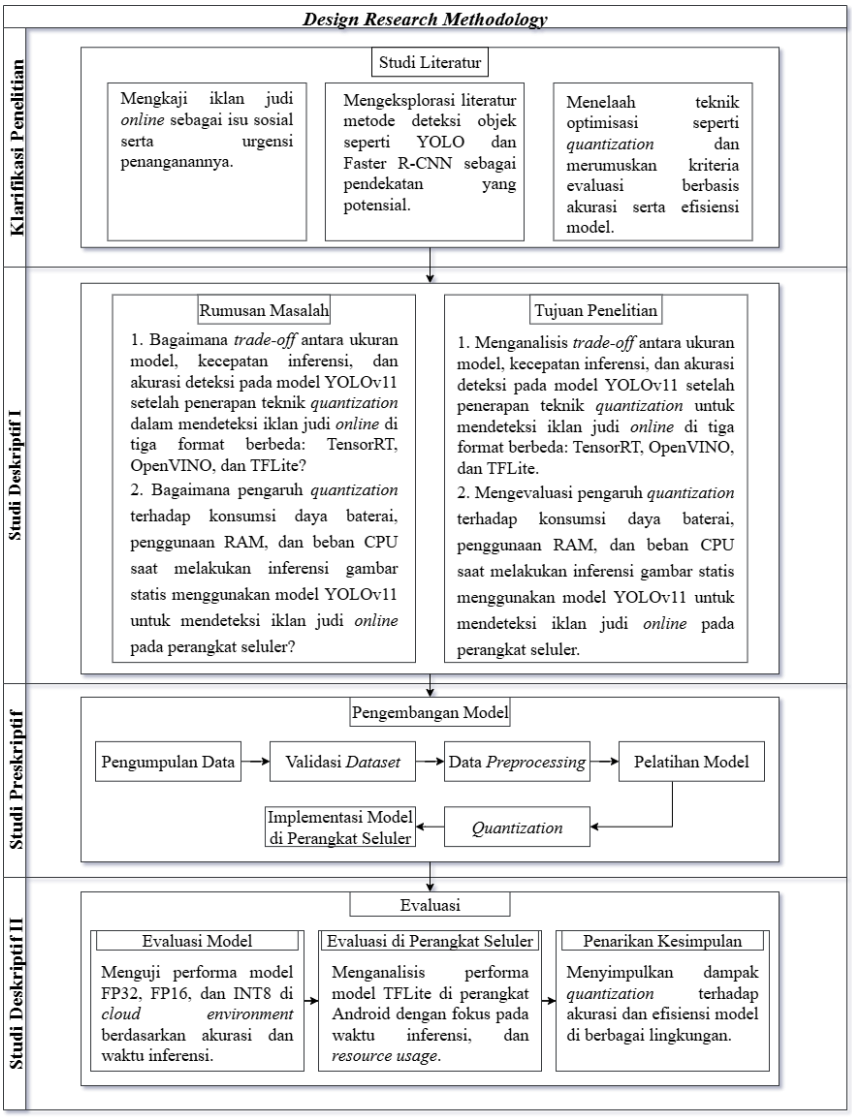
BAB III
METODE PENELITIAN

3.1 Desain Penelitian

Penelitian ini menggunakan *Design Research Methodology* (DRM), yang terdiri dari empat tahap, seperti terlihat pada Gambar 3.1. DRM dipilih karena pendekatannya yang sistematis dan terstruktur, mencakup studi deskriptif komprehensif dan pengembangan *support* (Blessing & Chakrabarti, 2009). Pendekatan ini mampu meningkatkan efektivitas penelitian desain serta menjaga keterkaitan antara pemahaman masalah, pengembangan solusi, dan evaluasi hasil.



Gambar 3.1 Desain Penelitian

Berdasarkan desain penelitian yang digambarkan, berikut penjelasan tahapan atau proses yang dilakukan:

3.1.1 Klarifikasi Penelitian

Langkah awal dalam pelaksanaan penelitian ini diawali dengan proses klarifikasi penelitian untuk memahami secara menyeluruh permasalahan utama yang diangkat, yaitu penyebaran iklan judi *online* di platform digital, khususnya media sosial. Permasalahan ini dipandang penting karena dapat berdampak negatif bagi masyarakat. Sebagai dasar untuk merumuskan masalah dan menentukan pendekatan solusi yang tepat, dilakukan kajian literatur secara mendalam. Referensi yang digunakan berasal dari berbagai sumber, seperti jurnal ilmiah, prosiding konferensi, artikel daring, serta dokumentasi teknis. Kajian ini difokuskan pada teknologi deteksi objek berbasis *deep learning* seperti YOLO (*You Only Look Once*) dan Faster R-CNN, serta teknik optimasi model seperti *quantization* yang mampu meningkatkan efisiensi pemrosesan tanpa mengurangi tingkat akurasi secara signifikan. Selain itu, proses klarifikasi ini juga mencakup pemahaman mengenai metode evaluasi performa model, seperti akurasi dan efisiensi komputasi, yang akan menjadi acuan dalam menilai keberhasilan pendekatan yang diterapkan. Melalui tahapan ini, diperoleh pemahaman awal yang menjadi dasar untuk merumuskan permasalahan dan tujuan penelitian secara lebih sistematis dan terarah.

3.1.2 Studi Deskriptif I

Tahap Studi Deskriptif I dilakukan untuk memperluas pemahaman mengenai permasalahan penyebaran iklan judi *online* di media sosial serta merumuskan tujuan penelitian secara lebih konkret. Hasil kajian literatur yang telah dilakukan pada tahap sebelumnya digunakan sebagai dasar untuk mendefinisikan inti permasalahan dan arah solusi yang akan dikembangkan. Dalam proses ini, peneliti menelaah lebih lanjut karakteristik permasalahan, termasuk tren, pola penyebaran, dan kompleksitas konten iklan judi *online* yang muncul di berbagai platform digital. Teknologi deteksi objek seperti YOLOv11 dan pendekatan optimasi model melalui

quantization mulai dipertimbangkan pada tahap ini sebagai alternatif solusi yang akan dieksplorasi lebih lanjut pada tahapan selanjutnya. Studi ini menjadi fondasi awal dalam merancang solusi berbasis *deep learning* sehingga analisis pada tahap ini berperan penting dalam memastikan pendekatan yang dipilih tetap relevan dengan tujuan dan ruang lingkup penelitian.

3.1.3 Studi Preskriptif

Studi preskriptif merupakan tahap inti dari penelitian ini yang mencakup seluruh proses teknis dalam pengembangan model deteksi iklan judi *online* berbasis YOLOv11. Tahapan ini terdiri dari beberapa subbagian, mulai dari pengumpulan data, validasi *dataset*, *preprocessing* data, pelatihan model YOLOv11, penerapan teknik *quantization*, hingga implementasi model pada perangkat seluler. Selain itu, model YOLOv11 sebelum optimisasi juga akan dievaluasi sebagai *baseline* untuk keperluan perbandingan yang akan dibahas lebih lanjut pada tahap berikutnya. Secara keseluruhan, tahapan ini merealisasikan pendekatan solusi yang telah dirumuskan sebelumnya melalui serangkaian eksperimen dan optimisasi model.

3.1.3.1 Pengumpulan Data

Tahap pengumpulan data dilakukan dengan memilih *dataset* yang relevan terhadap permasalahan penelitian, yaitu deteksi iklan judi *online* pada platform media sosial. *Dataset* yang digunakan adalah *dataset* Deteksi Judol yang dibuat oleh skripman (2025) di platform Roboflow Universe. *Dataset* ini terdiri atas 500 gambar yang berupa tangkapan layar dari postingan Instagram yang telah diberi anotasi dengan label “judi”. Jumlah ini dinilai memadai untuk pelatihan model YOLOv11 dengan satu kelas, sesuai dengan rekomendasi Ultralytics (2025) yang menyarankan setidaknya 100 gambar beranotasi per kelas untuk digunakan dalam pelatihan model. Pemilihan *dataset* ini relevan dengan latar belakang masalah mengingat Instagram merupakan platform media sosial dengan tingkat paparan iklan judi *online* tertinggi (Populix, 2024). Ciri utama dalam gambar-gambar tersebut adalah adanya logo judi *online* yang disisipkan pada postingan Instagram yang tampak seperti konten normal pada umumnya. Logo-logo ini digunakan

sebagai media promosi terselubung dan sering kali menyatu secara visual dengan elemen gambar lainnya. Karakteristik ini diperoleh melalui hasil observasi penulis terhadap postingan Instagram yang mengandung iklan judi *online* selama rentang waktu Februari hingga Mei 2025, termasuk yang terdapat dalam *dataset*. Selanjutnya, *dataset* ini akan melalui proses validasi untuk memastikan kelengkapan dan kesesuaian data sebelum memasuki tahap *preprocessing*.

3.1.3.2 Validasi *Dataset*

Validasi *dataset* dilakukan untuk memastikan bahwa data yang akan digunakan dalam pelatihan model memiliki kualitas yang baik, relevan, dan representatif. Proses ini dilaksanakan setelah *dataset* diperoleh dari Roboflow Universe dan sebelum memasuki tahap *preprocessing*. Validasi dilakukan secara menyeluruh melalui kombinasi pemeriksaan manual dan metode komputasional.

Pertama, dilakukan evaluasi manual terhadap isi gambar untuk memastikan bahwa seluruh gambar memang menampilkan logo yang berkaitan dengan judi *online*. Langkah ini penting agar model tidak dilatih dengan gambar yang tidak relevan, yang dapat menurunkan akurasi prediksi. Selanjutnya, dilakukan pengecekan terhadap kelengkapan anotasi dengan memastikan bahwa setiap gambar telah memiliki *bounding box* yang menandai objek target secara akurat. Evaluasi ketepatan posisi *bounding box* juga menjadi perhatian utama, karena kesalahan anotasi dapat mengganggu proses pembelajaran model.

Untuk mendeteksi potensi duplikasi gambar, digunakan metode *perceptual hashing*. Algoritma ini digunakan karena dapat menghasilkan *hash* yang stabil terhadap perubahan minor seperti kompresi, koreksi warna, atau penyesuaian kecerahan. Algoritma ini efektif untuk mengidentifikasi gambar yang identik atau sangat mirip, sehingga berguna untuk menyaring gambar redundan. Samanta & Jain (2021) menunjukkan bahwa *perceptual hashing* sangat akurat dalam mengenali salinan gambar dan modifikasi yang tidak mengubah konten secara signifikan. Namun, untuk memastikan keakuratan hasil deteksi, terutama pada kasus ambiguitas, peninjauan manual akan tetap dilakukan sebagai langkah verifikasi.

3.1.3.3 *Preprocessing* Data

Tahap *preprocessing* data dilakukan untuk menyiapkan *dataset* agar sesuai dengan kebutuhan model deteksi objek YOLOv11. Proses ini mencakup penyesuaian ukuran gambar, pembagian *dataset*, augmentasi data, normalisasi, serta konversi format *dataset* ke struktur yang sesuai dengan standar YOLOv11. Seluruh gambar terlebih dahulu diubah ukurannya menjadi 640 piksel, sesuai dengan resolusi *input* yang digunakan oleh model. Setelah itu, *dataset* dibagi menjadi tiga bagian, yaitu 70% untuk pelatihan, 20% untuk validasi, dan 10% untuk pengujian. Skema pembagian ini bertujuan untuk mempermudah pemantauan kinerja model selama proses pelatihan, sekaligus meningkatkan kemampuan generalisasi model terhadap data yang belum pernah ditemui sebelumnya (Ayturan dkk., 2025).

Untuk meningkatkan keragaman visual pada set pelatihan, digunakan teknik augmentasi seperti *horizontal flip*, rotasi 180°, serta rotasi acak dalam rentang −15° hingga +15°. Transformasi ini dipilih karena mampu memperluas variasi spasial tanpa mengubah konteks visual objek yang dianotasi, sehingga tetap relevan dengan tugas deteksi. Efektivitas pendekatan ini juga didukung oleh beberapa penelitian sebelumnya. Safdar dkk. (2020) menunjukkan bahwa rotasi 180° merupakan teknik augmentasi yang memberikan hasil terbaik pada model YOLOv3, sementara Zhou dkk. (2022) menemukan bahwa *horizontal flip* menjadi teknik augmentasi yang memberikan peningkatan performa terbesar. Selain itu, Nishio dkk. (2020) dan Nakamura dkk. (2020) menekankan efektivitas penerapan rotasi kecil, khususnya pada rentang −15° hingga +15°, yang terbukti efisien dalam memperkaya variasi data sekaligus membantu mencegah *overfitting*.

Setelah proses augmentasi, *dataset* diekspor ke dalam format yang kompatibel dengan YOLOv11. Setiap gambar dalam *dataset* dilengkapi dengan file label berekstensi ".txt" yang menyimpan informasi *bounding box* dalam format "*class x_center y_center width height*". Pada format ini, *class* merupakan indeks kelas objek, sedangkan *x_center*, *y_center*, *width*, dan *height* merepresentasikan koordinat pusat dan dimensi *bounding box* yang telah dinormalisasi ke rentang 0–1. Proses normalisasi dilakukan dengan membagi nilai absolut koordinat dan

ukuran *bounding box* terhadap lebar dan tinggi gambar sehingga representasi *bounding box* tidak bergantung pada resolusi gambar tertentu. Selain itu, file "data.yaml" juga disertakan untuk mendefinisikan informasi penting terkait *dataset*, seperti lokasi gambar untuk pelatihan, validasi, dan pengujian, jumlah kelas objek, serta nama kelas objek. Struktur direktori yang dihasilkan dari proses ekspor ini memudahkan integrasi langsung *dataset* dengan *pipeline* pelatihan model YOLOv11.

3.1.3.4 Pelatihan Model

Pelatihan model dilakukan menggunakan *framework* Ultralytics dengan dua varian arsitektur YOLOv11, yaitu *nano* (YOLOv11n) dan *small* (YOLOv11s). Pemilihan kedua varian ini didasarkan pada pertimbangan efisiensi untuk keperluan *deployment* pada berbagai lingkungan. Meskipun relatif ringan dari sisi kompleksitas komputasi, kedua varian tersebut tetap mampu memberikan performa yang kompetitif dalam tugas deteksi objek. Perbandingan arsitektural dari kedua model dapat dilihat pada Tabel 3.1 yang menampilkan jumlah *layer*, parameter, gradien, serta nilai GFLOPs berdasarkan ringkasan model.

Tabel 3.1 Perbandingan Arsitektur Model YOLOv11n dan YOLOv11s

Varian	Jumlah *Layer*	Jumlah Parameter	Jumlah Gradien	GFLOPs
YOLOv11n	181	2,624,080	2,624,064	6.6
YOLOv11s	181	9,458,752	9,458,736	21.7

Pelatihan model dilakukan dengan dua pendekatan berbeda untuk keperluan perbandingan. Pendekatan pertama adalah pelatihan dari awal (*training from scratch*) tanpa menggunakan bobot awal, sementara pendekatan kedua menggunakan *transfer learning* dengan teknik *fine-tuning* yang memanfaatkan bobot awal dari model yang telah dilatih pada dataset COCO. Pendekatan *transfer learning* bertujuan untuk mempercepat proses pelatihan serta meningkatkan akurasi model dengan memanfaatkan representasi fitur umum yang telah dipelajari sebelumnya.

Dataset hasil *preprocessing* kemudian digunakan dalam proses pelatihan model. Setiap gambar mengalami proses normalisasi nilai piksel dari rentang 0–255 menjadi 0–1 untuk menjaga stabilitas input dan mempercepat konvergensi model (J. Li dkk., 2018). Selama proses pelatihan, model dievaluasi secara berkala pada akhir setiap *epoch* menggunakan data validasi. Beberapa metrik evaluasi digunakan untuk memantau kinerja model dan proses konvergensi, di antaranya *mean Average Precision* (mAP), *precision*, *recall*, dan *F1-score*.

Konfigurasi *hyperparameter* pelatihan disusun berdasarkan acuan praktik terbaik dalam literatur dan penyesuaian terhadap kapasitas komputasi. Model dilatih hingga maksimum 150 *epoch* dengan mekanisme *early stopping* menggunakan *patience* sebesar 50 *epoch*, agar pelatihan dihentikan secara otomatis apabila tidak terjadi peningkatan performa signifikan. *Optimizer* yang digunakan adalah *Stochastic Gradient Descent* (SGD) dengan *learning rate* sebesar 0.0001 dan momentum 0.999, sesuai dengan konfigurasi terbaik dalam temuan Moraes dkk. (2025). Ukuran *batch* ditetapkan sebanyak 16, mengikuti konfigurasi *default* dari Ultralytics dan juga digunakan oleh Moraes dkk. (2025). Selain itu, *automatic mixed precision* (AMP) tidak diaktifkan agar seluruh proses pelatihan berjalan dalam presisi penuh (FP32), sehingga model yang dihasilkan dapat dijadikan *baseline* representatif sebelum dilakukan proses *quantization* pada tahap selanjutnya. Rincian lengkap konfigurasi *hyperparameter* yang digunakan disajikan pada Tabel 3.2.

Tabel 3.2 Konfigurasi Hyperparameter Pelatihan Model

Hyperparameter	**Nilai**
Epoch	150
Patience	50
Image Size	640×640 piksel
Learning Rate	0.0001
Momentum	0.999
Automatic Mixed Precision	*False*
Batch Size	16
Optimizer	SGD

3.1.3.5 *Quantization*

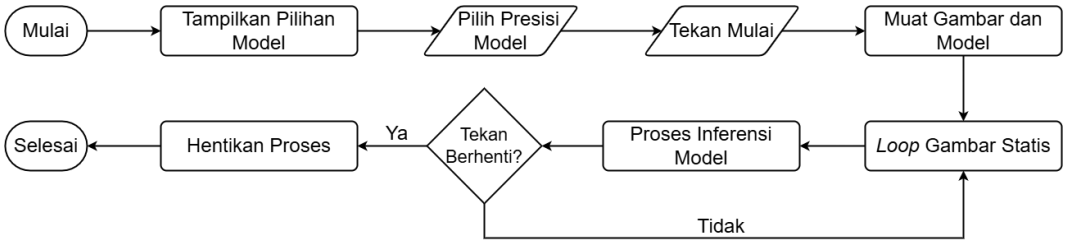
Setelah memperoleh model *baseline* dari tahap sebelumnya, langkah selanjutnya adalah melakukan proses konversi model ke dalam format yang sesuai dengan target *runtime* dan perangkat tujuan. Model hasil pelatihan, yang awalnya berada dalam format bawaan Ultralytics, terlebih dahulu dikonversi ke format ONNX sebagai format perantara. Konversi ke ONNX ini bertujuan untuk menjembatani kompatibilitas dengan berbagai *runtime* dan alat optimasi yang digunakan dalam penelitian ini. Model ONNX kemudian dikonversi ke tiga format akhir yang berbeda, yaitu TensorRT untuk pengujian pada *runtime* GPU, OpenVINO untuk pengujian pada *runtime* CPU, dan TensorFlow Lite (TFLite) untuk implementasi dan pengujian pada perangkat Android.

Konversi model dilakukan sebagai bagian dari proses optimasi melalui teknik *post-training quantization* (PTQ), dengan tujuan utama untuk mengurangi ukuran model dan mempercepat waktu inferensi tanpa melakukan pelatihan ulang. Dalam proses ini, model dikonversi ke tiga representasi numerik, yaitu FP32 sebagai acuan *full precision*, FP16 untuk *half precision*, dan INT8 sebagai representasi 8-bit yang paling efisien. Representasi FP32 digunakan sebagai *baseline* untuk perbandingan performa dan ukuran model sebelum dan sesudah optimasi. Proses kalibrasi untuk *quantization* ke INT8 dilakukan menggunakan data validasi, guna memastikan bahwa konversi tidak menurunkan performa model secara signifikan.

Meskipun PTQ merupakan pendekatan utama, apabila setelah proses *quantization* terjadi penurunan akurasi yang signifikan, maka akan diterapkan teknik *quantization-aware training* (QAT) sebagai alternatif. QAT memungkinkan model untuk belajar menyesuaikan bobotnya dengan mempertimbangkan efek dari *quantization* selama pelatihan, sehingga diharapkan dapat mempertahankan akurasi model mendekati performa sebelum *quantization*. Dengan demikian, model yang telah melalui proses ini akan siap digunakan pada berbagai platform dan perangkat, sesuai dengan kebutuhan implementasi di dunia nyata.

3.1.3.6 Implementasi Model di Perangkat Seluler

Tahap implementasi dilakukan untuk membangun aplikasi seluler sebagai sarana pengujian performa model deteksi hasil *quantization* pada perangkat Android. Model yang digunakan dalam tahap ini merupakan hasil konversi dari model *baseline* ke dalam format TensorFlow Lite (TFLite) yang telah dilakukan pada tahap sebelumnya. Aplikasi Android dikembangkan menggunakan bahasa pemrograman Java dan memanfaatkan TensorFlow Lite Java Interpreter untuk menjalankan proses inferensi secara lokal di perangkat seluler. Antarmuka aplikasi dirancang secara minimalis dan fungsional, dengan fokus pada fitur utama berupa pengujian terhadap gambar-gambar uji statis. Gambar yang digunakan dalam pengujian diambil langsung dari *dataset* pengujian yang telah disiapkan sebelumnya dan ditempatkan di direktori internal aplikasi. Alur kerja aplikasi, mulai dari pemuatan model hingga inferensi dihentikan, dijabarkan secara sistematis pada Gambar 3.2.

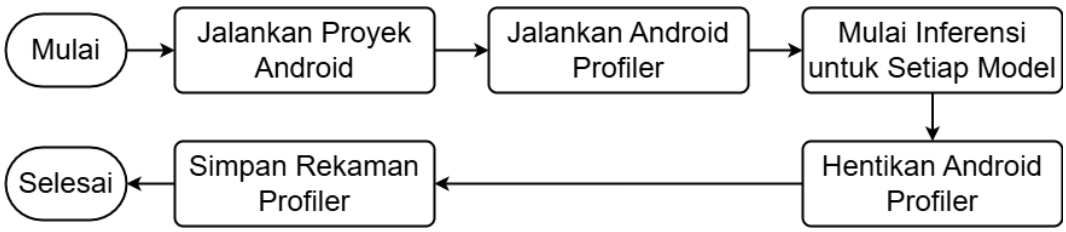


Gambar 3.2 Diagram Alur Implementasi Aplikasi

Untuk mengevaluasi dampak dari proses *quantization* secara objektif, dilakukan pengujian performa terhadap ketiga varian model TFLite (FP32, FP16, dan INT8). Pengujian ini dirancang untuk mengukur efisiensi setiap model dalam konteks penggunaan sumber daya perangkat. Parameter yang diukur secara spesifik mencakup penggunaan memori (RAM), beban CPU, konsumsi daya baterai, serta waktu inferensi dari masing-masing model saat memproses gambar dari *dataset* uji.

Untuk memperoleh data pengukuran yang presisi dan andal, proses pengujian performa dilakukan menggunakan Android Profiler. Alat ini menyediakan visualisasi data historis untuk metrik-metrik yang telah disebutkan. Langkah-langkah teknis dalam menyiapkan lingkungan pengujian, menjalankan

aplikasi, dan merekam data performa menggunakan Android Profiler dirangkum dalam Gambar 3.3.



Gambar 3.3 Diagram Alur Prosedur Pengujian Performansi

3.1.4 Studi Deskriptif II

Tahap ini bertujuan untuk mengevaluasi performa model deteksi yang telah melalui proses *quantization* dan konversi ke dalam berbagai representasi numerik, yaitu FP32, FP16, dan INT8. Model FP32 digunakan sebagai *baseline* karena mewakili kondisi sebelum dilakukan optimasi. Sementara itu, model FP16 dan INT8 merupakan hasil dari teknik *quantization* yang diterapkan dengan tujuan mengurangi ukuran model serta mempercepat waktu inferensi. Evaluasi dilakukan pada dua lingkungan berbeda yaitu *cloud environment* dan perangkat seluler untuk memperoleh gambaran menyeluruh tentang efisiensi model dari segi akurasi dan performa komputasi.

Evaluasi pertama dilakukan di *cloud environment* menggunakan dua jenis *runtime*, yaitu TensorRT untuk GPU dan OpenVINO untuk CPU. Model *baseline* FP32 serta model hasil *quantization* (FP16 dan INT8) dijalankan pada masing-masing *runtime* untuk memperoleh metrik evaluasi yang mencakup *mean Average Precision* (mAP), *precision*, *recall*, *F1-score*, serta waktu inferensi. Seluruh evaluasi dilakukan menggunakan *dataset* pengujian yang sama agar hasil perbandingan bersifat adil dan konsisten. Pengukuran waktu inferensi dilakukan dengan mencatat rata-rata waktu yang dibutuhkan model untuk memproses satu gambar, tanpa mempertimbangkan proses *preprocessing* atau *postprocessing* di luar arsitektur model.

Setelah pengujian di *cloud environment*, evaluasi dilanjutkan pada perangkat seluler Android untuk menguji kinerja model dalam kondisi nyata yang terbatas dari

sisi sumber daya. Model yang digunakan dalam tahap ini merupakan model dalam format TensorFlow Lite (TFLite) yang telah dikonversi dari model *baseline*. Sebelum diimplementasikan pada aplikasi Android, model TFLite ini juga dievaluasi akurasinya di *cloud environment* untuk memastikan kualitas deteksinya tetap terjaga setelah proses *quantization* dan konversi format.

Pada tahap tersebut, proses pengujian tidak hanya memperhatikan waktu inferensi, tetapi juga mencakup aspek efisiensi penggunaan sumber daya perangkat. Oleh karena itu, dilakukan pemantauan terhadap penggunaan memori (RAM), beban kerja CPU, dan konsumsi daya baterai selama proses inferensi berlangsung. Pemantauan dilakukan menggunakan fitur Android Profiler yang tersedia di Android Studio untuk memperoleh data performa selama aplikasi dijalankan.

3.2 Instrumen Penelitian

Instrumen penelitian merupakan alat bantu yang digunakan untuk mengumpulkan, mengukur, dan menganalisis data dalam proses penelitian. Dalam penelitian ini, instrumen yang digunakan bertujuan untuk mengevaluasi performa model deteksi iklan judi *online* baik dari segi akurasi maupun efisiensi komputasi. Instrumen tersebut terbagi menjadi dua kategori, yaitu instrumen evaluasi akurasi deteksi objek dan instrumen evaluasi efisiensi komputasi.

3.2.1 Instrumen Evaluasi Akurasi

Instrumen pertama digunakan untuk mengukur tingkat akurasi model dalam mendeteksi iklan judi *online*. Evaluasi dilakukan dengan pendekatan berbasis metrik klasifikasi dan deteksi objek. Struktur dasar evaluasi berbasis klasifikasi ditunjukkan melalui *confusion matrix* pada Tabel 3.2.

Tabel 3.3 *Confusion Matrix*

	Predicted Positive	*Predicted Negative*
Actual Positive	TP (*True Positive*)	FN (*False Negative*)
Actual Negative	FP (*False Positive*)	TN (*True Negative*)

Confusion matrix merupakan tabel yang menggambarkan performa model klasifikasi dengan membandingkan antara hasil prediksi model dan label sebenarnya. Tabel ini terdiri dari empat komponen utama yang menjadi dasar perhitungan berbagai metrik evaluasi akurasi (Géron, 2019), yaitu*:*

1. TP (*True Positive*): Prediksi positif yang benar, yaitu ketika model memprediksi positif dan kondisi sebenarnya juga positif.
2. TN (*True Negative*): Prediksi negatif yang benar, yaitu ketika model memprediksi negatif dan kondisi sebenarnya juga negatif.
3. FP (*False Positive*): Prediksi positif yang salah, yaitu ketika model memprediksi positif tetapi kondisi sebenarnya negatif.
4. FN (*False Negative*): Prediksi negatif yang salah, yaitu ketika model memprediksi negatif tetapi kondisi sebenarnya positif.

Berdasarkan komponen-komponen tersebut, berikut adalah metrik evaluasi akurasi yang digunakan dalam penelitian ini.

1. *Precision*

   Precision mengukur proporsi prediksi positif yang benar dari seluruh prediksi positif yang dihasilkan oleh model. Metrik ini dihitung dengan rumus:

$$Precision = \frac{TP}{TP + FP} \quad (1)$$

2. *Recall*

   Recall mengukur proporsi objek sebenarnya yang berhasil terdeteksi oleh model dari seluruh objek yang ada. Metrik ini dihitung dengan rumus:

$$Recall = \frac{TP}{TP + FN} \quad (2)$$

3. *F1-score*

F1-Score merupakan rata-rata harmonik dari *precision* dan *recall*, yang mencerminkan keseimbangan antara keduanya. Metrik ini dihitung dengan rumus:

$$F1 - score = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad (3)$$

4. *mean Average Precision* (mAP)

   mAP merupakan metrik utama dalam evaluasi deteksi objek. mAP adalah rata-rata dari nilai *Average Precision* (AP) untuk setiap kelas. Metrik ini dihitung dengan rumus:

$$mAP = \frac{1}{\mathrm{N}} \sum_{\mathrm{i}=1}^{\mathrm{N}} \mathrm{AP_i} \quad (4)$$

   Keterangan:

   - N : Jumlah total kelas
   - AP_i : *Average Precision* untuk kelas ke-i

5. *Average Precision* (AP)

   AP meringkas bentuk kurva *Precision-Recall* dengan menghitung luas area di bawahnya. Nilai ini merepresentasikan performa model pada satu kelas spesifik. Untuk menentukan sebuah deteksi adalah *True Positive* (TP) atau *False Positive* (FP), digunakan ambang batas *Intersection over Union* (IoU). Sebagai contoh, pada standar AP@0.5, sebuah deteksi dianggap TP jika nilai IoU-nya dengan *ground truth* lebih dari 0.5. AP dihitung dengan rumus:

$$AP = \int_0^1 P_{interp}(r)dr \quad (5)$$

   Keterangan:

- $P_{interp}(r)$: *Precision* hasil interpolasi linier pada titik *recall* r.
- Integral dihitung secara numerik menggunakan metode *trapezoidal* pada 101 titik *recall* antara 0 dan 1.

6. *Intersection over Union* (IoU)

   IoU digunakan untuk mengukur tingkat tumpang tindih antara *bounding box* hasil prediksi dengan *bounding box* sebenarnya (*ground truth*). Nilainya dihitung dengan rumus:

$$IoU = \frac{Luas\ dari\ Area\ Irisan\ (Intersection)}{Luar\ dari\ Area\ Gabungan\ (Union)} \quad (6)$$

3.2.2 Instrumen Evaluasi Efisiensi Komputasi

Selain akurasi, penelitian ini juga berfokus pada efisiensi model. Evaluasi efisiensi dilakukan untuk menilai kelayakan model digunakan secara *real-time*. Instrumen evaluasi efisiensi meliputi:

1. Waktu Inferensi

   Waktu inferensi adalah rata-rata waktu yang dibutuhkan model untuk melakukan deteksi objek pada satu gambar. Semakin rendah waktu inferensi, semakin cepat model dalam memberikan hasil prediksi. Dalam penelitian ini, waktu inferensi dihitung dengan rumus:

$$\bar{t} = \frac{1}{M}\sum_{j=1}^{M}\left(\frac{1}{N}\sum_{\mathrm{i}=1}^{N}\left(t_{1,ij} - t_{0,ij}\right)\right) \quad (7)$$

   Keterangan:

   - $\bar{t}$: Waktu inferensi rata-rata keseluruhan
   - M : Jumlah eksperimen atau pengujian berulang
   - N : Jumlah inferensi pada setiap eksperimen
   - $t_{0,ij}$: Waktu mulai inferensi ke-i pada eksperimen ke-j

- $t_{1,ij}$: Waktu selesai inferensi ke-i pada eksperimen ke-j

2. Penggunaan RAM

   Penggunaan RAM menunjukkan jumlah memori aktif yang digunakan aplikasi selama proses inferensi. Rata-rata penggunaan RAM dihitung berdasarkan data yang diambil setiap detik selama durasi pengukuran dengan rumus:

$$\bar{R} = \frac{1}{N} \sum_{i=1}^{N} R_i \quad (8)$$

   Keterangan:

   - $\bar{R}$: Rata-rata penggunaan RAM
   - R_i : Penggunaan RAM pada detik ke-i
   - N : Total durasi inferensi dalam detik

3. Beban CPU

   Beban CPU menunjukkan persentase penggunaan CPU oleh aplikasi selama proses inferensi. Rata-rata beban CPU dihitung per detik selama durasi pengukuran menggunakan rumus berikut:

$$\bar{C} = \frac{1}{N} \sum_{i=1}^{N} C_i \quad (9)$$

   Keterangan:

   - $\bar{C}$: Beban CPU rata-rata
   - C_i : Beban CPU pada detik ke-i
   - N : Total durasi pengukuran dalam detik

4. Konsumsi Daya Baterai

Konsumsi daya menunjukkan daya yang dikonsumsi perangkat selama proses inferensi berlangsung. Pengukuran dilakukan menggunakan Android Profiler pada Android Studio dengan mengambil nilai arus instan (*Current*) yang kemudian dikonversi menjadi daya dalam satuan J/s menggunakan rumus:

$$P = I \times V \tag{10}$$

Keterangan:

- P : Konsumsi daya (J/s atau Watt)
- I : Arus instan (A)
- V : Tegangan operasional perangkat (V)

3.3 Alat dan Bahan Penelitian

Penelitian ini menggunakan berbagai perangkat keras dan perangkat lunak untuk menunjang seluruh proses pelatihan, optimasi, implementasi, dan evaluasi model deteksi iklan judi *online*. Spesifikasi lengkap perangkat yang digunakan selama proses penelitian ditampilkan pada Tabel 3.3.

Tabel 3.4 Spesifikasi Perangkat

Kategori	Komponen	Spesifikasi
Laptop	Prosesor	Intel Core i3-1115G4
	RAM	8 GB
	Penyimpanan	SSD NVMe 256 GB
	GPU	Intel UHD Graphics
	Sistem Operasi	Windows 11
Cloud Environment	Platform	Google Colab
	Prosesor	Intel Xeon
	GPU	NVIDIA Tesla T4
	RAM	13 GB
Perangkat Seluler	Prosesor	Snapdragon 778G

	Jumlah Inti CPU	8-core: 1x Kryo 670 Prime (Cortex-A78) @2.4 GHz, 3x Kryo 670 Gold (Cortex-A78) @2.4 GHz, 4x Kryo 670 Silver (Cortex-A55) @1.8 GHz
	RAM	8 GB
	Penyimpanan	256 GB
	GPU	Adreno 642L
	Sistem Operasi	Android 14

3.4 Analisis Data

Tahap analisis data dilakukan untuk menilai performa model deteksi iklan judi *online* yang telah melalui proses *quantization*. Analisis ini mencakup dua konteks lingkungan pengujian yang berbeda, yaitu *cloud environment* dan perangkat seluler. Pada pengujian di *cloud environment*, model dievaluasi dalam dua konfigurasi *runtime*, yaitu TensorRT untuk GPU dan OpenVINO untuk CPU. Pengujian dilakukan terhadap model *baseline* serta model hasil *quantization*, dengan menggunakan *dataset* pengujian yang sama. Evaluasi dilakukan berdasarkan metrik yang mencerminkan kinerja deteksi seperti *mean Average Precision* (mAP), *precision*, *recall*, *F1-score*, dan waktu inferensi. Selain itu, model dalam format TFLite juga diuji di cloud environment, namun hanya pada metrik akurasinya saja.

Selanjutnya, analisis juga dilakukan pada perangkat seluler terhadap model yang telah diimplementasikan ke dalam aplikasi Android. Evaluasi pada tahap ini tidak hanya mencakup waktu inferensi, tetapi juga penggunaan sumber daya perangkat seperti RAM, CPU, dan konsumsi daya baterai. Data pengujian diperoleh dari hasil pemantauan performa aplikasi menggunakan Android Profiler. Hasil analisis dari kedua skenario ini akan menjadi dasar dalam menarik kesimpulan mengenai efektivitas pendekatan *quantization* dalam menunjang penerapan model deteksi iklan judi *online* pada kondisi dan kebutuhan penggunaan yang berbeda.