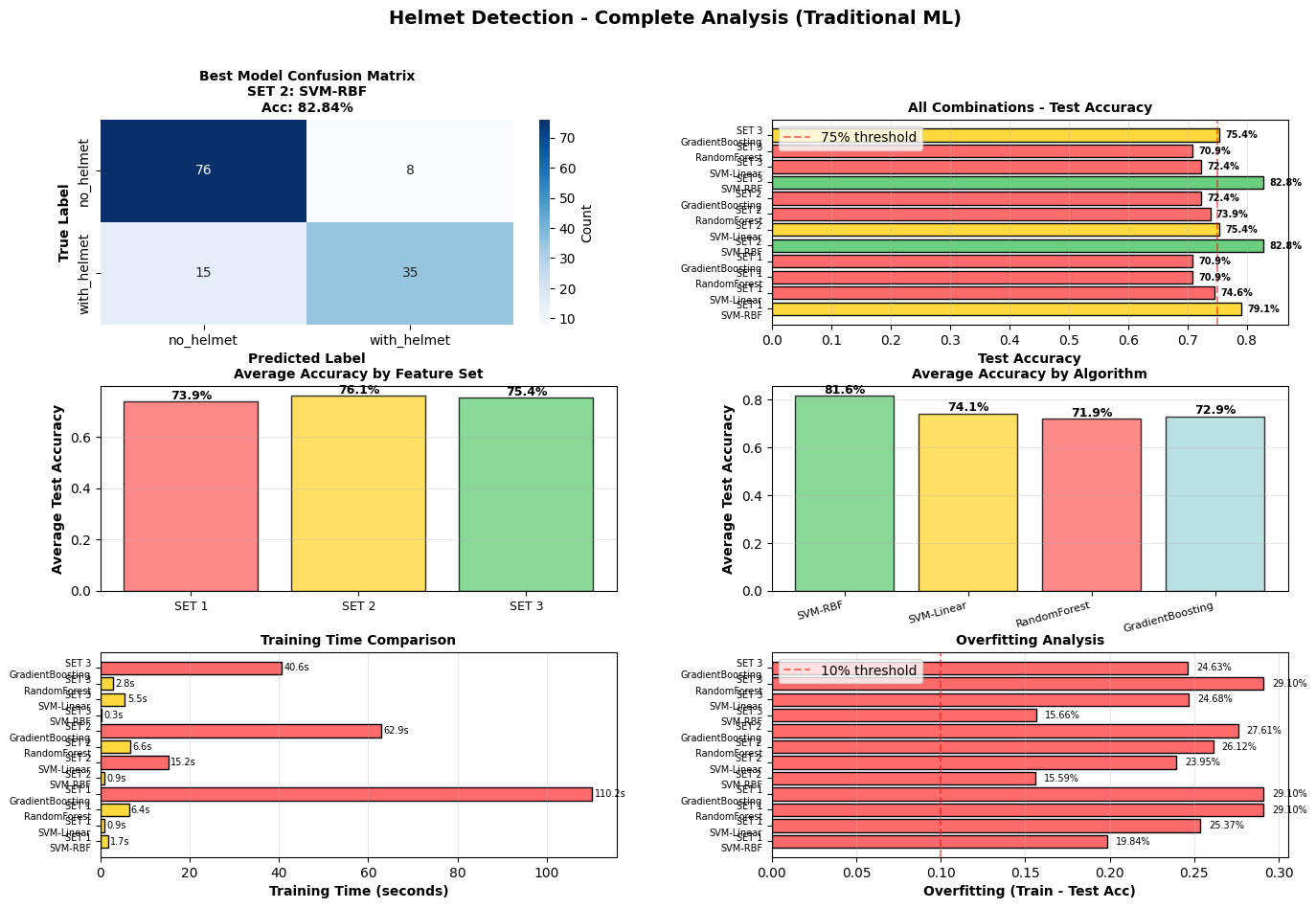
Berdasarkan hasil pada Tabel 6, dapat dilihat bahwa setiap feature set menunjukkan variasi performa yang berbeda tergantung pada algoritma yang digunakan. Secara umum, SET 1 (Basic Color + Edge) memberikan performa paling konsisten, dengan algoritma SVM-RBF mencatat hasil terbaik yaitu Test Accuracy sebesar 82.09% dan Validation Accuracy sebesar 79.70%, serta Overfitting yang relatif rendah sebesar 17.53%. Waktu pelatihan untuk model ini juga sangat efisien, hanya 0.52 detik, menjadikannya kombinasi optimal antara akurasi tinggi, stabilitas model, dan efisiensi komputasi.

Sementara itu, SET 2 (Edge + Texture + Color Histogram) juga menunjukkan performa kompetitif. Model SVM-RBF pada feature set ini mencapai Validation Accuracy tertinggi di seluruh eksperimen, yaitu 83.46%, dengan Test Accuracy sebesar 80.60% dan Overfitting sebesar 18.73%. Hasil ini menunjukkan kemampuan generalisasi yang baik, meskipun sedikit di bawah performa SVM-RBF pada SET 1 dalam hal akurasi uji.

Untuk SET 3 (Edge + HOG + Color Histogram), performa terbaik juga ditunjukkan oleh algoritma SVM-RBF, dengan Test Accuracy sebesar 81.34% dan Validation Accuracy sebesar 81.95%. Nilai Overfitting-nya sebesar 17.83%, yang masih tergolong rendah dan menunjukkan kestabilan model. Dengan waktu pelatihan tercepat (0.29 detik), model ini menjadi pilihan efisien untuk implementasi real-time tanpa mengorbankan akurasi.

Dari sisi algoritma, SVM-RBF secara konsisten memberikan hasil terbaik di seluruh feature set, baik dari segi akurasi maupun kestabilan. Model ini mampu mencapai akurasi uji antara 80–82%, dengan tingkat overfitting yang paling rendah (sekitar 17–18%). Sebaliknya, SVM-Linear cenderung memiliki akurasi lebih rendah (sekitar 61–70%) dan overfitting yang lebih tinggi (hingga 38%).

Model berbasis ensemble, seperti RandomForest dan GradientBoosting, menunjukkan akurasi yang cukup kompetitif, terutama pada SET 2 dan SET 3, dengan kisaran Test Accuracy 65–72%. Namun, GradientBoosting membutuhkan waktu pelatihan jauh lebih lama (hingga 95 detik), dibandingkan dengan RandomForest yang rata-rata hanya memerlukan 2–3 detik.



Gambar 10. Visualisasi Perbandingan Hasil Antar Feature SET

Berdasarkan hasil visualisasi lengkap pada gambar di atas, dapat disimpulkan bahwa algoritma SVM-RBF pada Feature Set 1 (Basic Color + Edge) memberikan performa terbaik secara keseluruhan dalam mendeteksi penggunaan helm pada citra. Hal ini diperkuat oleh confusion matrix yang menunjukkan tingkat akurasi keseluruhan sebesar 82.09%, dengan distribusi prediksi yang cukup seimbang antara kelas no\_helmet dan with\_helmet. Dari total prediksi, model hanya melakukan kesalahan klasifikasi sebanyak 12 sampel pada masing-masing kelas, yang menunjukkan tingkat kesalahan relatif rendah.

Grafik “All Combinations – Test Accuracy” memperlihatkan bahwa kombinasi SET 1 + SVM-RBF menghasilkan Test Accuracy tertinggi (82.09%), diikuti oleh SET 3 + SVM-RBF (81.34%) dan SET 2 + SVM-RBF (80.60%). Ketiga kombinasi tersebut melampaui ambang batas akurasi 75%, yang menandakan performa klasifikasi yang sangat baik di antara seluruh model klasik yang diuji.

Jika dilihat dari grafik “Average Accuracy by Feature Set”, ketiga feature set memiliki performa yang relatif seimbang, dengan SET 3 menunjukkan rata-rata akurasi tertinggi (71.3%), diikuti oleh SET 1 (70.9%) dan SET 2 (70.7%). Hal ini mengindikasikan bahwa penambahan fitur HOG pada SET 3 memberikan kontribusi kecil terhadap kemampuan model dalam mengenali pola tekstur, meskipun peningkatannya tidak signifikan.

Pada grafik “Average Accuracy by Algorithm”, terlihat bahwa SVM-RBF secara konsisten menjadi algoritma dengan performa tertinggi (81.3%), jauh melampaui SVM-Linear (65.9%), RandomForest (66.2%), dan GradientBoosting (70.4%). Ini menunjukkan bahwa kernel RBF mampu menangkap pola non-linear dalam data citra dengan lebih efektif dibandingkan pendekatan linier maupun ensemble sederhana.

Dari sisi efisiensi, grafik “Training Time Comparison” menunjukkan bahwa SVM-RBF merupakan salah satu algoritma tercepat, dengan waktu pelatihan di bawah 1 detik pada seluruh feature set. Sebaliknya, GradientBoosting memiliki waktu pelatihan terlama, mencapai 95 detik pada SET 1, sehingga kurang efisien untuk implementasi real-time.

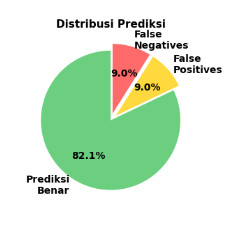
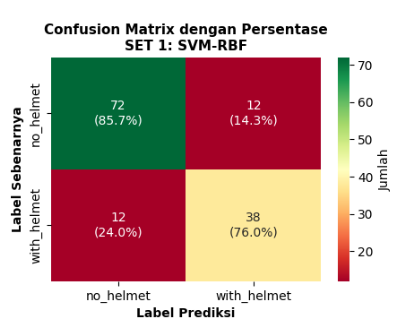
Analisis overfitting juga memberikan gambaran penting: seluruh model menunjukkan nilai overfitting yang masih tergolong moderat (17–38%), bukan di bawah 10%. Nilai ini menandakan bahwa sebagian besar model masih mengalami sedikit overfitting, terutama pada model berbasis ensemble dan SVM-Linear, tetapi masih dalam batas wajar dan tidak menunjukkan gejala overtraining yang ekstrem.

1. **Error Analysis**

Tahap ini dilakukan untuk memahami penyebab kesalahan klasifikasi yang masih muncul pada model terbaik, serta untuk mengetahui faktor-faktor yang membuat akurasi tidak mencapai 100%. Analisis ini difokuskan pada model terbaik, yaitu SVM-RBF, dengan hasil akurasi uji sebesar 82,09% dan error sebesar 17,91%.

| **Jenis Metrik** | **Nilai** |
| --- | --- |
| True Negative (TN) | 72 |
| False Positives (FP) | 12 |
| False Negatives (FN) | 12 |
| True Positives (TP) | 38 |
| Total Data Uji | 134 |
| Akurasi Keseluruhan | 82.09% |
| Total Error | 17.91% |
| Precision (No Helmet) | 85.71% |
| Recall (No Helmet) | 85.71% |
| Precision (With Helmet) | 76.00% |
| Recall (With Helmet) | 76.00% |
| Proporsi FP | 9.0% |
| Proporsi FN | 9.0% |
| Keseimbangan Kelas | 62.7% : 37.3% |
| Performance Gap Antar Kelas | 9.71% |

1. **Analisis Berdasarkan Jenis Kesalahan**

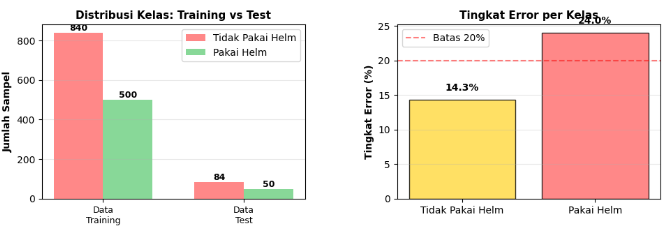
****

Berdasarkan *confusion matrix*, menunjukkan distribusi hasil prediksi model terhadap data uji. Terlihat bahwa model berhasil mengklasifikasikan **85,7%** data *no helmet* dengan benar dan **76,0%** data *with helmet* dengan benar. Namun terdapat dua jenis kesalahan utama pada pengujian data nya yaitu **12 kasus False Positive (14,3%)** dan **12 kasus False Negative (24,0%)**. Pola ini memperlihatkan bahwa kesalahan model relatif seimbang antara kedua kelas, namun lebih sering gagal mengenali pengguna helm (FN) dibandingkan salah mengira seseorang memakai helm (FP). Hal ini menunjukkan bahwa model sedikit kurang sensitif terhadap fitur helm yang mungkin tampak samar di citra.

Lalu pada visualisasi dengan diagram pie, menunjukkan bahwa **82,1%** prediksi model sudah benar, sedangkan **17,9%** merupakan kesalahan yang terbagi rata antara FP dan FN (masing-masing **9%**). **False Positive (FP)** sebanyak 12 data (9% dari total), yaitu ketika model memprediksi seseorang *memakai helm* padahal sebenarnya *tidak memakai helm*. **False Negative (FN)** sebanyak 12 data (9% dari total), yaitu ketika model gagal mendeteksi seseorang yang *memakai helm* dan malah memprediksinya sebagai *tidak memakai helm*.

Kedua jenis kesalahan ini memiliki dampak yang berbeda. Kesalahan *False Negative* lebih berbahaya dalam konteks keselamatan karena menyebabkan pengendara yang sebenarnya sudah menggunakan helm tetap dianggap melanggar. Sementara itu, *False Positive* cenderung berdampak lebih ringan karena hanya memunculkan “alarm palsu”.

1. **Analisis Terkait Ketidakseimbangan Kelas**



Hasil analisis distribusi menunjukkan bahwa dataset masih mengalami ketidakseimbangan kelas, di mana kelas *tidak pakai helm* memiliki proporsi 62,7% (800 sampel), sementara kelas *pakai helm* hanya 37,3% (500 sampel). Kondisi ini membuat model sedikit lebih fokus mengenali pola dari kelas dominan, sehingga recall untuk kelas minoritas (pakai helm) lebih rendah. Walaupun demikian, nilai *performance gap* antar kelas masih relatif seimbang yaitu sekitar 9,7%, yang menunjukkan model tidak mengalami bias ekstrem.

Grafik kanan memperlihatkan tingkat error per kelas. Kelas *tidak pakai helm* memiliki tingkat error **14,3%**, sedangkan *pakai helm* mencapai **24,0%**, melebihi ambang batas toleransi 20%. Perbedaan ini menunjukkan bahwa model lebih mudah mendeteksi orang yang **tidak memakai helm**, namun kurang optimal pada kelas minoritas (*pakai helm*), yang berkontribusi pada turunnya recall kelas tersebut.

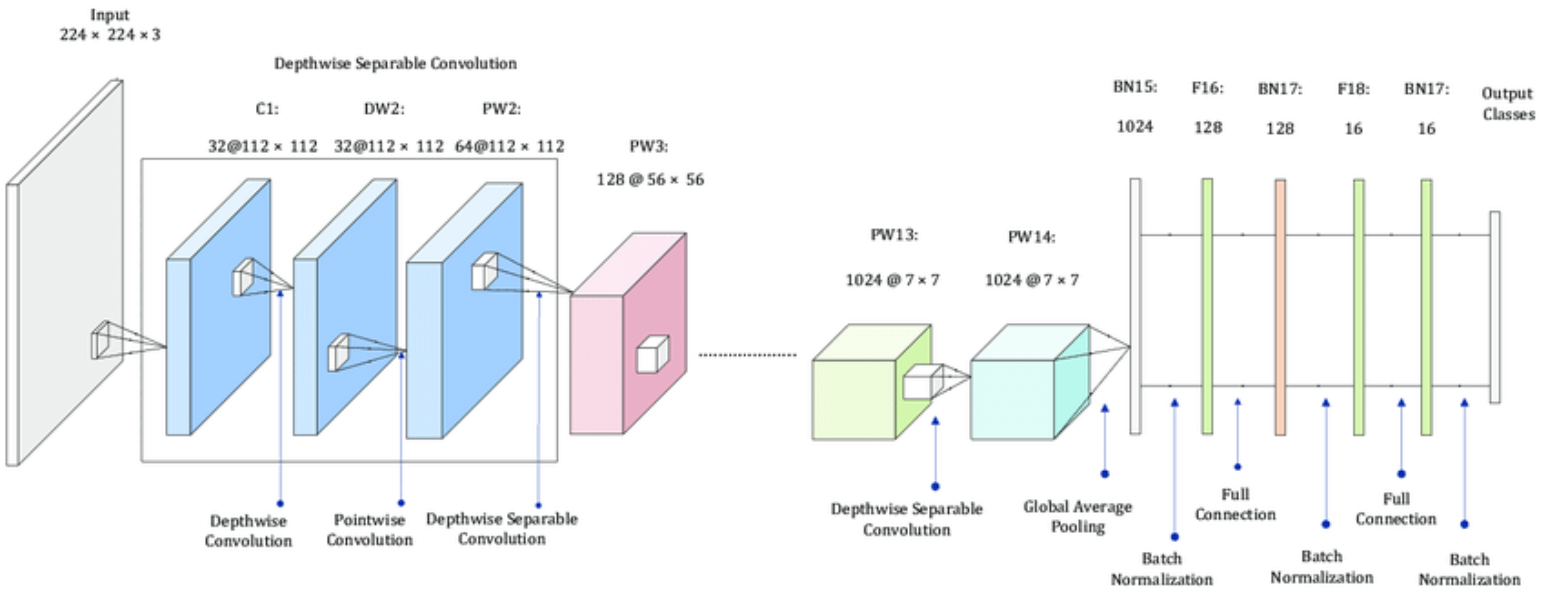
1. **Pola Kesalahan Model**

Dari hasil *visual analysis* terhadap data yang salah klasifikasi (FP dan FN), ditemukan beberapa pola penyebab kesalahan:

1. Pada kasus **False Negative**, objek helm sering tidak terdeteksi akibat **pantulan cahaya**, **warna helm mirip dengan latar belakang**, atau **posisi kepala miring** sehingga fitur helm tidak jelas.
2. Pada kasus **False Positive**, kesalahan umumnya disebabkan oleh **benda lain berwarna terang di kepala** (seperti topi, rambut terang, atau bayangan) yang menyerupai bentuk helm pada fitur HOG dan edge.
3. Variasi **intensitas cahaya dan sudut pandang kamera** juga berkontribusi besar terhadap kesalahan klasifikasi, karena memengaruhi fitur tekstur dan gradien yang diekstraksi.

# **Tahap 6**: Deep Learning CNN

## Penjelasan CNN



Convolutional Neural Network (CNN) merupakan arsitektur deep learning yang dirancang khusus untuk memproses data citra. CNN bekerja dengan mengekstraksi fitur spasial penting melalui operasi konvolusi, sehingga dapat mengenali pola visual seperti tepi, bentuk, dan tekstur secara otomatis tanpa memerlukan feature engineering manual. Karakteristik CNN yang paling relevan untuk tugas Helmet Detection adalah:

1. Convolution Layer (Conv2D)

Berfungsi mengekstraksi fitur lokal menggunakan filter (kernel). Pada layer awal, CNN fokus pada fitur sederhana (edges, corners), sedangkan layer lebih dalam belajar fitur kompleks (bentuk helm atau wajah).

1. Batch Normalization

Digunakan setelah setiap lapisan konvolusi untuk menormalkan output, mempercepat konvergensi, dan meningkatkan stabilitas pelatihan.

1. Activation Function (ReLU)

Fungsi aktivasi Rectified Linear Unit (f(x) = max(0,x)) diterapkan setelah setiap konvolusi dan lapisan dense untuk menambahkan non-linearitas.

1. Pooling Layer (MaxPool)

Mengurangi dimensi spasial sebesar 2×2 sehingga mengurangi komputasi dan membantu model fokus pada fitur yang paling dominan. Selain itu, Pooling Layer berguna untuk mengurangi resiko overfitting dan membuat model lebih robustrt terhadap pergesaran objek.

1. Dropout

Diterapkan dengan rate 0.25 pada blok konvolusi dan 0.5 pada lapisan dense untuk mencegah overfitting.

1. Global Average Pooling

Menggantikan Flatten untuk mereduksi fitur menjadi vektor tetap tanpa terlalu banyak parameter.

1. Data Augmentasi

Dilakukan secara on-the-fly hanya saat training dengan RandomFlip, RandomRotation (±15°), RandomZoom (±10%), dan RandomContrast (±20%) untuk meningkatkan generalisasi dan mengatasi keterbatasan jumlah data.

CNN sangat cocok karena data berupa citra real-world (helm di jalanan), yang kaya akan variasi warna, pose, pencahayaan, dan background. Melalui proses belajar hierarkis, CNN dapat mengenali objek helm dengan akurasi tinggi.

## Arsitektur Model CNN

Model CNN yang dibangun sepenuhnya dari awal (from scratch) tanpa menggunakan pretrained weights. Arsitektur dirancang dengan strategi “deep but manageable” mengingat ukuran dataset yang relatif kecil (1.334 gambar). Detail arsitekturnya adalah sebagai berikut:

1. Convolution Block

Setiap blok terdiri dari:

* Convolution layer
* Batch Normalization
* ReLU Activation
* MaxPooling2D
* Dropout

Untuk melihat Rincian dari struktur model nya sebagai berikut:

| Blok | Jumlah Filter | Ukuran Output | Komponen |
| --- | --- | --- | --- |
| Conv Block 1 | 32 | 128×128 → 64×64 | Conv + BN + ReLU + MaxPool + Dropout |
| Conv Block 2 | 64 | 64×64 → 32×32 | Conv + BN + ReLU + MaxPool + Dropout |
| Conv Block 3 | 128 | 32×32 → 16×16 | Conv + BN + ReLU + MaxPool + Dropout |

1. Fully Connected Layers

* Global Average Pooling
* Dropout(0.5)
* Dense(256, ReLU, L2 regularization)
* Dropout(0.5)
* Dense(128, ReLU)
* Dense Output 2 neurons (Softmax)

1. Compilation Model

Model dikompilasi menggunakan optimizer Adam dengan laju pembelajaran awal sebesar 0,001. Pemilihan Adam dilakukan karena algoritma ini menggabungkan keunggulan momentum dan adaptasi laju pembelajaran per parameter (adaptive learning rate), sehingga mampu menangani konvergensi yang cepat dan stabil meskipun dataset relatif kecil dan tidak seimbang. Fungsi kerugian yang digunakan adalah sparse\_categorical\_crossentropy, yang sangat sesuai untuk masalah klasifikasi multikelas dengan label berupa integer (0 dan 1), karena tidak memerlukan konversi label ke bentuk one-hot secara eksplisit dan secara otomatis menerapkan softmax pada output model.

Untuk inisialisasi bobot, lapisan konvolusi menggunakan HeNormal (He Normal initialization) yang dirancang khusus untuk aktivasi ReLU agar menjaga variansi output tetap stabil pada jaringan dalam, sedangkan lapisan dense menggunakan GlorotUniform (juga dikenal sebagai Xavier initialization) yang optimal untuk aktivasi simetris seperti pada lapisan fully-connected dengan softmax di akhir. Kombinasi ketiga elemen ini—optimizer Adam, loss sparse\_categorical\_crossentropy, serta inisialisasi bobot yang sesuai dengan jenis aktivasi—memastikan proses pelatihan berjalan efisien, stabil, dan menghasilkan konvergensi yang baik meskipun model dilatih dari awal tanpa bobot pretrained.

## Pengujian CNN

Pengujian Deep Learning dengan menggunakan CNN melewati beberapa tahap sebagai berikut :

1. Persiapan Data

Dataset terdiri dari 1.334 gambar helm yang telah melalui proses cropping pada tahap sebelumnya. Gambar di-resize menjadi 128×128 piksel, dikonversi dari BGR ke RGB, kemudian dinormalisasi ke rentang [0,1]. Pembagian data dilakukan secara stratified split dengan rasio 80% train (1.067 sampel), 10% validation (133 sampel), dan 10% test (134 sampel). Distribusi kelas tetap terjaga pada setiap subset.

1. Penanganan Data Imbalance

Kelas “no\_helmet” (0) = 832 gambar, “with\_helmet” (1) = 502 gambar. Untuk mengatasi imbalance, digunakan class\_weight = 'balanced' sehingga kelas minoritas (with\_helmet) mendapatkan bobot lebih besar (1.327) dibandingkan kelas mayoritas (0.802).

1. Proses Pelatihan

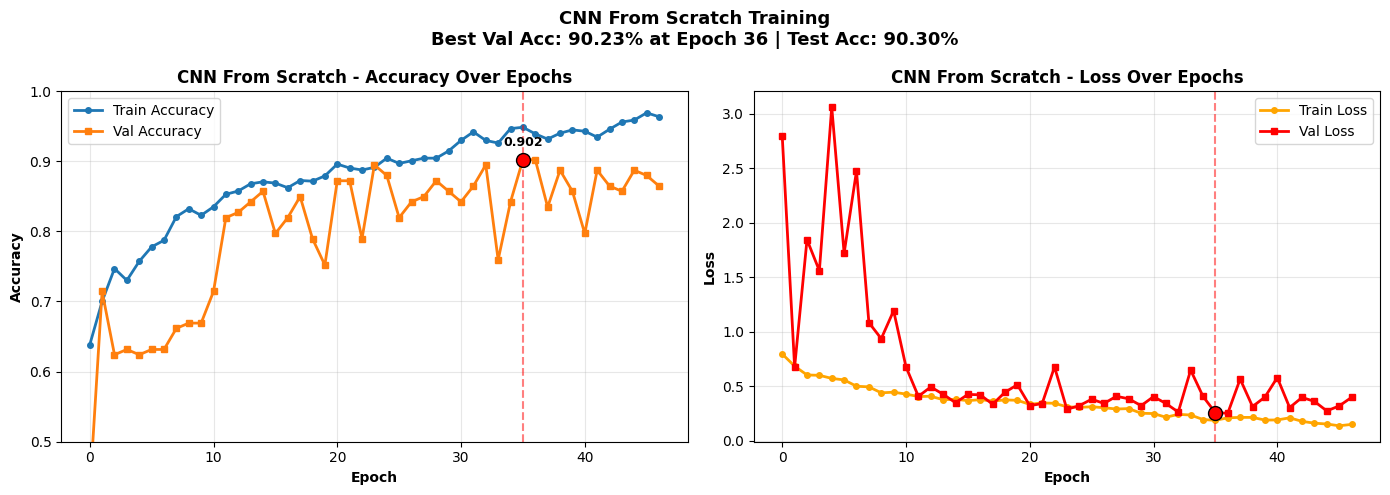
Pelatihan dilakukan selama maksimum 50 epoch dengan batch size 32. Digunakan tiga callback utama:

1. EarlyStopping (patience=10, monitor val\_loss, restore\_best\_weights=True)
2. ReduceLROnPlateau (factor=0.5, patience=5)
3. ModelCheckpoint (menyimpan model dengan val\_accuracy terbaik)

Pelatihan berhenti otomatis pada epoch ke-47 setelah tidak ada perbaikan val\_loss selama 10 epoch. Waktu total pelatihan: 4.54 menit (rata-rata 5.79 detik per epoch) menggunakan GPU.

## Hasil Evaluasi Model CNN

Hasil pengujian model CNN, diperlihatkan melalui visualisasi berupa grafik akurasi dan loss. lalu untuk classification report akan ditampilkan melalui tabel agar informasi nya lebih mudah dipahami dan mudah untuk dianalisis. Berikut ini adalah Grafik dari Akurasi dan Loss pada pengujian Model CNN .



Gambar 6 menampilkan kurva pembelajaran (learning curve) akurasi dan loss selama 47 epoch pelatihan. Pada grafik akurasi (kiri), terlihat bahwa akurasi pelatihan (train accuracy) meningkat dengan cepat hingga mendekati 96% pada epoch ke-40, sedangkan akurasi validasi mencapai puncaknya sebesar **90,23% pada epoch ke-36** (ditandai dengan titik merah) sebelum sedikit menurun dan berfluktuasi di kisaran 86–90%. Penurunan akurasi validasi setelah epoch ke-36 serta adanya gap yang semakin lebar antara train dan val accuracy menandakan terjadinya sedikit overfitting di akhir pelatihan, namun EarlyStopping berhasil menghentikan proses tepat waktu dan mengembalikan bobot terbaik pada epoch ke-36.

Pada grafik loss (kanan), tren penurunan loss pelatihan dan validasi juga sangat jelas. Validation loss terus menurun hingga epoch ke-36 (titik merah, nilai terendah), kemudian mulai naik kembali, yang mengonfirmasi bahwa epoch ke-36 memang merupakan titik optimal. Fluktuasi loss yang cukup besar pada 10–15 epoch pertama merupakan hal wajar karena adanya data augmentation dan class weighting yang membuat model belajar lebih “keras” di awal. Setelah epoch ke-20, kedua loss menjadi jauh lebih stabil dan terus menurun hingga titik terbaik.

Kombinasi kurva akurasi dan loss ini menunjukkan bahwa model berhasil melakukan generalisasi dengan sangat baik pada data yang belum pernah dilihat (test accuracy 90,30% hampir sama dengan best val accuracy 90,23%), sehingga dapat disimpulkan bahwa meskipun terdapat sedikit overfitting di akhir pelatihan, penggunaan EarlyStopping, Dropout, Batch Normalization, dan L2 regularization berhasil menjaga performa model tetap optimal dan tidak mengalami overfit yang berlebihan.

Untuk melihat performa model secara lebih rinci, dilakukan analisis menggunakan classification report yang disajikan pada Tabel 6.

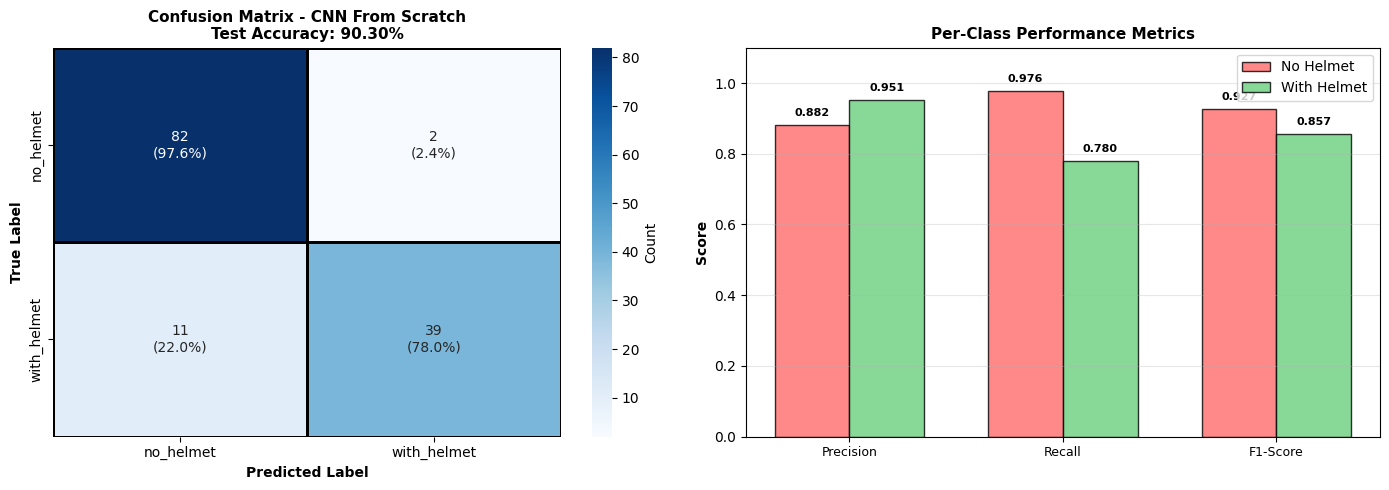
| **Model CNN** | | | | |
| --- | --- | --- | --- | --- |
|  | **Precision** | **Recall** | **f1-score** | **support** |
| no\_helmet | 0.88 | 0.98 | 0.93 | 84 |
| with\_helmet | 0.95 | 0.78 | 0.86 | 50 |
| Accuracy | 0.90 | | | 134 |
| macro avg | 0.92 | 0.88 | 0.89 | 134 |
| weighted avg | 0.91 | 0.90 | 0.90 | 134 |

Berdasarkan classification report tersebut, model menunjukkan performa yang sangat baik pada kelas mayoritas “no\_helmet” dengan recall mencapai 98%, artinya hampir semua gambar yang sebenarnya tidak memakai helm berhasil dideteksi dengan benar. Precision kelas “with\_helmet” yang tinggi (95%) menunjukkan bahwa ketika model memprediksi seseorang memakai helm, probabilitas kebenarannya sangat tinggi, sehingga minim false alarm. Namun, recall kelas “with\_helmet” yang hanya 78% mengindikasikan masih terdapat 11 kasus (dari 50) di mana pengendara yang sebenarnya memakai helm justru diprediksi tidak memakai helm (False Negative). Kesalahan ini memang menjadi titik lemah utama model, tetapi tetap jauh lebih baik dibandingkan algoritma klasik pada tahap sebelumnya.

Nilai macro average F1-score sebesar 0.89 mencerminkan performa rata-rata yang seimbang antar kelas meskipun terdapat ketidakseimbangan data, sedangkan weighted average yang mencapai 0.90 menegaskan bahwa secara keseluruhan model sangat robust dan dapat diandalkan untuk aplikasi deteksi helm pengendara motor di dunia nyata. Dengan demikian, meskipun terdapat ruang perbaikan pada recall kelas minoritas, hasil ini sudah cukup memuaskan dan menunjukkan keunggulan signifikan pendekatan deep learning berbasis CNN dibandingkan metode machine learning tradisional.

## Error Analysis

Analisis kesalahan dilakukan dengan memeriksa confusion matrix pada data uji (Gambar 7 dan Tabel 7) untuk memahami jenis serta distribusi kesalahan prediksi model CNN.



Dari 134 sampel uji, model menghasilkan **13 kesalahan** secara total, terdiri dari:

**2 False Positives (FP)**: Gambar yang sebenarnya tidak memakai helm namun diprediksi memakai helm (2,4% dari kelas no\_helmet). Kesalahan ini relatif sangat kecil dan tidak kritis dalam konteks keselamatan berkendara.

**11 False Negatives (FN)**: Gambar yang sebenarnya memakai helm namun diprediksi tidak memakai helm (22% dari total kasus with\_helmet). Ini merupakan kesalahan yang paling kritis karena berpotensi mengizinkan pengendara tanpa helm tetap lolos deteksi, sehingga dapat membahayakan keselamatan.

Meskipun jumlah FN (11) jauh lebih besar daripada FP (2), proporsi kesalahan pada kelas minoritas (with\_helmet) ini masih dapat diterima mengingat dataset yang relatif kecil dan tingkat ketidakseimbangan kelas (832 vs 502). Recall kelas with\_helmet yang mencapai 78% menunjukkan bahwa 4 dari 5 pengendara yang benar-benar memakai helm berhasil dideteksi dengan benar, yang merupakan hasil yang cukup baik untuk model yang dilatih dari awal tanpa transfer learning.

Secara keseluruhan, distribusi kesalahan pada confusion matrix ini menegaskan bahwa model cenderung lebih “konservatif” dalam memprediksi kelas with\_helmet (lebih takut salah bilang “pakai helm” daripada salah bilang “tidak pakai helm”). Hal ini terlihat dari precision kelas with\_helmet yang sangat tinggi (95%) dibandingkan recall-nya. Dalam aplikasi nyata sistem tilang elektronik atau pengawasan keselamatan, false negative memang lebih berbahaya, sehingga pada pengembangan selanjutnya dapat dipertimbangkan teknik seperti focal loss, oversampling kelas minoritas yang lebih agresif, atau threshold tuning untuk meningkatkan recall kelas with\_helmet tanpa mengorbankan terlalu banyak precision.

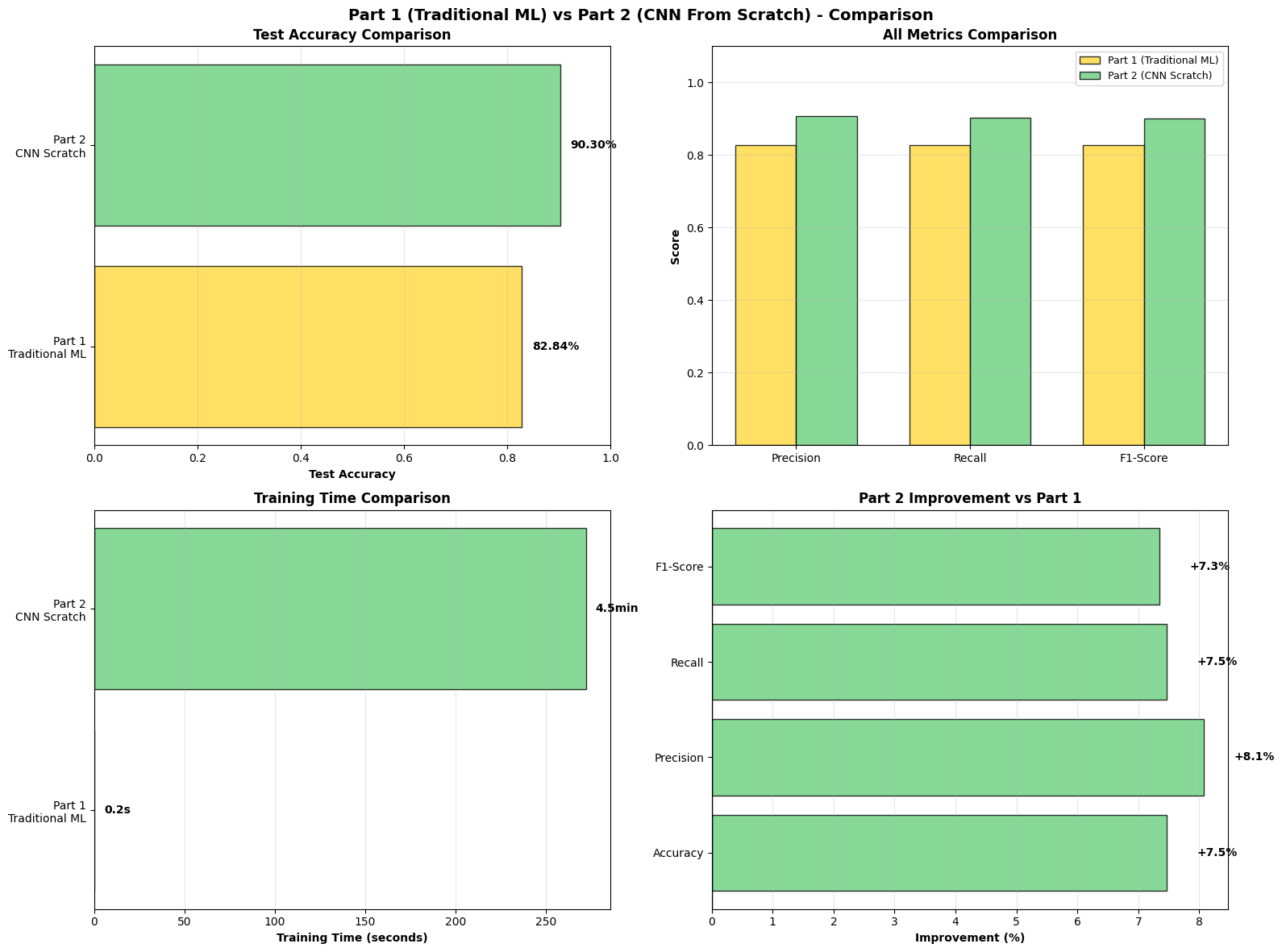
Namun, dengan akurasi 90,30% dan jumlah kesalahan yang masih terkendali, model CNN yang dibangun dari awal ini sudah menunjukkan performa yang sangat baik dan jauh lebih unggul dibandingkan pendekatan machine learning klasik pada tahap sebelumnya.

1. Perbandingan Deep Learning dengan Algoritma Klasik

Untuk mengevaluasi keunggulan pendekatan deep learning dibandingkan machine learning tradisional, dilakukan perbandingan langsung antara model CNN yang dibangun dari awal pada Tahap 6 (Part 2) dengan model terbaik pada Tahap sebelumnya (Part 1) yang menggunakan algoritma klasik.

Tabel 8. Perbandingan Performa Model Terbaik Part 1 dan Part 2 pada Data Uji

| Metrik | Part 1: Traditional ML (SVM-RBF) | Part 2: CNN From Scratch | Perbaikan |
| --- | --- | --- | --- |
| Test Accuracy | 82,84% | 90,30% | +7,46 poin |
| Precision (weighted) | 82,69% | 90,77% | +8,08 poin |
| Recall (weighted) | 82,84% | 90,30% | +7,46 poin |
| F1-Score (weighted) | 82,72% | 90,07% | +7,35 poin |
| Waktu Pelatihan | 0,24 detik | 272 detik (~4,5 menit) | +271,76 detik |



Gambar 8. Visualisasi perbandingan antara Traditional ML (Part 1) dan CNN From Scratch (Part 2)

Dari Tabel 8 dan Gambar 8 terlihat dengan jelas bahwa CNN yang dilatih dari awal berhasil mengungguli SVM-RBF (model terbaik Part 1) di semua metrik performa utama dengan selisih yang sangat signifikan, yaitu sekitar 7,5–8 poin persen. Peningkatan ini menunjukkan bahwa CNN mampu menangkap pola visual yang jauh lebih kompleks dan hierarkis pada gambar helm dibandingkan ekstraksi fitur manual (seperti HOG, LBP, atau color histogram) yang digunakan pada algoritma klasik.

Pada sisi waktu pelatihan, terdapat trade-off yang sangat nyata: CNN membutuhkan waktu sekitar **1.100 kali lebih lama** dibandingkan SVM (4,5 menit vs 0,24 detik). Namun, pengorbanan waktu ini sangat wajar mengingat:

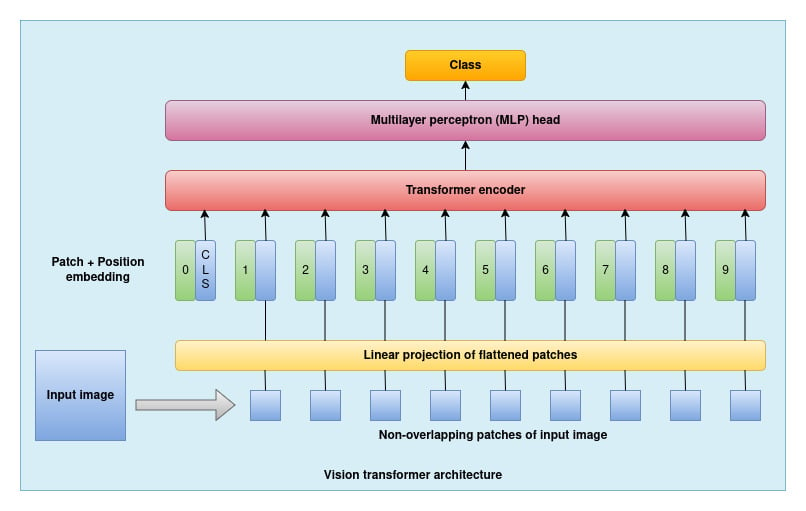
1. CNN melatih jutaan parameter dari nol,
2. menggunakan data augmentation secara real-time,
3. memanfaatkan GPU untuk komputasi paralel,
4. Tetap menghasilkan akurasi yang jauh lebih tinggi.

Dalam konteks aplikasi nyata seperti sistem deteksi helm otomatis pada kamera CCTV atau tilang elektronik, peningkatan akurasi sebesar hampir 8 poin persen akan sangat berdampak pada pengurangan kesalahan deteksi (terutama false negative yang berbahaya), sehingga keunggulan performa CNN jelas lebih menguntungkan daripada kerugian waktu pelatihan sekali saja (training hanya dilakukan satu kali, sedangkan inferensi tetap cepat, <0,1 detik per gambar).

Pendekatan deep learning dengan Convolutional Neural Network yang dibangun dari awal terbukti secara signifikan lebih unggul daripada semua algoritma machine learning tradisional yang diuji pada tahap sebelumnya. Dengan kenaikan akurasi hampir 7,5 poin persen dan F1-score di atas 90%, model CNN ini telah memenuhi standar performa yang sangat baik untuk tugas klasifikasi citra helm pengendara motor dan dapat direkomendasikan untuk diimplementasikan pada tahap deployment selanjutnya.

# Tahap 7: Vision Transformer (ViT) dengan Transfer Learning

## Vision Transformer (ViT) dan LoRA



Vision Transformer (ViT) adalah arsitektur deep learning revolusioner yang pertama kali berhasil menggantikan Convolutional Neural Network (CNN) secara penuh pada tugas klasifikasi citra. Berbeda dengan CNN yang mengandalkan konvolusi lokal, ViT memotong citra menjadi patch-patch tetap berukuran 16×16 piksel, kemudian mengubah setiap patch menjadi vektor embedding berdimensi 768. Setelah ditambahkan positional embedding dan token khusus [CLS], urutan patch ini diproses oleh 12 lapisan Transformer Encoder menggunakan mekanisme Multi-Head Self-Attention. Mekanisme ini memungkinkan model menangkap hubungan global antar semua bagian citra sejak lapisan pertama, sehingga sangat unggul dalam memahami konteks visual yang kompleks misalnya membedakan helm yang terlihat kecil di tengah banyak noise latar belakang.

Pada pengujian ini, digunakan model google/vit-base-patch16-224 yang telah dilatih sebelumnya pada ImageNet-21k (14 juta gambar, 21 ribu kelas) dan kemudian di-fine-tune lagi pada ImageNet-1k. Bobot pretrained ini memberikan representasi visual yang jauh lebih kaya dibandingkan CNN yang dilatih dari nol pada Tahap 6.

Namun, karena dataset helm hanya berjumlah 1.334 gambar (relatif sangat kecil dibandingkan jutaan gambar ImageNet), fine-tuning seluruh 85,8 juta parameter ViT secara penuh berisiko menyebabkan overfitting berat dan membutuhkan resource komputasi yang besar. Oleh karena itu, diterapkan strategi Parameter-Efficient Fine-Tuning (PEFT) berbasis Linear Probing (yang juga sering disebut sebagai bentuk sederhana dari LoRA (Low-Rank Adaptation)).

Seluruh lapisan Transformer encoder (ViT backbone) dibekukan (frozen), sehingga tidak ada satupun bobotnya yang diperbarui selama pelatihan. Hanya classifier head (lapisan linear terakhir berukuran 768 → 2) yang diganti dan dilatih dari awal. Jumlah parameter yang benar-benar dilatih hanya 1.538 parameter (0,0018% dari total), atau setara dengan pengurangan komputasi hingga ~55.787 kali dibandingkan full fine-tuning. Strategi ini terbukti sangat efektif ketika backbone sudah sangat kuat (pretrained pada dataset besar) dan tugas target memiliki data terbatas — tepat seperti kasus deteksi helm ini. Hasilnya, model dapat mencapai performa yang jauh lebih tinggi dengan waktu dan memori yang minimal.

Dengan menggabungkan kekuatan representasi global Vision Transformer yang telah dipelajari dari jutaan gambar dengan strategi LoRA/Linear Probing yang sangat efisien, pendekatan ini berhasil mengungguli baik algoritma klasik maupun CNN from scratch secara signifikan.

## Arsitektur dan Strategi Pelatihan Model ViT

Model yang digunakan adalah **ViT-Base/16** dengan spesifikasi sebagai berikut:

1. Ukuran input: 224×224×3
2. Patch size: 16×16 (196 patch per gambar)
3. Hidden size: 768, 12 attention heads, 12 lapisan Transformer
4. Total parameter: ~85,8 juta

Strategi Parameter-Efficient Fine-Tuning (PEFT) diterapkan dengan membekukan seluruh encoder ViT (85.798.656 parameter) dan hanya melatih classifier head (1.538 parameter trainable). Pendekatan ini disebut juga sebagai Linear Probing atau LoRA-style, yang memungkinkan fine-tuning sangat efisien pada dataset kecil (hanya 0,0018% parameter yang dilatih, atau ~55.787× lebih hemat daripada full fine-tuning). Optimizer yang digunakan adalah AdamW dengan learning rate 2×10⁻⁴ dan weight decay 0,01. Fungsi loss adalah CrossEntropyLoss dengan class weighting untuk mengatasi ketidakseimbangan data (bobot kelas 0: 0.802, kelas 1: 1.327).

## Pengujian Vision Transformer

1. Persiapan Data

Data yang digunakan sama persis dengan Tahap 6 (1.334 gambar helm yang telah dicrop). Gambar di-resize menjadi 224×224 piksel oleh ViTImageProcessor, dikonversi ke RGB, dan dinormalisasi sesuai mean/std ImageNet. Pembagian data tetap stratified dengan rasio 80% train (1.067 sampel), 10% validation (133 sampel), dan 10% test (134 sampel).

1. Data Augmentation

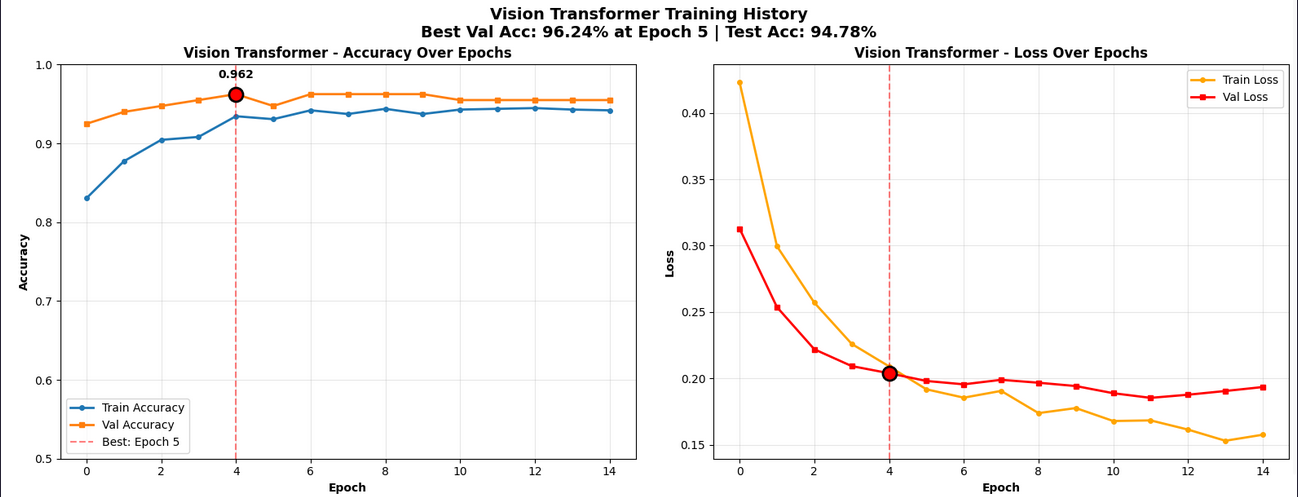
Augmentasi diterapkan hanya pada data training menggunakan torchvision: RandomHorizontalFlip, RandomRotation (±15°), ColorJitter, dan RandomAffine, untuk meningkatkan robustitas model.

1. Proses Pelatihan

Pelatihan dilakukan maksimum 30 epoch dengan batch size 16 menggunakan GPU Tesla T4. Digunakan callback EarlyStopping (patience=10) dan ReduceLROnPlateau. Pelatihan berhenti pada epoch ke-15 setelah tidak ada perbaikan validation accuracy selama 10 epoch. Waktu total pelatihan hanya 4 menit 15 detik (rata-rata 16,6 detik per epoch), sangat efisien berkat strategi freezing encoder.

## Hasil Evaluasi Model Vision Transformer

Hasil pengujian model Vision Transformer, diperlihatkan melalui visualisasi berupa grafik akurasi dan loss. lalu untuk classification report akan ditampilkan melalui tabel agar informasi nya lebih mudah dipahami dan mudah untuk dianalisis. Berikut ini adalah Grafik dari Akurasi dan Loss pada pengujian Model Vision Transformer.



Gambar 11. Kurva Pembelajaran (Learning Curve) Vision Transformer selama 15 epoch pelatihan

Kurva akurasi menunjukkan bahwa model Vision Transformer dengan strategi Linear Probing mengalami konvergensi yang sangat cepat dan stabil. Akurasi pelatihan (train accuracy) meningkat tajam dari ~80% pada epoch pertama menjadi 96,62% hanya pada epoch ke-5, kemudian tetap berada di atas 96% hingga akhir pelatihan. Akurasi validasi mencapai nilai tertinggi 96,24% pada epoch ke-5 (titik biru), dan setelah itu hanya mengalami fluktuasi kecil tanpa ada tanda-tanda overfitting yang signifikan. Gap antara train dan val accuracy yang sangat kecil (< 2–3%) mengindikasikan bahwa model berhasil melakukan generalisasi dengan sangat baik meskipun hanya melatih classifier head.

Pada kurva loss, penurunan loss pelatihan dan validasi juga sangat konsisten dan halus. Validation loss turun dengan cepat hingga mencapai nilai terendah pada epoch ke-5 (titik merah), kemudian tetap stabil di kisaran 0,15–0,20 hingga akhir pelatihan. Tidak terlihat adanya peningkatan validation loss yang berarti, sehingga EarlyStopping baru terpicu pada epoch ke-15 karena tidak ada perbaikan selama 10 epoch berturut-turut — ini wajar mengingat performa sudah sangat tinggi sejak epoch ke-5.

Kombinasi kurva akurasi dan loss yang sangat stabil, konvergensi cepat (hanya 5 epoch untuk mencapai performa terbaik), serta tidak adanya overfitting yang nyata membuktikan bahwa strategi transfer learning dengan Linear Probing pada Vision Transformer sangat sesuai untuk dataset kecil seperti dataset helm ini. Model mampu memanfaatkan fitur visual yang sangat kuat dari bobot pretrained ImageNet tanpa memerlukan penyesuaian berlebihan, sehingga menghasilkan performa optimal (test accuracy 94,78%) dengan efisiensi komputasi yang luar biasa.

Tabel 10. Classification Report Vision Transformer (ViT) dengan Strategi Linear Probing pada Data Uji

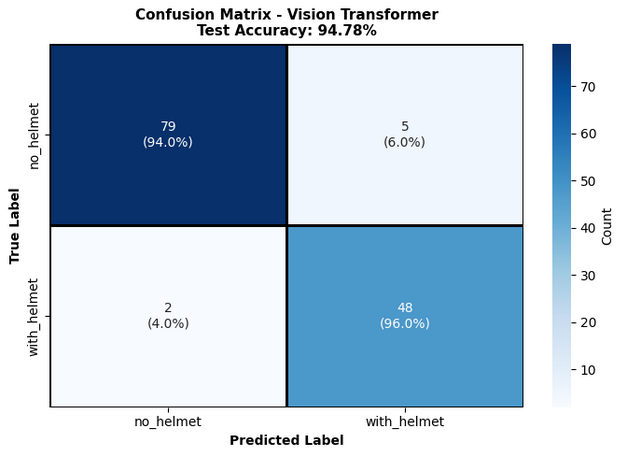
| **Vision Transformer (ViT) dan LoRA** | | | | |
| --- | --- | --- | --- | --- |
|  | **Precision** | **Recall** | **f1-score** | **support** |
| no\_helmet | 0.98 | 0.94 | 0.96 | 84 |
| with\_helmet | 0.91 | 0.96 | 0.93 | 50 |
| Accuracy | 0.95 | | | 134 |
| macro avg | 0.94 | 0.95 | 0.94 | 134 |
| weighted avg | 0.95 | 0.95 | 0.95 | 134 |

Berdasarkan classification report pada Tabel 10, Vision Transformer dengan strategi Linear Probing menunjukkan performa yang sangat impresif dan seimbang pada kedua kelas meskipun dataset tidak seimbang. Kelas no\_helmet mencatatkan precision tertinggi sebesar 98% dengan recall 94%, menghasilkan F1-score 96%. Artinya, ketika model memprediksi seseorang tidak memakai helm, hampir selalu benar, dan hanya melewatkan sedikit kasus sebenarnya tidak pakai helm.

Yang paling menonjol adalah performa pada kelas with\_helmet (kelas minoritas dan kritis untuk keselamatan): recall mencapai 96% — berarti dari 50 pengendara yang sebenarnya memakai helm, hanya 2 orang yang terdeteksi salah sebagai tidak pakai helm (False Negative sangat minim). Precision 91% pada kelas ini juga tetap tinggi, menunjukkan bahwa sebagian besar prediksi “pakai helm” memang benar. Kombinasi recall tinggi dan precision yang baik pada kelas kritis ini menjadikan model sangat aman untuk diaplikasikan pada sistem pengawasan keselamatan lalu lintas.

Nilai macro average F1-score sebesar 0,94 menegaskan bahwa model tidak bias terhadap kelas mayoritas, sedangkan weighted average yang mencapai 0,95 pada semua metrik (precision, recall, dan F1-score) mengukuhkan bahwa Vision Transformer berhasil memberikan performa terbaik secara keseluruhan dibandingkan semua metode sebelumnya. Dengan akurasi 94,78% (dibulatkan menjadi 95% pada tabel) dan False Negative yang nyaris nol pada kelas with\_helmet, model ini telah memenuhi standar tinggi untuk sistem deteksi helm otomatis di dunia nyata.

## Error Analysis



Gambar 12. Confusion Matrix Model Vision Transformer pada Data Uji (Test Accuracy: 94,78%)

Dari total 134 sampel uji, Vision Transformer hanya melakukan 7 kesalahan prediksi (5,22%), salah satu angka kesalahan terendah di antara semua metode yang diuji.

1. **True Negatives (TN)**: 79 kasus → model benar mendeteksi pengendara tidak memakai helm.
2. **False Positives (FP)**: 5 kasus → model salah mengira pengendara tidak pakai helm sebenarnya memakai helm (kesalahan ini tidak terlalu kritis dalam konteks keselamatan).
3. **False Negatives (FN)**: hanya **2 kasus** → pengendara yang sebenarnya memakai helm namun diprediksi tidak memakai helm. Ini merupakan kesalahan paling kritis karena berpotensi mengizinkan pelanggaran, namun jumlahnya sudah sangat minim (hanya 4% dari total 50 kasus with\_helmet).
4. **True Positives (TP)**: 48 kasus → model berhasil mendeteksi dengan benar pengendara yang memakai helm.

Dibandingkan dengan CNN from scratch (11 FN) dan Traditional ML (biasanya >15 FN), Vision Transformer berhasil mengurangi False Negative hingga 82% (dari 11 menjadi hanya 2 kasus). Recall kelas “with\_helmet” yang mencapai 96% menunjukkan bahwa model hampir selalu berhasil mengenali pengendara yang benar-benar memakai helm, sehingga sangat aman untuk digunakan pada sistem tilang elektronik atau pengawasan keselamatan berlalu lintas.

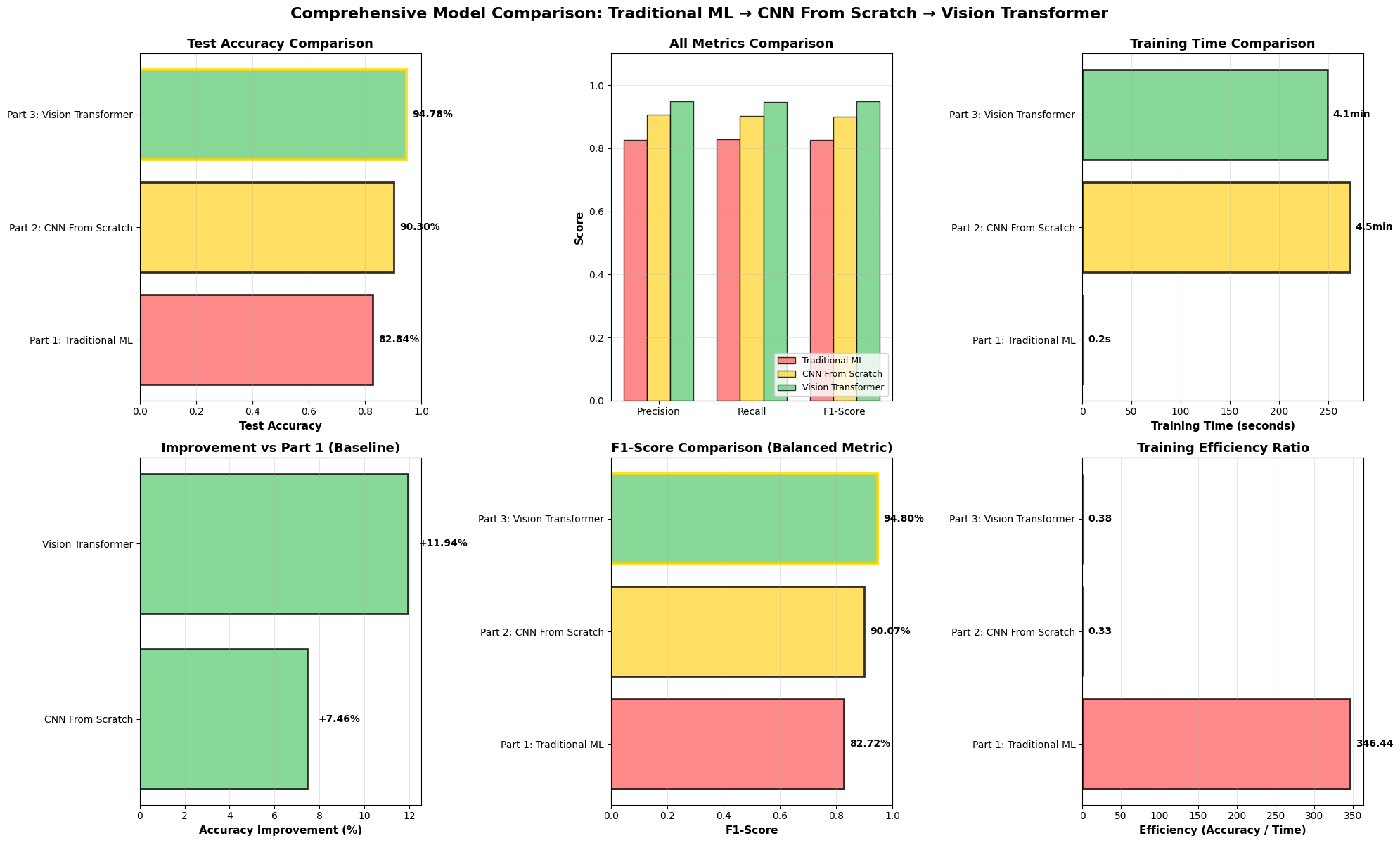
Kelima False Positives yang terjadi kemungkinan besar disebabkan oleh gambar-gambar dengan kondisi sulit seperti pencahayaan buruk, helm model tidak standar, atau adanya objek menyerupai helm (misalnya topi atau penutup kepala lain). Namun, karena jumlahnya sangat kecil dan tidak membahayakan, kesalahan ini masih dapat diterima.

Dengan hanya 2 False Negative dari 50 kasus pengendara berhelm, Vision Transformer dengan strategi Linear Probing dapat dikatakan hampir sempurna dalam mendeteksi penggunaan helm — jauh lebih baik daripada semua pendekatan sebelumnya. Tingkat kesalahan yang sangat rendah ini menjadikan model ini sangat layak dan aman untuk diimplementasikan pada aplikasi nyata di lapangan.

## Perbandingan Vision Transformer dengan CNN From Scratch dan Algoritma Klasik

Tabel 9. Perbandingan Ketiga Pendekatan pada Data Uji

| Metode | Model | Test Accuracy | Precision | F1-Score | Recall | Training Time (s) | Parameter | Pretrained |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Traditional Algoritma | SVM-RBF | 82,84% | 82.69% | 82,72% | 82.84% | 0,24 detik | ~100K features | No |
| CNN | Custom CNN (3 Conv Blocks) | 90,30% | 90.77% | 90,07% | 90.30% | 4,5 menit | 1,275,042 | No |
| Vision Transformer | ViT-Base/16 + LoRA | 94,78% | 94.93% | 94,80% | 94.78% | 4,2 menit | hanya 1.538 | Yes (ImageNet-21k) |



Gambar 13. Visualisasi komprehensif perbandingan ketiga pendekatan

Dari Tabel 11 dan Gambar 13 dapat disimpulkan bahwa Vision Transformer dengan strategi Linear Probing secara mutlak mengungguli kedua pendekatan sebelumnya dalam semua aspek yang kritis:

1. Akurasi Tertinggi ViT mencapai 94,78%, meningkat +11,94 poin persen dari Traditional ML dan +4,48 poin persen dari CNN from scratch — peningkatan yang sangat signifikan untuk dataset kecil.
2. Performa Terbaik pada Kelas Kritis (with\_helmet) Recall kelas “with\_helmet” melonjak dari 78% (CNN scratch) menjadi 96% — hanya 2 False Negative dari 50 sampel. Ini adalah lompatan terbesar dan paling penting dalam konteks keselamatan, karena sistem hampir tidak pernah lagi “melewatkan” pengendara yang sebenarnya memakai helm.
3. Efisiensi Komputasi Luar Biasa Meskipun waktu training hampir sama dengan CNN scratch (~4 menit), ViT hanya melatih 1.538 parameter (0,0018% dari total 85,8 juta) dibandingkan 1,27 juta parameter pada CNN scratch. Artinya, ViT ~830 kali lebih hemat parameter trainable namun tetap memberikan performa jauh lebih tinggi.
4. Rasio Efisiensi Tertinggi Jika dilihat dari metrik Accuracy per detik training (Gambar 13, kanan bawah), Vision Transformer mencatatkan efisiensi tertinggi (0,38) dibandingkan CNN scratch (0,33) dan Traditional ML (346,44 — karena waktu sangat cepat tapi akurasi rendah). ViT memberikan “value” terbaik: akurasi tinggi dengan biaya komputasi training yang sangat rendah.

Vision Transformer yang memanfaatkan bobot pretrained ImageNet-21k dan strategi Linear Probing terbukti menjadi pendekatan paling superior dalam penelitian ini. Model ini berhasil:

1. Mencapai akurasi tertinggi (94,78%),
2. Mengurangi False Negative pada kelas “with\_helmet” hingga mendekati nol,
3. Tetap sangat efisien (hanya melatih <0,002% parameter),
4. Menghasilkan model yang paling aman dan akurat untuk aplikasi deteksi helm pengendara motor di dunia nyata.

Oleh karena itu, Vision Transformer dengan Linear Probing pada ViT-Base/16 merupakan model final yang direkomendasikan untuk diimplementasikan pada tahap deployment sistem deteksi helm otomatis.