BAB I
PENDAHULUAN

1.1 Latar Belakang

Di media sosial saat ini, banyak orang terdorong untuk cepat membagikan peristiwa terbaru dan mengejar perhatian, sehingga konten yang belum disaring termasuk gambar atau video yang mengandung unsur kekerasan bisa ikut terunggah dan tersebar. Di Indonesia, skala penggunaan media sosial yang besar membuat risiko ini semakin nyata, karena pada awal 2025 diperkirakan ada 212 juta pengguna internet dan 143 juta identitas pengguna media sosial (Kemp, 2025). Volume data yang dibuat dan dipublikasikan ke internet per hari pada tahun 2023 melebihi 402 juta TB (Kearns, 2025). Penggunaan platform sosial ini sering kali didorong oleh kebutuhan untuk merasa terhubung dan diterima serta kebutuhan untuk menampilkan diri, yang dalam praktiknya dapat memicu perilaku mencari eksposur (Nadkarni & Hofmann, 2012). Dalam konteks berbagi informasi, tinjauan riset menemukan bahwa aktivitas *news sharing* di media sosial memang umum dan terkait dengan kebutuhan berbagi pembaruan (Kümpel dkk., 2015). Di sisi lain, studi tentang *virality* menunjukkan konten yang membangkitkan emosi berenergi tinggi seperti marah atau cemas cenderung lebih mudah menyebar, sehingga konten kekerasan berpotensi "meledak" jangkauannya ketika memicu reaksi kuat (Buckley dkk., 2012). Penyebaran ini makin dipercepat karena *feed* konten pada media sosial modern banyak dipengaruhi sistem rekomendasi dan perankingan untuk memilih konten yang dianggap paling relevan dan menarik bagi pengguna, bukan murni urutan waktu, sehingga konten yang memicu gejolak emosi dapat muncul termasuk pada orang yang tidak sengaja mencarinya (Covington dkk., 2016). Paparan yang berulang terhadap gambar grafis dari peristiwa yang mengandung kekerasan juga dapat dikaitkan dengan dampak psikologis. Hasil pada suatu riset menunjukkan paparan media grafis terkait peristiwa traumatis dapat menimbulkan peningkatan gejala *stress* pascatrauma, dan kajian terbaru menegaskan bahwa gambar perang dan teror yang grafis juga dapat memperkuat *stress* (Silver dkk., 2013).

Paparan konten kekerasan yang menyebar luas di media sosial perlu segera ditanggulangi, karena riset menunjukkan bahwa paparan gambar yang mengandung konten kekerasan dapat meningkatkan *stress* dan memperburuk dampak psikologis pada sebagian audiens (Holman dkk., 2024). Sebagai upaya penanggulangan penyebarluasan konten kekerasan, dilakukan moderasi untuk memilah konten sebelum dapat dinyatakan layak untuk dipublikasikan. Moderasi konten umumnya mengandalkan metode manual, yakni konten yang terindikasi melanggar dilihat oleh peninjau manusia lalu diputuskan untuk dihapus atau dibatasi. Namun, moderasi manual memiliki kelemahan besar pada skala dan kecepatan karena volume unggahan sangat tinggi, YouTube menyebut setiap menit ratusan jam konten baru diunggah sehingga otomatisasi diperlukan untuk keputusan yang cepat (Google, t.t.). Selain itu, tugas moderasi sangat bergantung pada konteks dan memiliki risiko kesalahan yang cukup besar, sehingga interpretasi manusia sering tetap dibutuhkan

(Steiger dkk., 2021). Di sisi lain, pekerjaan sebagai moderator juga membawa beban psikologis karena dapat terpapar materi yang mengganggu dan menunjukkan gejala terkait trauma berulang, *stress* sekunder, serta dampak kesehatan mental seiring meningkatnya frekuensi paparan (Steiger dkk., 2021).

Karena keterbatasan tersebut, moderasi manual kemudian banyak dioptimalkan menjadi moderasi *hybrid* AI dan manusia, yakni AI membantu mendeteksi, menyaring, dan memprioritaskan konten pada skala besar, lalu manusia menangani evaluasi kasus yang tidak jelas dan keputusan yang membutuhkan konteks. Secara ilmiah, Gorwa, Binns, dan Katzenbach menjelaskan bahwa sistem moderasi algoritmik seperti *hash matching* dan *machine learning* semakin dipakai platform besar untuk moderasi, dan dorongan ini muncul karena moderasi manual tidak lagi *feasible* (Gorwa dkk., 2020). Contoh implementasi *hybrid* dapat dilihat di YouTube yang menyatakan keputusan kebijakan dilakukan dengan kombinasi sistem otomatis dan tinjauan manusia, sistem otomatis belajar dari data tinjauan manusia, dan pada banyak kasus konten hanya ditandai agar ditinjau reviewer manusia sebelum tindakan diambil, termasuk saat banding (Google, t.t.). Meski demikian, *hybrid* AI dan manusia tetap membutuhkan manusia karena kebijakan yang kontekstual dan ada pengecualian tertentu yang harus dinilai kasus per kasus, serta karena kesalahan deteksi dapat berdampak pada pengguna. Dari sisi kelemahan, riset terkait juga menekankan risiko sistem otomatis yang semakin tidak transparan dan sulit diaudit serta memunculkan isu keadilan dalam keputusan pada skala besar, sementara beban psikologis tidak hilang sepenuhnya karena manusia tetap berada pada tahap peninjauan konten bermasalah (Gorwa dkk., 2020).

Moderasi *hybrid* dapat dirancang bukan hanya oleh AI yang dibantu manusia, tetapi juga AI yang dibantu oleh AI, misalnya model ringan di perangkat berdaya rendah seperti perangkat *mobile* sebagai penyaring pertama dan model yang lebih kuat di server sebagai verifikator. Alurnya, semua konten diproses terlebih dahulu di *mobile* untuk keputusan cepat, lalu hanya sampel yang belum meyakinkan yang diteruskan ke server agar diputuskan oleh model yang lebih berat dengan kapasitas lebih tinggi (Shirin Abkenar dkk., 2023). Pola *offloading* berbasis tingkat keyakinan ini sejalan dengan konsep eksekusi inferensi lintas *end device* dan *cloud* yang dibahas pada riset *edge computing*, termasuk pendekatan yang memakai ambang *confidence* untuk menentukan apakah sebuah sampel diproses lokal atau diteruskan (Chen & Ran, 2019).

Pada konteks moderasi *hybrid* tersebut, efisiensi inferensi menjadi beban utama, karena *deep learning* membutuhkan daya komputasi besar untuk berjalan cepat, sementara perangkat seperti *smartphone* memiliki keterbatasan sumber daya sehingga eksekusi sering dibagi antara perangkat *mobile* dan server (Chen & Ran, 2019). Karena itu, arsitektur CNN yang umum dipilih untuk tahap *mobile* biasanya yang berorientasi efisiensi seperti MobileNet dan EfficientNet. Sebagai gambaran beban komputasi, MobileNetV3 Large 224 dan multiplier 1.0 memiliki 5.4 juta parameter, di sisi lain, EfficientNet B0 memiliki sekitar 5.3 juta parameter dan 0.39B FLOPs, (Tan & Le, 2020). Dalam konteks ini, walaupun MobileNetV3 dan EfficientNet sudah termasuk keluarga yang “efisien”, angka ratusan juta operasi dan jutaan

parameter pada konfigurasi umumnya tetap menunjukkan bahwa optimasi *on device* tetap krusial untuk memastikan pengalaman penggunaan yang baik di *smartphone*.

Fokus pada perangkat *mobile* menjadi penting karena implementasi moderasi konten sering kali dibutuhkan pada aplikasi yang berjalan di ponsel yang umum dipakai di lapangan, bukan perangkat dengan daya komputasi tinggi seperti *Personal Computer* (PC), sementara itu, perusahaan-perusahaan berskala global biasanya dapat mengandalkan komputasi terpusat untuk menjalankan model yang jauh lebih besar. Literatur *edge computing* menekankan bahwa kebutuhan *realtime* dan keterbatasan sumber daya di *end device* mendorong pemisahan inferensi agar sistem tetap responsif dan skalabel (Chen & Ran, 2019). Dengan pemisahan model ringan di *mobile* dan model berat di server, sistem tetap bisa menjaga kecepatan respon pada mayoritas kasus dengan *confidence level* tinggi, sekaligus mempertahankan akurasi pada kasus sulit saat server mengambil alih. Metode pemisahan dengan tujuan utama untuk meminimalkan selisih probabilitas antara kedua model tersebut dikenal sebagai Knowledge Distillation yang terdiri dari 2 bagian, yaitu *teacher model* dan *student model* (Noothout dkk., 2022). Studi komparatif menunjukkan adanya variasi performa yang signifikan di antara model-model ringan. Sebagai contoh, dalam sebuah evaluasi, Teachable Machine (97%) dan MobileNet (94%) menunjukkan akurasi yang jauh lebih unggul dibandingkan YOLO (53%) pada deteksi objek *multi class (Saqib dkk., 2025)*. Perbedaan ini menegaskan bahwa tidak semua model ringan cocok untuk setiap kasus implementasi Knowledge Distillation, sehingga analisis mendalam diperlukan.

Penelitian ini berfokus pada perancangan arsitektur sistem moderasi gambar *hybrid* yang stabil dan efisien untuk perangkat dengan sumber daya komputasi terbatas, dalam hal ini adalah perangkat *mobile,* dengan menerapkan Knowledge Distillation yang diharapkan dapat membantu model yang lebih sederhana untuk meniru perilaku model yang lebih kompleks. Pendekatan ini menggabungkan model besar di sisi server untuk akurasi maksimum dengan model ringan di sisi perangkat *mobile* untuk responsivitas dan efisiensi daya komputasi. Sehingga penelitian ini juga mengakomodir pemilihan model di sisi *client* yang paling efektif yang memiliki keseimbangan antara efisiensi komputasi dan akurasi tinggi, terutama untuk aplikasi *realtime* di perangkat dengan sumber daya terbatas (Saqib dkk., 2025).

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan, maka permasalahan dalam penelitian ini dapat dirumuskan sebagai berikut:

1. Bagaimana merancang sebuah arsitektur sistem moderasi gambar *hybrid* (*client-server*) yang efisien dan dapat diimplementasikan pada perangkat dengan sumber daya terbatas?
2. Bagaimana teknik *knowledge distillation* dapat meningkatkan performa model ringan di sisi klien dengan mewariskan pengetahuan dari model yang lebih besar di sisi *server*, tanpa menurunkan efisiensi secara signifikan?

1.3 Tujuan Penelitian

Sesuai dengan rumusan masalah di atas, tujuan dari penelitian ini adalah:
1. Merancang dan membangun *prototype* arsitektur sistem moderasi gambar *hybrid* yang membagi beban komputasi antara perangkat *client* dan *server*.
2. Menerapkan dan menganalisis efektivitas teknik *knowledge distillation* pada beberapa kandidat model *computer vision* ringan untuk menghasilkan model *student* dengan keseimbangan akurasi dan latensi terbaik.

1.4 Manfaat Penelitian

Penelitian ini diharapkan dapat memberikan serangkaian manfaat yang saling terkait, baik dari sisi pengembangan ilmu pengetahuan maupun penerapan praktis, sebagai berikut:
1. Memberikan kontribusi pada pengembangan arsitektur sistem terintegrasi AI, khususnya pada skema *hybrid client-server* yang efisien.
2. Menghasilkan sebuah arsitektur yang dapat menjadi acuan bagi lembaga pemerintah atau layanan publik untuk membangun sistem moderasi konten yang efektif dari segi biaya.
3. Menambah wawasan ilmiah mengenai perbandingan performa model-model *computer vision* ringan pada tugas spesifik moderasi gambar dalam lingkungan dengan sumber daya terbatas.

1.5 Batasan Masalah

Penelitian ini dibatasi pada ruang lingkup berikut:
1. Sistem yang dibangun berfokus pada klasifikasi citra biner menjadi dua kelas, yaitu *violence* (1) dan *safe* (0).
2. Objek penelitian dibatasi pada citra statis, penelitian tidak membahas moderasi untuk video, audio, maupun teks.
3. Model *teacher* ditempatkan pada sisi server dan model *student* ditempatkan pada sisi perangkat *mobile*. Implementasi dan pengujian *student* hanya difokuskan pada perangkat *mobile* (Android), bukan pada perangkat desktop, *web browser*, atau perangkat *embedded*/IoT lainnya.
4. Mekanisme *hybrid* yang diuji dibatasi pada skema keputusan lokal + eskalasi ke server berbasis nilai probabilitas/threshold yang ditetapkan, penelitian tidak membahas kebijakan eskalasi yang kompleks.
5. Evaluasi performa model dibatasi pada metrik AUC, F1-score, dan loss, serta pengukuran latensi inferensi (di *mobile* dan server) pada skenario pengujian yang ditentukan.
6. Pengukuran performa sistem end-to-end dibatasi pada parameter yang diamati pada pengujian (misalnya latensi lokal, latensi server saat eskalasi, dan total waktu respons), penelitian tidak memodelkan variasi kondisi jaringan secara menyeluruh.

7. Dataset yang digunakan adalah dataset yang disiapkan/digunakan pada penelitian ini, penelitian tidak membahas perluasan ke domain lain di luar karakteristik dataset tersebut maupun proses kurasi dataset berskala industri.
8. Aspek produksi seperti keamanan tingkat lanjut (misalnya enkripsi end-to-end atau rate limiting), manajemen beban (autoscaling), serta optimasi biaya cloud dibahas terbatas dan tidak menjadi fokus utama.
9. Penelitian tidak secara khusus membahas isu non-teknis lanjutan seperti interpretabilitas mendalam, pengujian bias/fairness lintas demografi, serta ketahanan terhadap serangan adversarial.

1.6 Landasan Teori

1.6.1 Moderasi Konten Digital

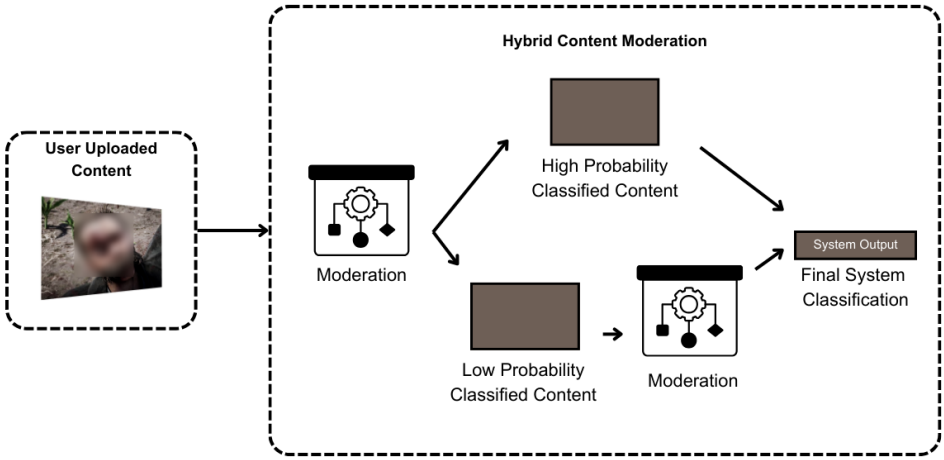
Pertumbuhan eksponensial dalam pembuatan konten digital telah menjadi aspek rutin dari budaya online, dengan volume data yang dibuat setiap hari pada tahun 2023 melebihi 402 juta TB (Kearns, 2025). Lonjakan ini menciptakan tantangan besar dalam pengelolaan dan pelacakan media, sehingga moderasi konten menjadi sebuah kebutuhan krusial. Dalam konteks ini, moderasi konten dapat dipahami sebagai upaya yang dilakukan oleh platform digital, untuk mengelola cara pengguna membuat dan berbagi konten. Lebih dari sekedar proses teknis, Clune dan McDaid (2023) mengkonseptualisasikan moderasi konten sebagai sebuah sistem akuntabilitas yang disebut "akuntabilitas media sosial" (*social media accountability*). Sistem ini dirancang secara spesifik untuk meminta pertanggungjawaban pengguna atas interaksi digital mereka, yang sangat dipengaruhi oleh agenda komersial, karena konten kontroversial dapat secara langsung memengaruhi pendapatan iklan.

Menurut Clune dan McDaid (2023), proses akuntabilitas ini berjalan melalui empat tahap utama yang terstruktur:

1. Menetapkan Standar (*Setting Standards*): Tahap pertama adalah mendefinisikan dan mengkomunikasikan ekspektasi kinerja kepada pengguna melalui "standar komunitas" (*community standards*). Standar ini menguraikan aturan dan perilaku yang diharapkan dari pengguna saat mereka membuat dan berbagi konten di platform.
2. Identifikasi dan Penilaian Pelanggaran (*Identifying and Adjudicating Violations*): Tahap ini berfokus pada penegakan standar komunitas melalui kombinasi tiga elemen utama: teknologi otomatis (seperti model *machine learning*) untuk deteksi proaktif, pelaporan dari pengguna untuk deteksi, dan tinjauan oleh manusia untuk melakukan penilaian akhir.
3. Penegakan dan Banding Pengguna (*Enforcement and User Appeal*): Setelah pelanggaran teridentifikasi, *platform* akan melakukan tindakan penegakan. Tindakan ini dapat berupa sanksi pada level konten (seperti penghapusan atau pelabelan) atau pada level akun (seperti sistem "serangan" atau *strike system* yang dapat berujung pada penangguhan). Pengguna yang merasa keputusan moderasi tidak adil diberikan kesempatan untuk mengajukan banding.

4. Pengungkapan (*Disclosure*): Sebagai tahap akhir, *platform* mempublikasikan Laporan Moderasi Konten (*Content Moderation Reports*) secara berkala. Laporan ini bertujuan memberikan transparansi kepada publik mengenai skala dan jenis aktivitas penegakan yang telah dilakukan oleh platform.

Meskipun kerangka kerja di atas memberikan struktur yang jelas, implementasinya di lapangan masih menghadapi tantangan praktis yang signifikan. Moderasi konten yang sepenuhnya manual sering kali menjadi tugas yang membebani, lambat, dan rentan terhadap kesalahan manusia (*error-prone*), terutama dengan volume konten yang sangat masif. Di sisi lain, moderasi yang sepenuhnya otomatis, meskipun cepat, berisiko tinggi menghasilkan positif palsu (*false positives*) karena input pengguna yang tidak dapat diprediksi dan model *computer vision* yang terkadang kurang presisi dibandingkan manusia. Kegagalan dalam menghasilkan keputusan yang akurat dapat mengurangi kepercayaan pengguna (Kearns, 2025).



Gambar 1. Skema arsitektur moderasi konten hibrida client–server (student model di perangkat dan teacher model di server)

Pada praktik moderasi konten yang umum, istilah *hybrid* sering merujuk pada penggabungan beberapa komponen dalam satu alur keputusan, misalnya filter berbasis aturan (*rule-based* seperti *keyword*, *blocklist*, atau heuristik), satu model *machine learning*, atau *human review* untuk konten yang dilaporkan atau berada pada area abu-abu. Dalam skema seperti ini, “hibriditas” lebih menekankan kombinasi sumber pengambil keputusan (aturan–AI–manusia) serta mekanisme operasional, bukan pada pembagian inferensi model ke dalam dua tingkat komputasi yang saling melengkapi.

Berbeda dari pendekatan tersebut, *hybrid* pada penelitian ini didefinisikan sebagai hibriditas pada sisi komputasi dan tingkat model. Sistem menjalankan moderasi tahap pertama di perangkat (*on-device*) menggunakan model *student* yang

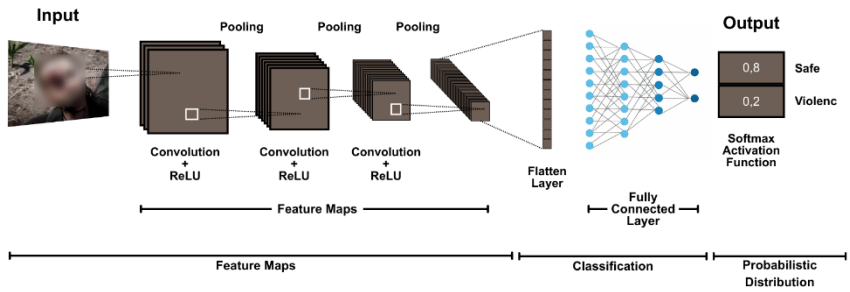
ringan untuk menghasilkan prediksi awal secara cepat. Konten yang terklasifikasi dengan keyakinan tinggi dapat langsung diselesaikan pada tahap ini. Sebaliknya, ketika prediksi berada pada kondisi *low-confidence* atau ambigu, konten akan dieskalasi ke moderasi tahap kedua di server (*cloud*) menggunakan model *teacher* yang lebih kuat untuk melakukan verifikasi sebelum keputusan akhir ditetapkan. Dengan demikian, pemicu perpindahan tahap bukan laporan pengguna atau aturan manual, melainkan indikator ketidakpastian prediksi yang dapat diatur melalui ambang tertentu.

Definisi *hybrid* seperti ini membawa implikasi teknis yang jelas pada sistem. Pertama, latensi rata-rata dapat ditekan karena sebagian besar konten selesai pada tahap *on-device* tanpa perlu komunikasi jaringan. Kedua, kualitas keputusan pada kasus-kasus sulit meningkat karena tetap tersedia pemeriksaan lanjutan oleh model yang lebih kuat. Ketiga, *Knowledge Distillation* menjadi komponen penting untuk memperkecil kesenjangan kemampuan student terhadap *teacher*, sehingga akurasi tahap pertama dapat ditingkatkan dan frekuensi eskalasi dapat ditekan tanpa mengorbankan ketelitian klasifikasi akhir.

1.6.2 Convolutional Neural Network

Convolutional Neural Network (CNN) merupakan salah satu kelas model yang paling berpengaruh dalam ranah *deep learning* dan telah menjadi pendekatan utama dalam berbagai tugas pengenalan pola visual, seperti deteksi dan identifikasi objek. Arsitektur CNN dirancang secara spesifik untuk mengolah data dengan struktur *array* berganda (misalnya, citra berwarna yang memiliki tiga lapisan *array* dua dimensi untuk saluran warnanya) (Lecun dkk., 2015).

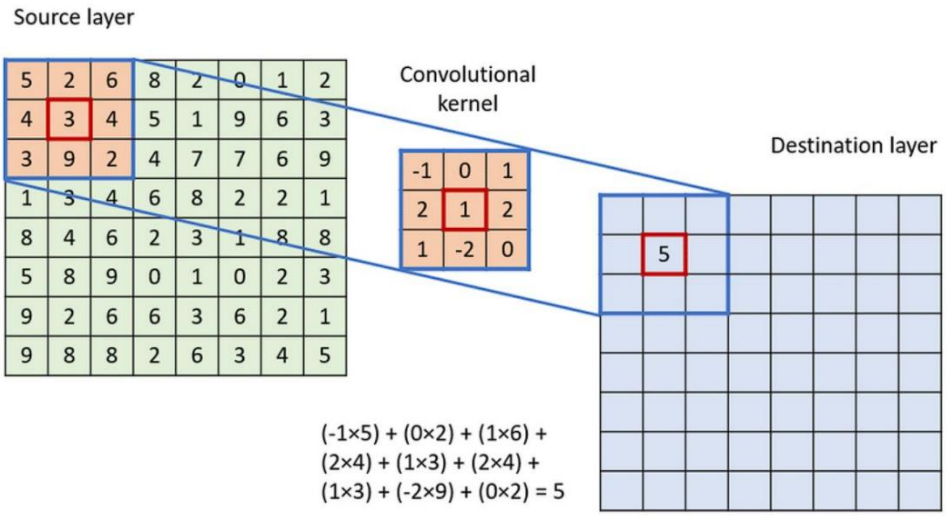
Secara konseptual, perbedaan mendasar CNN terletak pada kemampuannya untuk mempelajari fitur-fitur secara otomatis dan berlapis langsung dari data, menggantikan metode lama yang bergantung pada fitur yang dirancang secara manual oleh manusia (*hand-crafted features*) (Lecun dkk., 2015). Keberhasilan model ini didukung oleh dua asumsi kuat yang sebagian besar valid mengenai sifat data citra, yakni lokalitas ketergantungan piksel dan stasionaritas statistik (Krizhevsky dkk., 2012).



Gambar 2. Visualisasi arsitektur CNN

Arsitektur CNN yang terbukti sukses dan efisien dalam tugas pengenalan objek visual memanfaatkan empat pilar konseptual utama. Keempat ide tersebut adalah: koneksi lokal (*local connections*), bobot bersama (*shared weights*), *pooling* (sering juga disebut *subsampling*), dan penggunaan banyak lapisan (*deep layers*) untuk membangun arsitektur yang hierarkis (Krizhevsky dkk., 2012; Lecun dkk., 2015).

1. Lapisan Konvolusional (*Convolutional Layer*)



Gambar 3. Ilustrasi proses konvolusi (Podareanu dkk., 2019)

Lapisan ini merupakan inti dari CNN yang berfungsi untuk ekstraksi fitur (*feature extraction*) dengan cara mendeteksi pola atau fitur lokal dari input (atau *feature map*) lapisan sebelumnya (Lecun dkk., 2015).

a) Peta Fitur (*Feature Maps*): Output dari lapisan konvolusional terdiri dari sekumpulan "peta fitur". Setiap unit (*neuron*) dalam satu peta fitur terhubung hanya ke sebuah area lokal kecil dari input (dikenal sebagai *receptive field*). Koneksi ini diatur oleh seperangkat bobot yang disebut filter atau kernel.
b) Bobot Bersama (*Shared Weights*): Ini adalah prinsip fundamental CNN. Semua unit dalam satu peta fitur yang sama berbagi satu set filter (bobot) yang identik. Asumsinya adalah jika sebuah motif (misalnya, tepi atau sudut) penting untuk dideteksi di satu bagian citra, motif itu juga penting di bagian lain (asumsi stasionaritas statistik). Dengan "menggeser" filter yang sama ke seluruh input, model dapat mendeteksi fitur yang sama di lokasi berbeda dengan jumlah parameter yang jauh lebih sedikit. Secara matematis, operasi ini adalah konvolusi diskret.
c) Non-Linearitas: Hasil dari operasi konvolusi (penjumlahan berbobot lokal) kemudian dilewatkan melalui fungsi aktivasi non-linear, seperti ReLU

(Rectified Linear Unit). Penerapan non-linearitas ini sangat krusial, tanpa konsep ini, tumpukan beberapa lapisan konvolusional secara matematis hanya akan setara dengan satu lapisan konvolusional tunggal (karena komposisi dari beberapa fungsi linear tetaplah linear). Fungsi non-linear seperti ReLU memungkinkan model untuk mempelajari representasi yang jauh lebih kompleks dan hierarkis.

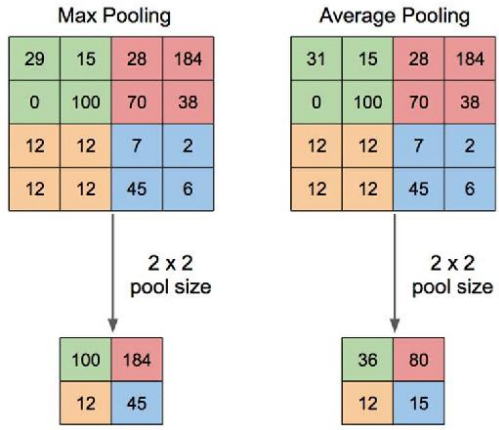
2. Lapisan *Pooling* (*Pooling Layer*)

Lapisan *pooling*, yang sering juga disebut sebagai lapisan *subsampling*, memiliki peran fundamental untuk mengagregasi atau mengonsolidasikan fitur-fitur yang memiliki kemiripan semantik menjadi satu representasi (Lecun dkk., 2015).

Tujuan utamanya adalah untuk menciptakan invariansi translasi (ketahanan terhadap pergeseran) dalam skala kecil. Dengan mereduksi informasi posisi yang presisi, lapisan ini memungkinkan representasi model tetap stabil (hanya bervariasi sedikit) meskipun elemen pada lapisan sebelumnya mengalami pergeseran posisi atau perubahan penampilan minor (Krizhevsky dkk., 2012).

Lapisan ini memiliki beberapa peran konseptual yang fundamental dalam arsitektur CNN (Khosla, 2025):

a) Konsolidasi Fitur: Mengagregasi atau mengonsolidasikan fitur-fitur yang memiliki kemiripan semantik di dalam sebuah patch lokal menjadi satu representasi tunggal.
b) Invariansi Translasi (*Translation Invariance*): Ini adalah tujuan utamanya. Dengan mereduksi informasi posisi yang presisi, lapisan *pooling* menciptakan ketahanan (invariansi) terhadap pergeseran atau distorsi kecil. Representasi model tetap stabil meskipun fitur pada input bergeser sedikit.
c) Pengurangan Komputasi: Dengan mengurangi ukuran *feature map*, lapisan pooling secara efektif menurunkan jumlah parameter dan beban komputasi pada lapisan-lapisan berikutnya di dalam jaringan.
d) Regularisasi: Pengurangan dimensionalitas ini juga bertindak sebagai bentuk regularisasi yang membantu mencegah overfitting.



Gambar 4. Ilustrasi proses pooling (Yani dkk., 2019)

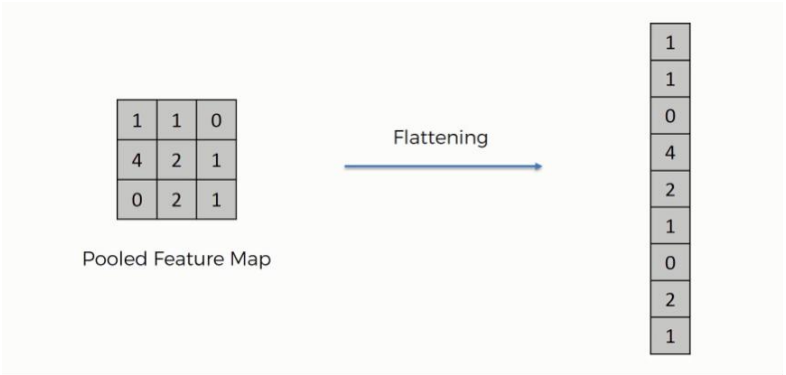
Terdapat beberapa varian dan strategi pooling, dengan dua yang paling umum adalah (Zafar dkk., 2024):

a) Max-Pooling: Ini adalah metode yang paling dominan digunakan dalam arsitektur CNN modern. Max-pooling bekerja dengan cara memilih nilai maksimum (aktivasi yang paling kuat) dari setiap patch lokal di dalam feature map. Pendekatan ini terbukti efektif karena ia secara agresif meneruskan fitur yang paling menonjol (prominent features) sinyal terkuat yang terdeteksi di area tersebut ke lapisan berikutnya. Sebuah studi komparatif untuk klasifikasi skoliosis, misalnya, menunjukkan bahwa penggunaan max-pooling mampu mencapai akurasi yang lebih tinggi (84.6%) dibandingkan dengan average-pooling.

b) Average-Pooling: Metode ini bekerja dengan menghitung nilai rata-rata dari semua elemen di dalam patch lokal.

Pada implementasi arsitektur CNN berskala besar, varian yang disebut *overlapping pooling* telah terbukti memberikan hasil yang efektif. Dalam skema tersebut, patch yang diringkas saling tumpang tindih, yang dicapai dengan mengatur jarak spasi atau *stride* (s) lebih kecil daripada ukuran *patch* ($\boldsymbol{z} \times \boldsymbol{z}$). Penggunaan teknik ini terbukti secara empiris dapat membantu menekan tingkat kesalahan (*error rate*) model (Krizhevsky dkk., 2012).

3. Lapisan Penjajaran (*Flatten Layer*)



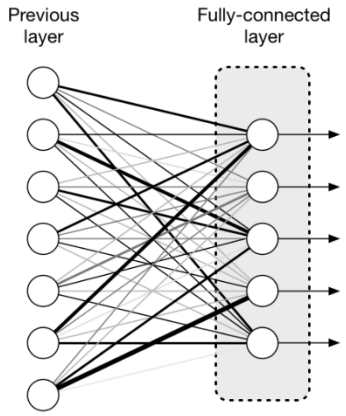
Gambar 5. *Ilustrasi flatten layer* (*Convolutional Neural Networks (CNN): Step 3 - Flattening*, 2018)

Lapisan penjajaran atau *Flatten* berfungsi sebagai "jembatan" antara dua blok arsitektur. *Output* dari blok konvolusional/*pooling* terakhir adalah sebuah tensor 3D (misalnya, berdimensi lebar tinggi saluran, seperti $\mathbf{7 \times 7 \times 512}$). Namun, lapisan klasifikasi standar (jaringan saraf *feed-forward*) memerlukan input berupa vektor 1D.

Lapisan *Flatten* adalah lapisan non-parametrik (tidak memiliki bobot yang dapat dilatih) yang mekanisme kerjanya sederhana: "meratakan" atau "menjajarkan" tensor 3D tersebut menjadi satu vektor 1D yang sangat panjang (misalnya, $\mathbf{7 \times 7 \times 512 = 25.088}$ elemen).

Meskipun sederhana, lapisan *Flatten* merepresentasikan sebuah momen krusial dalam pemrosesan informasi. Lapisan Konvolusional dan *Pooling* secara inheren bersifat spatially-aware (sadar spasial); mereka secara eksplisit menggunakan dan mempertahankan informasi tentang lokasi relatif fitur (melalui prinsip koneksi lokal). Momen ketika data melewati lapisan Flatten adalah titik di mana informasi spasial secara eksplisit dihancurkan. Model beralih dari penalaran spasial ("ada tekstur bulu di sini dan fitur mata di sana") menjadi penalaran fitur holistik ("terdapat fitur 'bulu', 'mata', dan 'telinga' dalam gambar ini, di mana pun lokasinya"). Transisi ini penting untuk memungkinkan lapisan berikutnya membuat keputusan klasifikasi global berdasarkan kehadiran fitur-fitur tingkat tinggi yang telah diekstraksi.

4. Lapisan Terhubung Penuh (*Fully-Connected Layers*)



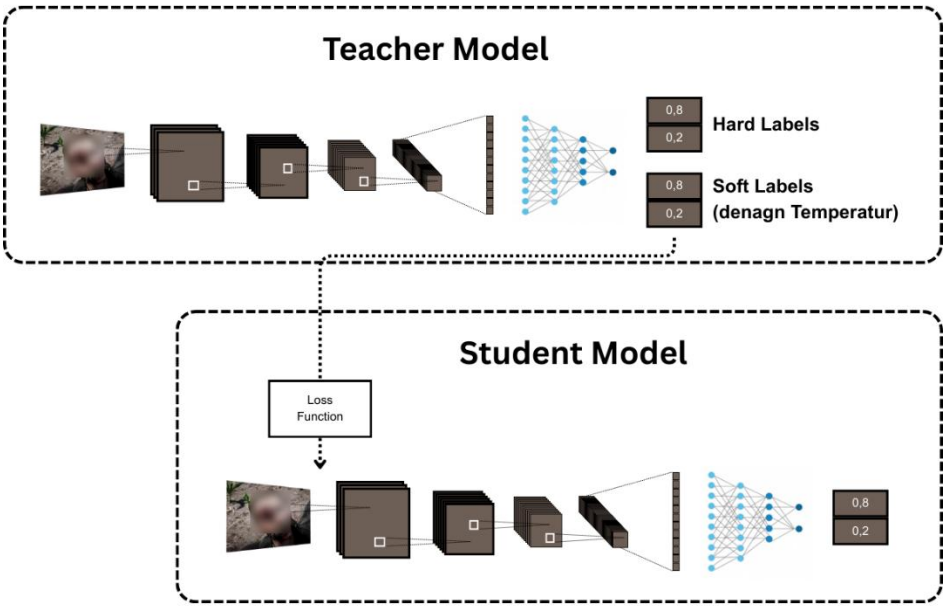
Gambar 6. Ilustrasi fully connected layers *(Thesis & Top, 2018)*

Setelah serangkaian lapisan konvolusional dan *pooling* selesai mengekstraksi fitur-fitur visual dari input, arsitektur CNN biasanya diakhiri dengan satu atau lebih Lapisan Terhubung Penuh (*Fully-Connected Layers*). Lapisan ini berfungsi sebagai "klasifikator" (*classifier*) dalam jaringan, yang bertugas menginterpretasikan fitur-fitur hierarkis yang telah dipelajari.

Pada tahap akhir, output dari lapisan terhubung penuh yang terakhir (sering disebut logits atau skor mentah) akan diproses oleh sebuah fungsi aktivasi Softmax. Untuk tugas klasifikasi multi-kelas (misalnya, 1000 kelas pada dataset ImageNet), fungsi Softmax ini akan mengonversi skor-skor tersebut menjadi sebuah distribusi probabilitas di seluruh label kelas yang ada, di mana total probabilitasnya adalah 1.

1.6.3 Knowledge Distillation

Knowledge Distillation adalah sebuah teknik dalam *machine learning* yang bertujuan untuk memindahkan dan memadatkan pengetahuan dari model yang sudah dilatih, besar, dan kompleks (dikenal sebagai *teacher model*) ke model yang lebih kecil, lebih cepat, dan lebih efisien (dikenal sebagai *student model*). Metode ini merupakan strategi kompresi model yang penting, terutama untuk jaringan saraf dalam (*deep neural networks*) berukuran besar (Noothout dkk., 2022).



Gambar 7. Ilustrasi Knowledge Distillation

Kebutuhan akan *knowledge distillation* muncul karena meskipun *ensemble model*, yaitu koleksi model yang prediksinya digabungkan seringkali memberikan peningkatan performa yang luar biasa pada hampir semua algoritma *machine learning*, penggunaannya menjadi tidak praktis saat waktu inferensi, sumber daya komputasi, atau memori menjadi kendala yang ketat. Misalnya, di perangkat portable atau dalam sistem skala besar yang melayani banyak pengguna (seperti Google). Analogi yang sering digunakan adalah serangga, yang memiliki bentuk larva yang dioptimalkan untuk mengekstrak struktur data (fase pelatihan yang membutuhkan komputasi besar) dan bentuk dewasa yang dioptimalkan untuk perjalanan dan reproduksi (fase *deployment* yang membutuhkan latensi dan sumber daya rendah) (Bucilă dkk., 2006; Hinton dkk., 2015).

1. Mekanisme Transfer Pengetahuan

   Konsep pengetahuan (*knowledge*) dalam model yang telah dilatih cenderung diidentifikasi dengan nilai parameter yang dipelajari. Namun, pandangan yang lebih abstrak membebaskan pengetahuan tersebut dari instansiasi tertentu, mendefinisikannya sebagai pemetaan yang dipelajari dari vektor *input* ke vektor *output* (Hinton dkk., 2015).

   Dalam model besar (*cumbersome models*) yang dilatih untuk mendiskriminasi sejumlah besar kelas, meskipun tujuan pelatihan normal adalah memaksimalkan log probabilitas dari jawaban yang benar, efek samping dari pembelajaran adalah bahwa model juga menetapkan probabilitas untuk semua jawaban yang salah. Probabilitas relatif dari jawaban yang salah inilah yang mengungkapkan banyak hal tentang bagaimana model besar tersebut cenderung melakukan generalisasi. Sebagai contoh, gambar sebuah mobil

mungkin hanya memiliki probabilitas yang sangat kecil untuk salah diklasifikasikan sebagai truk sampah, tetapi kesalahan tersebut masih berkali-kali lipat lebih mungkin terjadi daripada salah diklasifikasikan sebagai wortel. Informasi berharga ini mendefinisikan struktur kesamaan yang kaya (*rich similarity structure*) atas data (Hinton dkk., 2015).

2. Target Lunak (*Soft Targets*) dan Suhu (*Temperature*)

Cara yang paling jelas untuk mentransfer kemampuan generalisasi model besar adalah dengan menggunakan probabilitas kelas yang dihasilkannya, dikenal sebagai "target lunak" (*soft targets*) untuk melatih model *student*. Target lunak memiliki keuntungan signifikan karena memiliki entropi tinggi, yang berarti mereka menyediakan lebih banyak informasi per kasus pelatihan dibandingkan dengan target keras (*hard targets* atau label *ground truth* biasa). Tingginya entropi ini juga menghasilkan varians yang jauh lebih kecil dalam gradien antar kasus pelatihan, memungkinkan model student untuk dilatih dengan lebih sedikit data dan laju pembelajaran (*learning rate*) yang lebih tinggi (Hinton dkk., 2015; Noothout dkk., 2022).

Pada tugas-tugas seperti MNIST, di mana model teacher hampir selalu menghasilkan jawaban yang benar dengan keyakinan yang sangat tinggi, sebagian besar informasi berharga tentang fungsi yang dipelajari justru terletak pada rasio probabilitas yang sangat kecil dalam target lunak. Probabilitas yang sangat dekat dengan nol ini memiliki sedikit pengaruh pada fungsi cross-entropy standar yang digunakan selama tahap transfer (Hinton dkk., 2015).

Untuk mengatasi masalah ini, solusi yang lebih umum yang disebut "*distillation*" adalah dengan memperkenalkan suhu (*temperature*, $\boldsymbol{T}$) pada lapisan softmax akhir. Jaringan saraf secara umum menghasilkan probabilitas kelas $\boldsymbol{q_i}$ dari logit $\boldsymbol{z_i}$ menggunakan fungsi *softmax*:

$$\boldsymbol{q_i} = \frac{\boldsymbol{e}^{\boldsymbol{z_i}/\boldsymbol{T}}}{\sum_{\boldsymbol{j}} \boldsymbol{e}^{\boldsymbol{z_j}/\boldsymbol{T}}} \qquad (1)$$

Keterangan

$\boldsymbol{q_i}$	: probabilitas untuk kelas ke-$\boldsymbol{i}$
$\boldsymbol{z_i}$	: logit untuk kelas ke-$\boldsymbol{i}$
$\boldsymbol{z_j}$	: logit untuk kelas ke-$\boldsymbol{j}$
$\boldsymbol{T}$	: temperatur

Di mana $\boldsymbol{T}$ biasanya diatur ke 1. Penggunaan nilai $\boldsymbol{T}$ yang lebih tinggi akan menghasilkan distribusi probabilitas yang lebih lunak (*softer probability distribution*) di atas kelas. Dalam bentuk distilasi yang paling sederhana, pengetahuan ditransfer dengan melatih model student pada set transfer dan menggunakan distribusi target lunak yang dihasilkan oleh model teacher dengan *T* yang tinggi. Student model dilatih menggunakan *T* yang sama tingginya untuk

mencocokkan target lunak ini, tetapi setelah dilatih, ia kembali menggunakan suhu *T*=1 untuk membuat prediksi normal (Hinton dkk., 2015).

Metode kompresi model yang menggunakan logits (input ke *softmax*) sebagai target untuk pembelajaran model kecil sebenarnya merupakan kasus khusus dari distilasi. Dalam batas suhu yang sangat tinggi ($\boldsymbol{T}$ tinggi dibandingkan dengan besaran logit), distilasi terbukti setara dengan meminimalkan selisih kuadrat logit

$$\frac{\mathbf{1}}{\mathbf{2}}(\boldsymbol{z_i} - \boldsymbol{v_i})^{\mathbf{2}} \qquad (2)$$

Keterangan
$\boldsymbol{z_i}$: logit model *student* untuk kelas ke-$\boldsymbol{i}$
$\boldsymbol{v_i}$: logit model *teacher* untuk kelas ke-$\boldsymbol{i}$

asalkan logit tersebut disetarakan nol-rata-ratanya (*zero-meaned*) untuk setiap kasus transfer (Bucilă dkk., 2006; Hinton dkk., 2015).

3. Fungsi Objektif Pelatihan *Student*

Ketika label yang benar (*hard labels*) untuk set transfer diketahui, kinerja model student dapat ditingkatkan secara signifikan. Cara terbaik untuk mencapai ini adalah dengan menggunakan rata-rata berbobot dari dua fungsi objektif yang berbeda (Hinton dkk., 2015; Noothout dkk., 2022):

a) Fungsi objektif pertama: *Cross-entropy* dengan target lunak, dihitung menggunakan suhu $\boldsymbol{T}$ tinggi yang sama dengan yang digunakan model teacher.
b) Fungsi objektif kedua: *Cross-entropy* dengan label yang benar (hard labels), dihitung menggunakan logit yang sama tetapi pada suhu $\boldsymbol{T} = \mathbf{1}$.

Karena besaran gradien yang dihasilkan oleh target lunak berskala sebagai $\frac{\mathbf{1}}{\boldsymbol{T^2}}$, penting untuk mengalikannya dengan $\boldsymbol{T^2}$ saat menggabungkan target keras dan target lunak. Hal ini memastikan bahwa kontribusi relatif dari kedua target tersebut tetap kira-kira tidak berubah jika suhu distilasi $\boldsymbol{T}$ diubah (Hinton dkk., 2015).

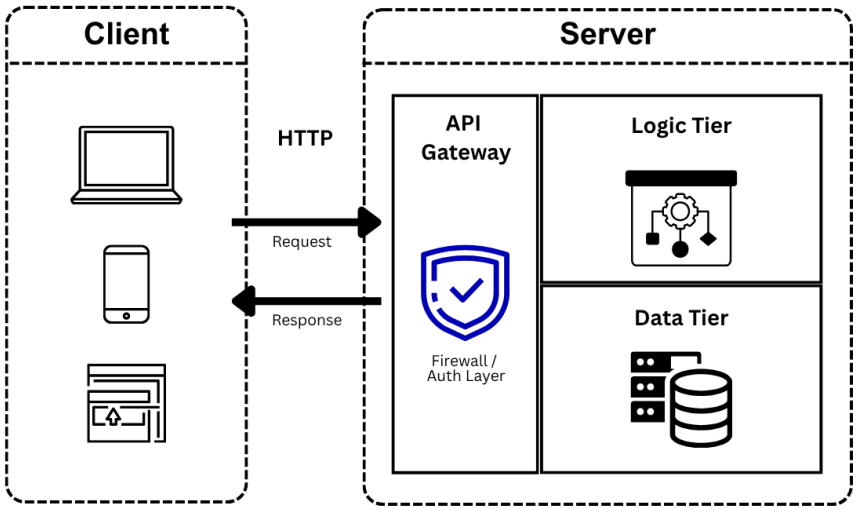
Knowledge distillation bertindak sebagai bentuk regulasi yang sangat efektif, memungkinkan model untuk bergeneralisasi dengan baik meskipun dengan jumlah data pelatihan yang jauh lebih sedikit. Sebagai contoh, dalam eksperimen *speech recognition*, model distilasi mampu memperoleh hampir semua informasi dari set pelatihan lengkap meskipun dilatih hanya dengan 3% dari data tersebut, dan bahkan tidak memerlukan *early stopping* (penghentian dini) karena tidak terjadi *overfitting (Bucilă dkk., 2006; Hinton dkk., 2015)*.

Keunggulan praktis dari teknik ini sangat substansial untuk *deployment*. Dalam aplikasi segmentasi citra medis, misalnya, model distilasi yang dihasilkan terbukti 4 hingga 89 kali lebih kecil dalam jumlah parameter yang dapat dilatih, membutuhkan 4 hingga 10 kali lebih sedikit penggunaan memori GPU, dan 5

hingga 8 kali lebih cepat dalam waktu inferensi dibandingkan dengan ensemble yang menjadi teachernya. Dengan mempertahankan kinerja ensemble atau bahkan melebihinya dalam beberapa kasus, model distilasi menjadi solusi yang jauh lebih layak untuk lingkungan yang sensitif terhadap biaya dan waktu (Bucilă dkk., 2006; Noothout dkk., 2022).

1.6.4 Arsitektur *Client-Server*

Arsitektur *client-server* adalah sebuah model perangkat lunak di mana sumber daya dan permintaan layanan diproses melalui jaringan. Dalam model komputasi ini, satu atau lebih komputer yang disebut klien akan meminta sumber daya dari komputer lain yang disebut server melalui koneksi jaringan. Server, setelah menerima permintaan, akan memprosesnya dan merespons klien dengan tepat (Nyabuto, 2023).



Gambar 8. Ilustrasi arsitektur client-server

Model ini memungkinkan banyak pengguna untuk secara simultan mengakses dan menggunakan sumber daya yang tersimpan di lokasi terpusat. Klien, seperti *web browser* atau aplikasi seluler, biasanya adalah komputer sederhana yang hanya menjalankan program klien untuk meminta layanan, sehingga tidak memerlukan banyak konfigurasi yang kompleks. Sebaliknya, server adalah komputer yang kompleks dengan daya pemrosesan yang tinggi, yang tidak hanya menjalankan program untuk memproses permintaan, tetapi juga mekanisme keamanan untuk memvalidasi dan mengautentikasi permintaan pengguna (Nyabuto, 2023).

Komunikasi antara klien dan server dimungkinkan melalui komunikasi antar-proses (*inter-process communication*) menggunakan protokol standar seperti

Hypertext Transfer Protocol (HTTP), *File Transfer Protocol* (FTP), atau *Simple Mail Transfer Protocol* (SMTP). Arsitektur ini juga dikenal memiliki skalabilitas yang baik, baik secara horizontal (menambah lebih banyak server) maupun vertikal (meningkatkan kemampuan server seperti RAM dan CPU) (Nyabuto, 2023).

Arsitektur client-server dapat diimplementasikan dalam berbagai konfigurasi berdasarkan jumlah lapisan (*tier*) yang terlibat (Nyabuto, 2023):

1. *Two-Tier Architecture* (Arsitektur Dua Lapis): Dalam model ini, hanya ada dua tingkatan perangkat komputasi: klien dan server. Klien menghosting antarmuka pengguna dan mengirim permintaan, sedangkan server (yang menghosting sistem aplikasi dan basis data) memprosesnya. Model ini efektif untuk aplikasi sederhana dengan sedikit pengguna yang mengakses sistem secara bersamaan.
2. *Three-Tier Architecture* (Arsitektur Tiga Lapis): Model ini memperkenalkan lapisan tambahan, yaitu *Logic or Application Tier*, yang memisahkan logika bisnis dari lapisan klien dan lapisan data. Tiga komponen utamanya adalah:
   a) *Client Tier*: Berfungsi menampilkan informasi dan mengirim permintaan.
   b) *Logic or Application Tier*: Lapisan tengah yang memproses permintaan dari klien dan mengambil data dari lapisan data.
   c) *Data Tier*: Lapisan yang bertanggung jawab untuk penyimpanan dan pengambilan informasi menggunakan sistem manajemen basis data. Arsitektur ini menawarkan pemeliharaan yang lebih mudah, skalabilitas yang lebih baik, dan keamanan yang ditingkatkan karena semua permintaan harus melalui lapisan aplikasi, mencegah manipulasi basis data secara langsung oleh pengguna.
3. *N-Tier Architecture* (Arsitektur Multi-Lapis): Disebut juga arsitektur multi-tier, model ini memisahkan logika presentasi, pemrosesan, dan manajemen data secara logis dan fisik menggunakan beberapa server. Model ini sangat skalabel dan efisien, sering diterapkan dalam pengembangan aplikasi besar seperti sistem *Customer Relationship Management* (CRM).

1.6.5 Perangkat *Mobile*

Secara konseptual, perangkat *mobile* (perangkat bergerak) didefinisikan sebagai perangkat komputasi portabel. Menurut *National Institute of Standards and Technology* (NIST), perangkat ini memiliki empat karakteristik utama (Franklin dkk., 2020):

1. Faktor bentuk kecil sehingga mudah dibawa oleh satu individu.
2. Dirancang untuk beroperasi tanpa koneksi fisik (misalnya, mentransmisikan atau menerima informasi secara nirkabel).
3. Memiliki penyimpanan data lokal.
4. Menyertakan sumber daya daya mandiri (yaitu, baterai). Contoh umum dari perangkat ini termasuk smartphone, tablet, dan E-reader.

Komputasi *mobile*, atau *mobile computing*, adalah teknologi yang memungkinkan transmisi data, suara, dan video melalui perangkat nirkabel tanpa

harus terhubung ke tautan fisik tetap. Paradigma ini dapat dilihat sebagai "integrasi komputer portabel dan jaringan nirkabel". Namun, landasan teoritis dari komputasi *mobile* didasarkan pada tantangan utamanya yaitu koneksi nirkabel yang "bersifat sementara dengan periode pemutusan" serta keterbatasan sumber daya yang fundamental (Alakbarov, 2021).

Tantangan desain utama untuk perangkat mobile berpusat pada dua kendala utama yaitu daya (baterai) dan konektivitas. Meskipun kemampuan perangkat terus meningkat, literatur akademis menegaskan bahwa "perbedaan keterbatasan sumber daya antara perangkat mobile dan perangkat *non-mobile* akan selalu ada". Peningkatan permintaan kinerja, seperti integrasi akselerator AI dan konektivitas 5G, menciptakan "tantangan yang menghambat dengan output termal dan konsumsi daya". Selain keterbatasan sistemik ini, perangkat mobile juga menghadapi tantangan interaksi (Mobile HCI), yaitu "upaya untuk menyediakan layanan komputasi yang kuat melalui antarmuka kecil" , yang menuntut desain aplikasi yang sangat efisien (Alakbarov, 2021; Pereira dkk., 2020).

Untuk mengatasi keterbatasan sumber daya komputasi dan daya tahan baterai inilah, paradigma arsitektur *Mobile Cloud Computing* (MCC) muncul. MCC didefinisikan sebagai "kombinasi perangkat *mobile* dan komputasi *cloud*," di mana tugas-tugas komputasi intensif dipindahkan (*offloaded*) dari perangkat ke infrastruktur *cloud* yang kuat. Namun, arsitektur ini menghadapi hambatannya sendiri, yaitu "latensi WAN yang panjang" yang dapat "merusak kegunaan" dan respons interaktif. Sebagai respons, arsitektur *Edge Computing* (seperti teori Cloudlet dari Satyanarayanan) diusulkan, yang memanfaatkan proksimitas untuk menyediakan sumber daya komputasi yang kaya di tepi jaringan ("*data center in a box*"). Perangkat mobile kemudian dapat mengakses sumber daya ini melalui koneksi WLAN berlatensi rendah. Pendekatan hybrid inilah yang menjadi relevan dalam penelitian ini, di mana perangkat mobile bertindak sebagai client yang perlu menyeimbangkan beban kerja antara komputasi lokal (di perangkat) dan komputasi jarak jauh (di server) (Lewis & Lago, 2015; Satyanarayanan et al., 2009).

BAB II
METODE PENELITIAN

2.1 Tempat dan Waktu Penelitian

2.1.1 Tempat Penelitian

Seluruh proses penelitian dilaksanakan di Ruang Belajar dan Laboratorium Rekayasa Perangkat Lunak, Program Studi Sistem Informasi, Departemen Matematika, FMIPA, Universitas Hasanuddin.

2.1.2 Waktu Penelitian

Penelitian dilakukan selama 16 pekan dengan tahapan pelaksanaan mengacu pada tahapan penelitian yang akan dipaparkan pada bagian berikutnya.

Tabel 1. Diagram waktu pelaksanaan penelitian

No.	Tahapan Penelitian	Pekan / Bulan															
		Agustus				September				Oktober				November			
		1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1	Perancangan Arsitektur	■	■	■	■												
2	Pengembangan Model					■	■	■	■	■	■						
3	Analisis Performa Model									■	■	■	■				
4	Analisis Performa Sistem											■	■	■	■	■	■

2.2 Instrumen Penelitian

Penelitian dilakukan dengan menggunakan instrumen berupa perangkat keras dan perangkat lunak yang digunakan selama proses pengembangan dan evaluasi sistem.

2.2.1 Perangkat Keras

Selama proses pengembangan dan evaluasi model *computer vision,* serta *deployment* model untuk *API Service*, peneliti menggunakan *Personal Computer* (PC) dengan spesifikasi sebagai berikut:

Tabel 2. Spesifikasi perangkat server yang digunakan selama pengembangan sistem

Komponen	Spesifikasi
Sistem Operasi	Windows 11 (WSL2 + Ubuntu24.04)
Prosesor	12th Gen Intel(R) Core(TM) i7-12700
RAM	16 GB
Penyimpanan	512 GB
GPU	NVIDIA GeForce RTX 4060 Ti

Selain PC untuk pengembangan model dan *deployment* model *teacher*, peneliti menggunakan perangkat *mobile* yang menjalankan model *student* sekaligus mencatat metrik hasil klasifikasi sebagai laporan untuk hasil penelitian ini dengan spesifikasi sebagai berikut:

Tabel 3. Spesifikasi perangkat mobile yang digunakan selama pengembangan dan implementasi sistem

Komponen	Spesifikasi
Merek & Model perangkat	Oppo Reno6 4G
Sistem Operasi & Versi Android	ColorOS 11.1 berbasis Android 13
Chipset / SoC	Qualcomm Snapdragon 720G, 8 nm
CPU	Octa-core Kryo 465
GPU	Adreno 618
RAM	8 GB LPDDR4X
Penyimpanan	128 GB (UFS 2.1), dukung microSD
Resolusi layar	6.4" AMOLED, 2400×1080 (FHD+), 90 Hz
Kamera (utama/belakang) & resolusi	64 MP

2.2.2 Perangkat Lunak dan Pustaka

Selain perangkat keras, proses pengembangan sistem juga menggunakan beberapa perangkat lunak dan *library* untuk beberapa kebutuhan seperti merancang arsitektur, melatih model, *deployment* aplikasi pendukung, dan berbagai kebutuhan lainnya terkait berjalannya sistem.

Berbagai perangkat lunak yang digunakan dijelaskan pada daftar berikut:

Tabel 4. Daftar perangkat lunak dan pustaka yang digunakan selama pengembangan sistem

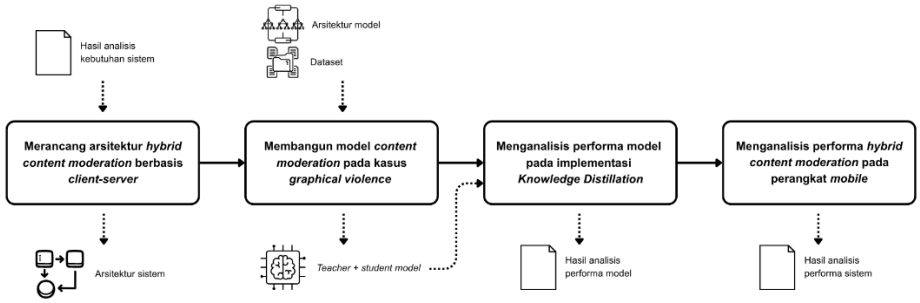
Perangkat Lunak	Deskripsi
Miniconda	Menyediakan lingkungan pengembangan sistem dan pengelolaan *library*
JupyterLab	Perangkat lunak untuk menjalankan kode penelitian, dan visualisasi.
Python	Bahasa pemrograman yang digunakan untuk mengembangkan model moderasi
PyTorch	*Framework deep learning* untuk membangun arsitektur model, training, dan data loader
CUDA	Library untuk akselerasi GPU
Torchvision	Memberikan bantuan *computer vision* (transformasi/augmentasi gambar, model, dan dataset)

TIMM	Kumpulan model SOTA untuk PyTorch; yang digunakan untuk membuat arsitektur
TorchMetrics	Menyediakan metrik evaluasi model terintegrasi
Scikit-learn	Menyediakan metrik performa model
NumPy	Memberikan bantuan komputasi numerik dan operasi *array*
Pandas	Memberikan bantuan manipulasi data tabular (*DataFrame*) untuk analisis/rekap hasil pelatihan model
Matplotlib	*Plotting* untuk visualisasi hasil *training*/evaluasi
Pillow	Pemroses gambar (*load*, *resize*, simpan)
Collections	Menyediakan struktur data untuk membantu penanganan data
Dataclasses	Menyediakan tipe data & anotasi untuk konfigurasi/struktur kode yang lebih rapi
Pathlib	Memberikan akses *filesystem* dan I/O sistem operasi
Time	Menyediakan bantuan penanganan waktu, fungsi matematis, dan deterministik

2.3 Prosedur Penelitian

Sebelum penelitian dilaksanakan, peneliti menentukan terlebih dahulu prosedur penelitian guna menciptakan lingkungan penelitian yang terstruktur. Peneliti merancang tahapan penelitian sebagai panduan teknis pelaksanaan penelitian, dan juga prasyarat pelaksanaan penelitian yang menunjukkan berbagai perangkat atau alat bantu yang digunakan selama pelaksanaan penelitian.

2.3.1 Tahapan Penelitian



Gambar 9. Diagram tahapan, masukan, dan keluaran penelitian

Gambar 9 menunjukkan diagram tahapan penelitian beserta masukan dan keluaran untuk setiap tahap. Selain menjadi pedoman bagi peneliti untuk

melaksanakan penelitian, diagram tersebut juga yang menentukan susunan penyusunan hasil penelitian yang akan dipaparkan pada bagian berikutnya.

2.3.2 Persiapan Perangkat Penelitian

Penelitian dilaksanakan setelah seluruh prasyarat penelitian terpenuhi untuk menghindari terhambatnya pelaksanaan penelitian akibat instrument penelitian yang tidak tersedia. Penelitian ini memprasyaratkan berbagai instrument yang telah disebutkan pada bagian sebelumnya. Instrumen-instrumen seperti perangkat keras dan perangkat lunak tersebut dipenuhi oleh peneliti sebelum dilaksanakannya penelitian ini dan telah dipastikan fungsionalitasnya sehingga instrument tersebut bekerja dengan baik.

2.4 Pelaksanaan Penelitian

2.4.1 Perancangan Arsitektur *Hybrid Content Moderation*

Pada tahap ini peneliti mengidentifikasi berbagai dependensi sistem moderasi konten. Peneliti menganalisis berbagai komponen penyusun sistem untuk dapat merancang arsitektur dengan kapabilitas yang baik pada skema yang telah direncanakan. Arsitektur yang dirancang mengadopsi arsitektur *client-server* seperti yang telah dibahas pada bagian sebelumnya untuk dapat menyesuaikan kondisi implementasi sistem pada perangkat *mobile*.

1. *Client Side*

   Penelitian ini menjadikan perangkat mobile khususnya telepon seluler cerdas (*smartphone*) berbasis Android pada sisi client dalam sistem. Perangkat lunak atau software dikembangkan untuk mereplikasi perangkat lunak yang nantinya akan mengimplementasikan sistem yang dikembangkan pada penelitian ini. Perangkat lunak yang dikembangkan peneliti nantinya akan bertanggung jawab untuk menjalankan model ringan (*student model*) dan mengeksekusi logika ambang batas (T_{low} dan T_{high}) untuk membuat keputusan lokal atau memicu eskalasi. Perangkat lunak ini juga akan mencatat metrik performa model pada sisi client sebagai data pendukung luaran pada penelitian ini.

2. *Server Side*

   Pada sisi server, peneliti membagi sistem menjadi dua bagian. Bagian pertama adalah *API service*, di mana ia hanya akan menangani *request* eskalasi yang dikirimkan dari *client* (yaitu, citra yang berada di 'zona abu-abu'). *Request* ini kemudian ditangani dan diolah untuk dimasukkan pada teacher model (model yang lebih besar dan akurat). Serupa dengan perangkat lunak yang dikembangkan pada sisi *client*, *service* yang dijalankan pada sisi server juga akan mencatat metrik performa klasifikasi teacher model untuk menilai seberapa baik model dalam mengklasifikasikan citra yang dikirimkan client dalam rentang confidence level tertentu pada model di sisi *client*.

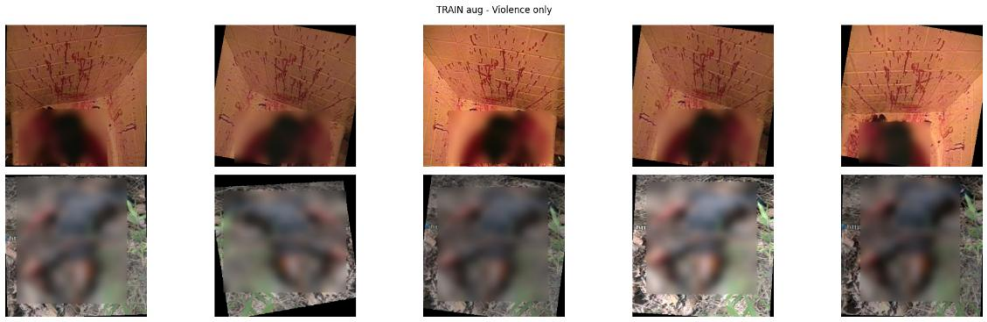
2.4.2 Pengembangan Model Moderasi Konten

Penelitian ini berfokus pada pengembangan model klasifikasi konten digital khususnya dalam format citra yang menunjukkan unsur kekerasan pada manusia. Luaran model klasifikasi yang dikembangkan menunjukkan apakah citra input aman (tanpa unsur kekerasan), atau mengandung unsur kekerasan seperti luka, darah, atau organ manusia yang terlihat dalam citra.

1. *Dataset* dan *Preprocessing*
   a) *Dataset*

      Dataset yang dianalisis untuk penelitian ini adalah "Graphical Violence and Safe Images Dataset", sebuah koleksi data yang tersedia untuk publik di platform Kaggle. Dataset ini dipublikasikan pada tahun 2023 oleh kreator Kaggle, Kartikey Bartwal. Dataset ini mengadopsi skema klasifikasi berbasis direktori (atau ImageFolder), di mana label kelas secara implisit ditentukan oleh nama folder tempat gambar tersebut berada. Strukturnya terdiri dari dua direktori utama:

      - Graphically Violent Images: Folder ini berisi semua gambar yang termasuk dalam kelas positif, yaitu yang menggambarkan kekerasan grafis.



Gambar 10. Sampel gambar pada direktori graphically safe images

      - Graphically Safe Images: Folder ini berisi semua gambar yang termasuk dalam kelas negatif, yaitu yang aman atau bebas dari kekerasan grafis.

Gambar 11. Sampel gambar pada direktori graphically safe images

Meskipun merupakan dataset yang relatif baru dan memiliki ukuran yang terbatas, "Graphical Violence and Safe Images Dataset" telah mendapatkan pengakuan dan validasi dari komunitas peneliti. Penggunaannya dalam literatur akademis dan teknis yang baru-baru ini diterbitkan memvalidasi relevansinya untuk penelitian tingkat tesis (Avinash dkk., 2025).

Dataset ini telah dikutip dan digunakan sebagai komponen dalam tumpukan data (*data stack*) yang lebih besar untuk melatih dan mengevaluasi sistem guardrail AI multimodal yang canggih. Contoh penting termasuk penggunaannya dalam penelitian "Protect: Towards Robust Guardrailing Stack for Trustworthy Enterprise LLM Systems". Dalam konteks ini, dataset ini berfungsi sebagai tolok ukur (benchmark) standar, meskipun kecil, untuk kategori spesifik "kekerasan grafis" dalam evaluasi keamanan model bahasa besar (LLM) (Avinash dkk., 2025).

Lebih lanjut, dataset ini telah dikutip dalam penelitian yang berfokus pada keamanan multimodal, AI yang dapat dijelaskan (Explainable AI atau XAI), dan dalam diskusi yang lebih luas tentang arsitektur sistem moderasi konten. Adopsi oleh para peneliti, termasuk yang berasal dari entitas seperti FutureAI, menunjukkan bahwa nilai dataset ini tidak terletak pada ukurannya, melainkan pada relevansi topikalnya yang tinggi dan fungsinya sebagai proksi yang diakui komunitas untuk domain masalah spesifik dari kekerasan grafis online. Penggunaannya dalam sebuah tesis, oleh karena itu, menempatkan penelitian tersebut dalam konteks akademis yang sudah ada dan relevan (Avinash dkk., 2025).

Secara kuantitatif, dataset ini berisi total 1168 gambar. Distribusi gambar di antara kedua kelas adalah sebagai berikut:

- Kelas Graphically Violent Images: Berisi 66 gambar. Total ukuran file untuk folder ini adalah 12 MB.
- Kelas Graphically Safe Images: Berisi 1102 gambar. Total ukuran file untuk folder ini adalah 200 MB.

Gambar-gambar dalam dataset disimpan dalam format file gambar standar, yang secara eksplisit dinyatakan di halaman Kaggle sebagai "JPEG atau PNG". Seperti halnya dataset yang dikurasi dari berbagai sumber, gambar-gambar di dalamnya memiliki resolusi, rasio aspek, dan profil warna yang bervariasi. Sifat data yang heterogen ini mengharuskan penerapan preprocessing pipeline (alur prapemrosesan) yang konsisten seperti pengubahan ukuran (*resizing*), pemotongan (*cropping*), dan normalisasi sebelum gambar-gambar tersebut dapat dimasukkan ke dalam model *deep learning*.

Selama masa penelitian, peneliti juga menambahkan beberapa gambar khususnya pada kelas “*violence*” yang didapatkan dari berbagai sumber seperti Google dan Telegram. Jumlah total keseluruhan data perkelas dan pembagiannya dijelaskan pada bagian berikutnya.

b) *Preprocessing*

Tahap pra-pemrosesan data pada penelitian ini dilakukan secara bertahap, mulai dari penyiapan struktur *dataset*, pembuangan duplikasi, pembagian data ke dalam subset pelatihan, validasi, dan uji, hingga transformasi dan augmentasi citra sebelum dimasukkan ke model pembelajaran mendalam. Seluruh proses diimplementasikan menggunakan Python dan pustaka PyTorch serta torchvision.

Dataset Graphical Violence and Safe Images disimpan di dalam folder data/raw dengan dua kelompok utama:

- Folder Graphically Safe Images: diberi label 0 (safe)
- Folder Graphically Violent Images: diberi label 1 (violence)

Peneliti membuat sebuah skrip tambahan yang dibuat terpisah dari skrip pengembangan model utama yang digunakan untuk:

- Membaca seluruh subdirektori di dalam data/raw.
- Mencocokkan nama folder dengan alias kata kunci seperti "Graphically Safe Images", "Graphically_Safe_Images", maupun nama yang mengandung kata "safe" untuk kelas aman, dan "Graphically Violent Images", "Graphically_Violent_Images", "violent", atau "violence" untuk kelas kekerasan.

Untuk mencegah kebocoran data antar-split (misalnya satu gambar yang sama muncul di train dan test), dilakukan deteksi dan penghapusan file duplikat secara global. Setiap file citra dihitung nilai hash MD5-nya, jika ada dua file yang memiliki hash sama, salah satunya dibuang, sehingga setiap citra unik hanya muncul satu kali di keseluruhan dataset.

Setelah citra dikumpulkan dan dibersihkan, seluruh data dibagi ke dalam tiga subset. Pembagian dilakukan secara stratified per kelas, artinya proporsi kelas safe dan violence dipertahankan pada setiap subset. Selanjutnya, setiap subset disimpan dalam format CSV.

Tabel 5. Sebaran data pelatihan

Jenis Data	Persentase	Jumlah Data
pelatihan (train)	70%	576 citra (487 aman, 89 kekerasan)
validasi (validation)	15%	123 citra (104 aman, 19 kekerasan)
pengujian (test)	15%	126 citra (106 aman, 20 kekerasan)

Pada tahap pelatihan, citra dibaca kembali dari file CSV, pada proses ini setiap baris CSV dibaca, lalu kemudian file citra dibuka dengan pustaka PIL, dikonversi ke RGB, dan kemudian dibuat menjadi rangkaian transformasi yang telah didefinisikan.

Transformasi dibagi menjadi dua jenis yaitu augmentasi (untuk *train*) dan transformasi deterministik (untuk val/test):

- Pra-pemrosesan dasar

Seluruh citra diubah ke ukuran tetap $img\_size \times img\_size$ piksel. Citra kemudian diubah menjadi tensor dan dinormalisasi menggunakan teknik z-score normalization dengan nilai mean [0.485,0.456,0.406] dan standard deviation [0.229,0.224,0.225] yang berasal dari dataset ImageNet. Secara teknis, proses ini dilakukan dengan mengurangkan nilai piksel pada setiap kanal warna (RGB) dengan nilai mean tersebut, kemudian membaginya dengan nilai standard deviation. Langkah normalisasi ini krusial untuk menyelaraskan distribusi statistik citra input dengan distribusi data yang digunakan saat model dilatih pertama kali (pre-training). Penyelarasan ini memastikan bahwa bobot (weights) pada model pre-trained dapat bekerja optimal dalam mengekstraksi fitur, mempercepat konvergensi saat pelatihan ulang (fine-tuning), serta mencegah terjadinya pergeseran kovariat (covariate shift) yang dapat menurunkan akurasi model.

- Augmentasi untuk data pelatihan

  Untuk meningkatkan kemampuan generalisasi model terhadap variasi tampilan citra, digunakan beberapa jenis augmentasi:
  - RandomResizedCrop dengan skala (0.85, 1.0)
  - RandomHorizontalFlip
  - ColorJitter (brightness dan contrast 0.15)

  Selain augmentasi standar, diterapkan beberapa augmentasi robust khusus untuk meniru kondisi dunia nyata:
  - RandomLowLight (faktor acak 0.6–0.9)
  - RandomJpegArtifacts (kualitas acak 40–70)
  - RandomPerspective (distortion_scale=0.2)
  - RandomRotation (±10 derajat)
- Transformasi untuk validasi dan pengujian

  Pada subset validasi dan pengujian, augmentasi acak tidak digunakan agar evaluasi lebih stabil dan merepresentasikan performa model pada data nyata. Transformasi yang digunakan hanya:
  - Resize((img_size, img_size))
  - ToTensor()
  - Normalize(mean, std)

Distribusi kelas pada dataset tidak seimbang (kelas kekerasan jauh lebih sedikit daripada kelas aman). Untuk mengurangi bias model terhadap kelas mayoritas, penelitian ini menggunakan *class weighting* pada fungsi loss yang dibuat dengan rumus

$$w_c = \frac{N_{total}}{K \times N_c} \tag{3}$$

dengan N_c adalah jumlah sampel kelas ke-c, dan K adalah jumlah kelas (2 kelas) (Bakirarar & Elhan, 2023).

2. *Teacher Model*

Model guru yang digunakan adalah EfficientNetV2-L. Pelatihan dilakukan dengan konfigurasi seperti yang tertera pada Tabel 6.

Tabel 6. Konfigurasi pelatihan teacher model

Hyperparapeter	**Nilai**
img_size	384
batch_size	8
epochs	30
use_class weighting	true
use_distillation	false

Untuk mengatasi ketidakseimbangan kelas, bobot kelas dihitung dari frekuensi data train dengan class counts [487, 89] dan bobot [0.591, 3.235].

Evaluasi post-training dilakukan di set uji dengan prosedur sebagai berikut:

a) Memuat model terbaik hasil pelatihan
b) Menghitung ROC–PR
c) Mencari ambang terbaik-F1 melalui grid (0.05–0.95)
d) Menampilkan *confusion matrix* pada thr=0.5 dan thr=best-F1.

3. *Student Model*

Model siswa (untuk inferensi sisi klien/*edge*) dilatih agar latensi rendah dengan akurasi setinggi mungkin melalui *Knowledge Distillation* (KD) dari *teacher*. Daftar kandidat yang dievaluasi mencakup arsitektur ringan beserta resolusi input realistis untuk perangkat:

Tabel 7. Daftar kandidat student model

Nama Kandidat Model	***Input Size***
tf_efficientnet_lite0.in1k	224
tf_efficientnet_lite2.in1k	260
tf_mobilenetv3_small_100.in1k	224
tf_mobilenetv3_large_100.in1k	224
mobilenetv3_large_100.ra_in1k	224

Pemilihan kandidat model student pada penelitian ini tidak dilakukan secara acak, melainkan sebagai keputusan strategis yang menentukan viabilitas arsitektur hybrid pada perangkat mobile. Lima kandidat dipilih berdasarkan analisis trade-off antara latensi dan akurasi, kompatibilitas terhadap karakteristik perangkat keras target, serta kemampuan arsitektur untuk menyerap pengetahuan dari model teacher melalui mekanisme Knowledge Distillation (KD). Tujuan pemilihan ini adalah memperoleh model sisi klien yang tetap responsif dan hemat sumber daya, namun memadai untuk menyaring konten secara andal sebelum eskalasi ke server.

Kelima kandidat dirancang sebagai eksperimen komparatif antara dua filosofi desain arsitektur ringan. Keluarga MobileNetV3 merepresentasikan pendekatan Neural Architecture Search (NAS) dan NetAdapt yang menargetkan minimisasi latensi pada perangkat mobile, sedangkan EfficientNet-Lite merepresentasikan pendekatan compound scaling yang menyeimbangkan kedalaman, lebar, dan resolusi jaringan secara proporsional. Dengan mempertentangkan kedua paradigma ini, penelitian dapat mengamati faktor arsitektural yang paling berkontribusi terhadap performa deteksi pada lingkungan edge.

MobileNetV3-Small dipilih untuk merepresentasikan batas bawah komputasi dan menguji seberapa kecil model dapat dibuat sebelum kehilangan kemampuan menangkap fitur kekerasan yang kompleks. MobileNetV3-Large dipilih sebagai opsi kapasitas yang lebih tinggi untuk menguji trade-off akurasi–latensi pada kelas perangkat yang sama. Selain itu, MobileNetV3-Large varian RandAugment dimasukkan untuk mengisolasi pengaruh "resep pelatihan/inisialisasi" dari faktor arsitektur, sehingga penelitian dapat menilai apakah fitur yang lebih robust dari pra-pelatihan/augmentasi berdampak pada kualitas distilasi dan performa akhir.

EfficientNet-Lite0 dipilih sebagai baseline dari keluarga Lite untuk pembandingan langsung terhadap MobileNetV3, sedangkan EfficientNet-Lite2 dipilih untuk menguji efek scaling (termasuk resolusi input yang lebih tinggi) terhadap akurasi dan latensi pada perangkat target. Varian "Lite" juga relevan untuk deployment karena menyederhanakan komponen tertentu yang berpotensi menjadi hambatan pada akselerator edge, sehingga memberi jalur kompatibilitas deployment yang lebih luas.

Pemilihan kandidat juga mempertimbangkan keselarasan struktural dengan model teacher (EfficientNetV2-L). Dalam KD, transfer pengetahuan cenderung lebih efektif ketika "bahasa fitur" teacher dan student relatif selaras. Karena kandidat yang dipilih sama-sama menggunakan variasi blok MBConv-like, diharapkan proses distilasi lebih stabil dibanding memilih arsitektur yang sangat berbeda yang berisiko menimbulkan kesenjangan semantik.

Class weighting tetap digunakan mengikuti distribusi data train. *Checkpoint* terbaik tiap eksperimen disimpan dengan *callback* best model.

a) Pencarian α

   Setiap kandidat dilatih 150 *epoch* tanpa *early stopping*, KD ON dengan *grid* α 0 - 1 dan suhu $T = 6$, menggunakan beberapa *seed* berbeda. Tujuan tahap ini adalah untuk mendapatkan α (bobot KD) yang konsisten unggul antarseed, dan mengamati konvergensi untuk tiap kandidat. Model yang disimpan hanyalah checkpoint terbaik berdasarkan metrik validasi utama ([AUC/F1]).

b) Pembandingan KD vs non-KD

   Untuk setiap kandidat, dilakukan pelatihan 400 epoch tanpa early stopping dengan beberapa *seed* dengan kondisi berikut:

- KD ON: menggunakan α dan T yang telah didapatkan pada bagian sebelumnya
- KD OFF (*baseline*): menggunakan loss supervised murni (tanpa komponen KL)

Setelah pelatihan, model terbaik diuji di set uji dengan menghitung ROC/PR, Precision/Recall/F1, serta dilakukan pencarian ambang terbaik-F1 (grid 0.05–0.95) dan ditampilkan confusion matrix pada thr=0.5 dan thr=best-F1. Latensi (ms/gambar) diukur pada perangkat *target/input size* kandidat.

2.4.3 Analisis Performa Model

Pada tahap ini, performa *teacher model* dan *student model* dianalisis untuk melihat seberapa baik model melakukan klasifikasi *violence* dan *safe* pada data uji. Evaluasi tidak hanya melihat akurasi prediksi, tetapi juga memperhatikan perilaku probabilitas keluaran model, terutama terkait pemilihan *threshold* (ambang keputusan). Karena keluaran model berbentuk probabilitas, threshold standar 0,5 digunakan sebagai pembanding, lalu dilakukan pencarian threshold terbaik (melalui grid search) untuk memperoleh performa yang lebih optimal pada metrik tertentu.

Pengukuran performa dilakukan menggunakan metrik klasifikasi seperti Precision, Recall, F1-Score, serta AUC (ROC). Selain itu, confusion matrix juga ditampilkan untuk memperlihatkan distribusi prediksi benar dan salah (TN, FP, FN, TP), baik pada threshold 0,5 maupun *threshold* terbaik. Kurva ROC dan PR digunakan untuk memberikan gambaran tambahan tentang trade-off performa model pada berbagai nilai *threshold*, sehingga analisis tidak hanya bergantung pada satu titik keputusan saja.

Khusus untuk student model, analisis dilakukan dengan membandingkan dua kondisi, yaitu tanpa knowledge distillation (KD OFF) dan dengan *knowledge distillation* (KD ON), agar terlihat kontribusi distillation terhadap kualitas prediksi. Karena student ditargetkan untuk perangkat *mobile*, maka selain metrik klasifikasi, juga dicatat latensi inferensi untuk menilai kelayakan implementasi di sisi klien. Jika eksperimen dijalankan dengan beberapa seed atau pengulangan, hasil akhirnya dirangkum dalam bentuk rata-rata dan variasinya agar kesimpulan tidak hanya bergantung pada satu kali pelatihan.

2.4.4 Analisis Performa Sistem

Analisis performa sistem dilakukan untuk menilai kinerja *pipeline end-to-end* pada arsitektur yang digunakan, yaitu kombinasi inferensi di *client* (*student*) dan verifikasi di server (*teacher*). Sistem tidak hanya dinilai dari akurasi model, tetapi dari bagaimana keputusan final dihasilkan dalam konteks penggunaan nyata mengenai kapan sistem cukup memutuskan di perangkat, dan kapan perlu melakukan eskalasi ke server untuk verifikasi agar hasil lebih meyakinkan.

Pengujian sistem dilakukan dengan mencatat waktu pada tiap komponen proses. Untuk sisi *client*, dicatat waktu inferensi student. Ketika terjadi eskalasi,

dicatat tambahan waktu yang mencakup proses pengiriman data, proses inferensi teacher di server, hingga respons kembali ke *client*. Dari sini dapat dihitung total waktu respons (*end-to-end*). Selain aspek waktu, sistem juga dianalisis dari sisi efisiensi melalui rasio eskalasi, yaitu seberapa sering prediksi perlu dikirim ke server. Rasio eskalasi ini penting karena berhubungan langsung dengan beban server dan pengalaman pengguna.

Selain pengukuran performa pada threshold standar, penelitian ini juga melakukan analisis sensitivitas threshold (*grid search*) pada rentang 0,1 hingga 0,9. Analisis ini bertujuan untuk mencari titik keseimbangan (*trade-off*) terbaik antara akurasi sistem dan rata-rata latensi. Threshold yang optimal dipilih berdasarkan kemampuan sistem meminimalkan *false negative* (lolosnya konten kekerasan) tanpa menyebabkan tingkat eskalasi ke server yang berlebihan.

Hasil analisis kemudian digunakan untuk menunjukkan trade-off utama sistem hybrid untuk mengetahui kasus yang jelas bisa diputuskan cepat di perangkat, sedangkan kasus yang “ragu-ragu” dapat dibuat lebih aman lewat verifikasi *teacher*. Dengan begitu, sistem diharapkan tetap responsif untuk penggunaan *mobile*, namun tetap menjaga kualitas keputusan pada sampel yang sulit.