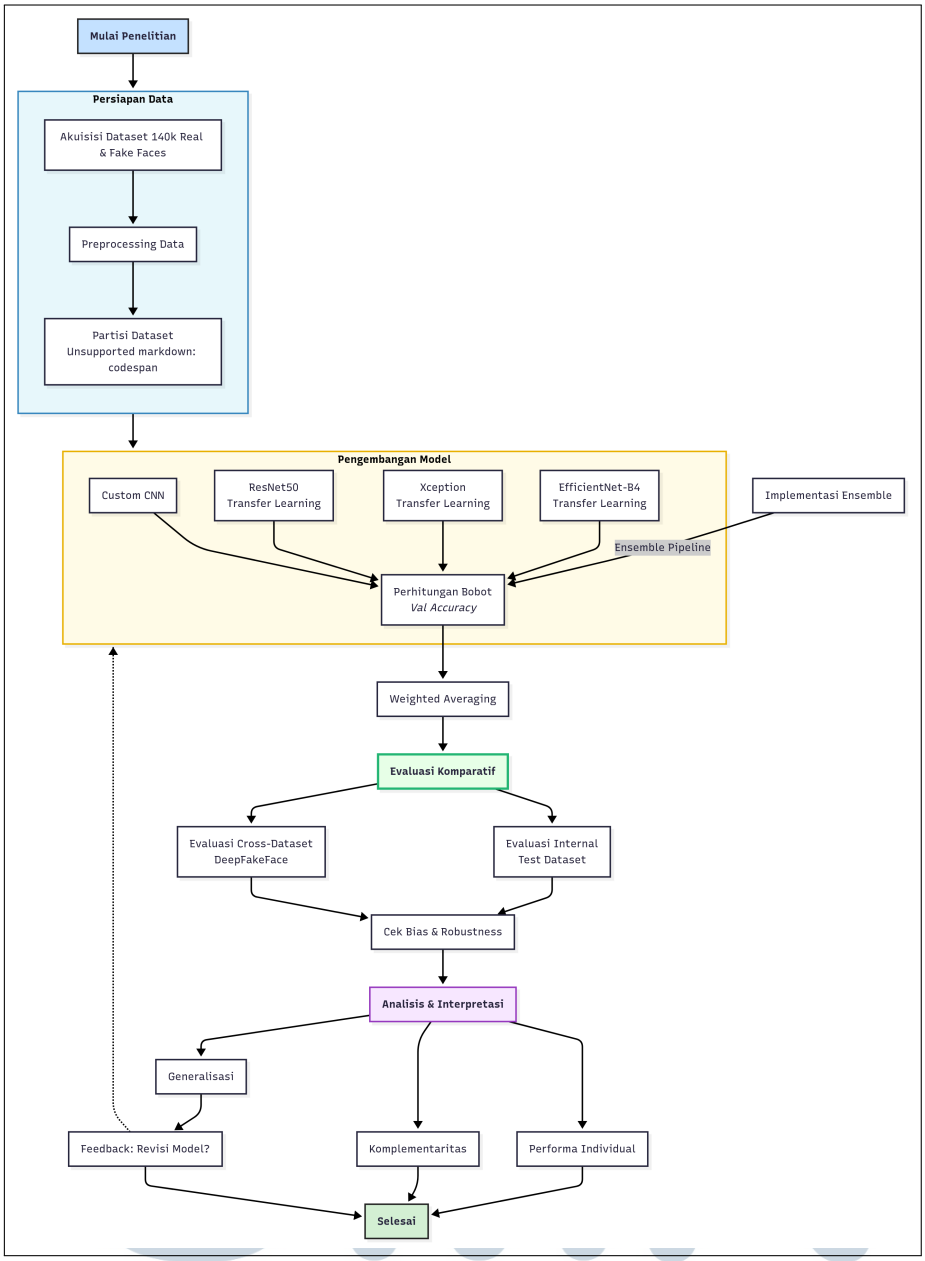
BAB 3
Metodologi Penelitian

3.1 Desain Penelitian

Penelitian ini mengadopsi pendekatan eksperimental kuantitatif dengan metodologi *comparative analysis* untuk mengevaluasi efektivitas *ensemble weighted averaging* dalam deteksi *deepfake*. Kerangka penelitian dirancang untuk membandingkan secara sistematis performa model *deep learning* individual dengan pendekatan *ensemble*, menggunakan metrik evaluasi komprehensif pada *dataset* terstandarisasi.

3.1.1 Alur Penelitian

Untuk memberikan gambaran menyeluruh tentang metodologi yang diterapkan, Gambar 3.1 menyajikan kerangka kerja penelitian secara konseptual, mulai dari persiapan data hingga analisis interpretasi.



Gambar 3.1. Pipeline penelitian deepfake

Seperti yang terlihat pada Gambar 3.1, metodologi penelitian mengikuti alur sistematis yang dimulai dengan persiapan data, dilanjutkan dengan pengembangan model individual dalam empat arsitektur berbeda, penghitungan bobot ensemble berdasarkan validation accuracy, implementasi weighted averaging, dan diakhiri dengan evaluasi komprehensif melalui pengujian internal maupun cross-dataset untuk analisis komplementaritas dan performa individual.

3.1.2 Kerangka Penelitian

Untuk mencapai tujuan tersebut, metodologi penelitian disusun dalam lima tahapan utama yang berurutan, sebagaimana diuraikan di bawah ini:

1. **Persiapan Data**: Meliputi akuisisi, *preprocessing*, dan partisi *dataset* untuk memastikan data siap digunakan dalam pelatihan dan evaluasi.
2. **Pengembangan Model Individual**: Merancang, melatih, dan mengevaluasi empat arsitektur *deep learning* yang berbeda untuk menciptakan keragaman dalam *ensemble*.
3. **Implementasi *Ensemble***: Mengembangkan model *ensemble weighted averaging* dengan menghitung bobot berdasarkan performa validasi dari setiap model individual.
4. **Evaluasi Komparatif**: Melakukan perbandingan sistematis antara performa model individual dengan model *ensemble* menggunakan *dataset* pengujian.
5. **Analisis dan Interpretasi**: Menganalisis hasil secara statistik untuk menarik kesimpulan mengenai efektivitas pendekatan *ensemble* dan implikasi praktisnya.

3.1.3 Desain Eksperimen

Penelitian ini menggunakan desain eksperimen terkendali untuk mengukur pengaruh arsitektur model terhadap performa deteksi. Variabel-variabel dalam penelitian ini didefinisikan sebagai berikut:

- **Variabel Bebas**: Arsitektur model yang digunakan, yaitu *CNN*, *ResNet50*, *Xception*, dan *EfficientNet-B4*.
- **Variabel Terikat**: Metrik performa yang diukur, meliputi *Accuracy*, *Precision*, *Recall*, *F1-Score*.
- **Variabel Terkendali**: Faktor-faktor yang dijaga konstan untuk memastikan validitas perbandingan, seperti *dataset*, pipa *preprocessing*, parameter pelatihan utama, dan metodologi evaluasi.

Validitas internal dijaga melalui prosedur standarisasi yang ketat pada seluruh tahapan eksperimen. Sementara itu, validitas eksternal ditingkatkan

dengan menggunakan *dataset* publik yang representatif dan metodologi yang dapat direplikasi.

3.2 Dataset dan Preprocessing

Pemilihan dan persiapan *dataset* merupakan fondasi krusial dalam penelitian ini. Tahapan ini bertujuan untuk menyediakan data yang berkualitas, seimbang, dan relevan untuk melatih serta mengevaluasi model deteksi *deepfake*.

3.2.1 Karakteristik Dataset

Penelitian ini menggunakan *dataset* "140k Real and Fake Faces" yang diperoleh dari repositori *Kaggle*. *Dataset* ini dipilih karena relevansinya dengan tugas deteksi wajah asli versus palsu dan skalanya yang besar. Karakteristik utama dari *dataset* ini dirangkum dalam Tabel 3.1.

Tabel 3.1. Karakteristik data "140k Real and Fake Faces"

Karakteristik	**Spesifikasi**
Total Gambar	140.000
Gambar Asli (*Real*)	70.000 (50%)
Gambar Palsu (*Fake*)	70.000 (50%)
Format Gambar	JPEG
Resolusi	256×256 piksel
Color Space	RGB
Sumber Gambar Asli	*Flickr* (Koleksi *Nvidia*)
Sumber Gambar Palsu	*StyleGAN Generated*

3.2.2 Partisi Dataset

Untuk memastikan evaluasi yang objektif, *dataset* dibagi menjadi tiga subset: pelatihan, validasi, dan pengujian. Pembagian ini menggunakan teknik *stratified sampling* untuk menjaga distribusi kelas yang seimbang pada setiap subset. Proporsi pembagiannya adalah sebagai berikut:

$$D_{train} = 100.000 \text{ gambar } (71.4\%) \tag{3.1}$$

$$D_{val} = 20.000 \text{ gambar } (14.3\%) \tag{3.2}$$

$$D_{test} = 20.000 \text{ gambar } (14.3\%) \tag{3.3}$$

3.2.3 Pipa Preprocessing

Pipa *preprocessing* diimplementasikan menggunakan generator data dari framework deep learning. Proses ini mencakup normalisasi nilai piksel dan augmentasi data. Konfigurasi yang berbeda diterapkan untuk data pelatihan dan data evaluasi (validasi dan pengujian).

A Preprocessing Data Pelatihan

Generator data pelatihan dikonfigurasi dengan berbagai parameter untuk mengoptimalkan proses pembelajaran. Pertama, dilakukan normalisasi piksel dengan membagi setiap nilai piksel dengan 255 untuk mengubah skala dari rentang 0-255 menjadi 0-1, yang membantu stabilitas proses pelatihan. Augmentasi data berupa *horizontal flip* diterapkan secara acak pada gambar untuk meningkatkan variasi data dan mencegah overfitting. Ukuran target gambar ditetapkan pada 256×256 piksel untuk konsistensi dengan arsitektur model. Batch size diatur pada 32 untuk keseimbangan antara efisiensi komputasi dan penggunaan memori. Mode kelas ditetapkan sebagai binary karena ini adalah masalah klasifikasi dua kelas (real vs fake). Terakhir, shuffle diaktifkan untuk memastikan urutan data acak pada setiap epoch, yang membantu proses konvergensi model.

B Preprocessing Data Validasi dan Pengujian

Untuk data validasi dan pengujian, konfigurasi generator dibuat lebih sederhana dan deterministik. Normalisasi piksel tetap diterapkan dengan cara yang sama seperti data pelatihan untuk konsistensi. Namun, tidak ada augmentasi data yang diterapkan untuk memastikan evaluasi dilakukan pada data orisinal yang tidak dimodifikasi. Ukuran target gambar tetap 256×256 piksel untuk konsistensi. Untuk data validasi, batch size diatur pada 16 untuk efisiensi proses evaluasi, sedangkan untuk data pengujian menggunakan batch size 1 untuk prediksi

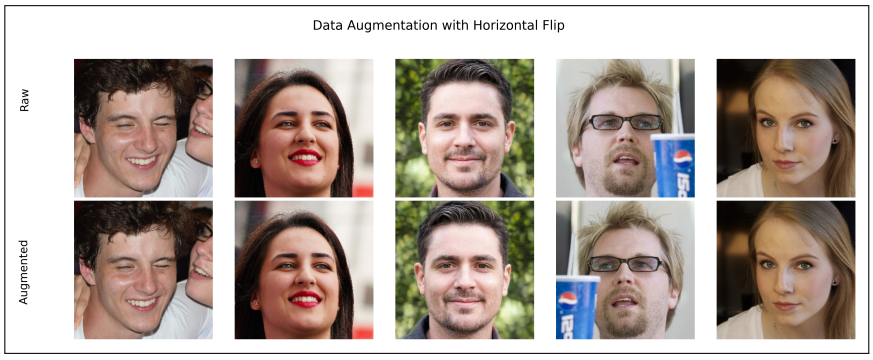
individual yang lebih presisi. Mode kelas tetap binary, dan shuffle dinonaktifkan untuk mempertahankan urutan data asli selama evaluasi.

3.2.4 Strategi Augmentasi Data

Strategi augmentasi data secara sengaja dibatasi hanya pada *horizontal flipping* untuk *set* pelatihan. Keputusan ini didasarkan pada pertimbangan berikut:

- **Preservasi Artefak**: Menghindari augmentasi yang lebih agresif (seperti rotasi, *zoom*, atau *shear*) yang berpotensi menghilangkan atau merusak artefak kompresi halus yang sering menjadi penanda utama citra *deepfake*.
- **Integritas Spasial**: Mempertahankan hubungan spasial fitur wajah yang penting untuk proses deteksi.
- **Efisiensi Komputasi**: Menjaga proses pelatihan agar tetap efisien secara komputasi.

Contoh visual dari penerapan augmentasi sederhana ini disajikan pada Gambar 3.2, yang menunjukkan perbandingan gambar sebelum dan sesudah operasi *horizontal flip*.



Gambar 3.2. Perbandingan gambar sebelum dan sesudah augmentasi sederhana (*horizontal flip*)

3.3 Arsitektur Model Individual

Penelitian ini mengimplementasikan empat arsitektur *deep learning* yang dipilih untuk memberikan keragaman pada model ansambel. Setiap model memiliki karakteristik komplementer, mulai dari arsitektur sederhana yang dirancang khusus hingga model *state-of-the-art* yang memanfaatkan *transfer learning*.

3.3.1 Arsitektur Custom CNN

Sebuah *Convolutional Neural Network (CNN)* dirancang dengan arsitektur hierarkis yang memanfaatkan karakteristik CNN dalam ekstraksi fitur lokal-ke-global, dengan harapan dapat mendeteksi inkonsistensi yang umum pada citra *deepfake*. Spesifikasi lengkap arsitektur disajikan pada Tabel 3.2. Setiap blok konvolusi mengintegrasikan regularisasi *Dropout* untuk mencegah *overfitting*. Peningkatan jumlah *filter* secara hierarkis ($32 \rightarrow 64 \rightarrow 128 \rightarrow 256$) memungkinkan model untuk mempelajari fitur dari level rendah (tepi, tekstur) hingga level tinggi (pola wajah).

Tabel 3.2. Arsitektur dari model CNN

Jenis *Layer*	***Output Shape***	**Konfigurasi**	**Aktivasi**
InputLayer	(256, 256, 3)	-	-
Conv2D	(256, 256, 32)	32 *filter*, 3×3	*ReLU*
Conv2D	(256, 256, 32)	32 *filter*, 3×3	*ReLU*
MaxPooling2D	(128, 128, 32)	2×2 *pool size*	-
Dropout	(128, 128, 32)	*rate*=0.25	-
Conv2D	(128, 128, 64)	64 *filter*, 3×3	*ReLU*
Conv2D	(128, 128, 64)	64 *filter*, 3×3	*ReLU*
MaxPooling2D	(64, 64, 64)	2×2 *pool size*	-
Dropout	(64, 64, 64)	*rate*=0.25	-
Conv2D	(64, 64, 128)	128 *filter*, 3×3	*ReLU*
Conv2D	(64, 64, 128)	128 *filter*, 3×3	*ReLU*
MaxPooling2D	(32, 32, 128)	2×2 *pool size*	-
Dropout	(32, 32, 128)	*rate*=0.25	-
Conv2D	(32, 32, 256)	256 *filter*, 3×3	*ReLU*
Conv2D	(32, 32, 256)	256 *filter*, 3×3	*ReLU*
MaxPooling2D	(16, 16, 256)	2×2 *pool size*	-
Dropout	(16, 16, 256)	*rate*=0.25	-
Flatten	(65536,)	-	-
Dense	(512,)	*L2 reg*=0.01	*ReLU*
BatchNormalization	(512,)	-	-
Dropout	(512,)	*rate*=0.5	-
Dense	(128,)	*L2 reg*=0.01	*ReLU*
BatchNormalization	(128,)	-	-
Dropout	(128,)	*rate*=0.5	-
Dense (Output)	(1,)	-	*Sigmoid*

3.3.2 Arsitektur Transfer Learning

Tiga model lainnya diimplementasikan menggunakan pendekatan *transfer learning* dengan bobot *pre-train* dari *ImageNet*. Strategi *fine-tuning* diterapkan pada semua model ini.

A ResNet50

Model *ResNet50* dipilih karena kemampuannya mengatasi masalah *vanishing gradient* melalui *residual connection*. Strategi *fine-tuning* diterapkan dengan membekukan 100 *layer* pertama untuk mempertahankan fitur umum tingkat rendah, sementara *layer-*layer** berikutnya dilatih ulang untuk beradaptasi dengan domain deteksi *deepfake*.

B Xception

Arsitektur *Xception* (Extreme Inception) memanfaatkan *depthwise separable convolutions* untuk ekstraksi fitur yang efisien. Operasi ini memisahkan korelasi spasial dan *cross-channel*, menjadikannya sangat efektif dalam menangkap inkonsistensi halus pada citra yang dimanipulasi secara digital seperti *deepfake*.

C EfficientNet-B4

EfficientNet-B4 dipilih karena pendekatan *compound scaling* yang inovatif, yang secara seimbang menskalakan kedalaman (*depth*), lebar (*width*), dan resolusi jaringan. Hal ini menghasilkan keseimbangan optimal antara akurasi dan efisiensi komputasi, menjadikannya kandidat yang kuat untuk aplikasi praktis.

3.4 Metodologi Ensemble Weighted Averaging

Untuk meningkatkan robustisitas dan akurasi deteksi, prediksi dari keempat model individual digabungkan menggunakan metode *ensemble weighted averaging*. Bagian ini menjelaskan dasar teoretis, strategi penentuan bobot, implementasi teknis, dan contoh perhitungan konkret dari pendekatan ini.

3.4.1 Fondasi Teoretis

Ensemble weighted averaging beroperasi pada prinsip bahwa kombinasi linear dari beberapa model yang beragam, dengan bobot yang ditetapkan secara strategis, dapat menghasilkan prediksi yang lebih akurat dan stabil daripada prediksi model tunggal mana pun. Berdasarkan formulasi matematika yang telah dijelaskan dalam Bab 2 (Persamaan 2.12).

3.4.2 Strategi Perhitungan Bobot

Bobot (w_i) untuk setiap model tidak ditetapkan secara seragam, melainkan dihitung berdasarkan performa masing-masing model pada *dataset* validasi menggunakan formula yang telah didefinisikan dalam Persamaan 2.24. Pendekatan ini secara intuitif memberikan pengaruh yang lebih besar kepada model dengan performa validasi terbaik, sambil tetap mempertimbangkan kontribusi dari model lainnya.

3.4.3 Contoh Perhitungan Bobot Ensemble

Untuk memberikan pemahaman yang lebih konkret mengenai mekanisme perhitungan bobot, berikut disajikan contoh perhitungan menggunakan data hipotetis yang mudah dipahami:

A Skenario Contoh

Misalkan kita memiliki empat model dengan validation accuracy sebagai berikut:

- Model A (Custom CNN): 95,0%
- Model B (ResNet50): 88,0%
- Model C (Xception): 97,0%
- Model D (EfficientNet): 92,0%

B Langkah 1: Menghitung Total Validation Accuracy

$$\text{Total} = 95,0 + 88,0 + 97,0 + 92,0 = 372,0$$

C Langkah 2: Menghitung Bobot Individual

Menggunakan Persamaan 2.24:

$$w_A = \frac{95,0}{372,0} = 0,255\ (25,5\%) \tag{3.4}$$

$$w_B = \frac{88,0}{372,0} = 0,237\ (23,7\%) \tag{3.5}$$

$$w_C = \frac{97,0}{372,0} = 0,261\ (26,1\%) \tag{3.6}$$

$$w_D = \frac{92,0}{372,0} = 0,247\ (24,7\%) \tag{3.7}$$

D Langkah 3: Verifikasi

$$w_A + w_B + w_C + w_D = 0,255 + 0,237 + 0,261 + 0,247 = 1,000\checkmark$$

3.4.4 Mekanisme Prediksi Ensemble

Setelah bobot diperoleh, prediksi ensemble untuk sebuah input x dihitung sebagai berikut:

A Skenario Prediksi

Misalkan untuk sebuah citra input, masing-masing model menghasilkan probabilitas:

- Model A: 0,85 (85% yakin citra adalah deepfake)
- Model B: 0,78 (78% yakin citra adalah deepfake)
- Model C: 0,92 (92% yakin citra adalah deepfake)
- Model D: 0,81 (81% yakin citra adalah deepfake)

B Perhitungan Prediksi Ensemble

Menggunakan Persamaan 2.12:

$$\hat{y}_{ensemble} = (0,255 \times 0,85) + (0,237 \times 0,78) + (0,261 \times 0,92) + (0,247 \times 0,81) \tag{3.8}$$

$$= 0,217 + 0,185 + 0,240 + 0,200 \tag{3.9}$$

$$= 0,842 \tag{3.10}$$

Hasil ini berarti model ensemble memiliki confidence 84,2% bahwa citra tersebut adalah deepfake.

3.4.5 Kerangka Implementasi

Model *ensemble* dibangun menggunakan API fungsional untuk menggabungkan beberapa model klasifikasi citra yang telah dilatih sebelumnya. Setiap model individual diperlakukan sebagai *feature extractor* dan tidak diperbarui lagi selama proses inferensi dengan membekukan bobotnya.

Implementasi ansambel ini mengadopsi pendekatan *weighted averaging*, di mana proses dimulai dengan pembuatan layer input yang konsisten untuk semua model dengan bentuk (256, 256, 3). Setiap model yang telah dilatih kemudian menerima input yang sama dan menghasilkan output probabilistik individual. Output dari masing-masing model kemudian dikalikan dengan bobot yang telah dihitung berdasarkan performa validasi mereka. Bobot ini mencerminkan kontribusi relatif setiap model terhadap prediksi akhir, dimana model dengan akurasi validasi yang lebih tinggi mendapat bobot yang lebih besar. Hasil perkalian antara output model dan bobotnya kemudian dijumlahkan untuk menghasilkan prediksi ensemble final. Seluruh proses ini diintegrasikan dalam satu model ensemble yang dapat melakukan inferensi secara end-to-end, mengambil input citra dan menghasilkan prediksi gabungan yang memanfaatkan kekuatan kolektif dari semua model individual.

3.5 Prosedur Pelatihan

Untuk memastikan perbandingan yang objektif dan valid antara model individual dan pendekatan *ensemble*, penelitian ini menerapkan serangkaian prosedur pelatihan yang terstandarisasi. Bagian ini merinci protokol pelatihan yang digunakan, termasuk konfigurasi *hyperparameter* dan *callback*.

3.5.1 Protokol Pelatihan

Setiap model individual dilatih menggunakan protokol yang konsisten untuk menjamin keadilan dalam perbandingan. Perbedaan hanya terdapat pada *learning rate* awal, yang disesuaikan berdasarkan karakteristik arsitektur (*custom* vs. *transfer learning*).

A Konfigurasi *Hyperparameter*

Hyperparameter yang digunakan selama fase pelatihan dirangkum dalam Tabel 3.3.

Tabel 3.3. *Hyperparameter* yang digunakan untuk pelatihan model

Parameter	***Custom CNN***	**Model *Transfer Learning***
Optimizer	*Adam*	*Adam*
Learning Rate	1×10^{-4}	5×10^{-5}
Fungsi *Loss*	*Binary Crossentropy*	*Binary Crossentropy*
Batch Size (Latih)	32	32
Batch Size (Val)	16	16
Max Epochs	15	15
Early Stopping Patience	3	3

Hyperparameter dipilih berdasarkan best practices dan eksperimen awal:

- Learning Rate: Model custom menggunakan 1×10^{-4} (lebih tinggi karena training from scratch), sedangkan transfer learning menggunakan 5×10^{-5} (lebih rendah untuk fine-tuning)
- *Adam Optimizer*: Dipilih karena *adaptive learning rate* dan *momentum*
- *Binary Crossentropy*: Sesuai untuk klasifikasi biner
- Batch Size: 32 untuk *training* memberikan keseimbangan antara stabilitas dan efisiensi memori
- *Early Stopping Patience 3*: Mencegah *overfitting* dengan menghentikan *training* jika tidak ada peningkatan selama 3 *epoch* berturut-turut

B Konfigurasi callback pada model pelatihan

Untuk mengelola proses pelatihan secara otomatis, serangkaian *callback* diimplementasikan untuk mengoptimalkan hasil pelatihan. Konfigurasi *callback* memiliki dua komponen utama yang bekerja secara sinergis.

Komponen pertama adalah *EarlyStopping* yang berfungsi sebagai mekanisme pengawasan otomatis terhadap proses pelatihan. Callback ini secara kontinyu memonitor *validation loss* pada setiap epoch dan akan menghentikan proses pelatihan secara otomatis jika tidak ada perbaikan yang signifikan selama 3 epoch berturut-turut. Parameter *patience* diatur pada nilai 3 untuk memberikan toleransi yang wajar terhadap fluktuasi minor dalam performa validasi. Fitur *restore best weights* diaktifkan untuk memastikan bahwa ketika pelatihan dihentikan, model akan dikembalikan ke bobot terbaik yang pernah dicapai selama proses pelatihan, bukan bobot dari epoch terakhir yang mungkin sudah mengalami overfitting.

Komponen kedua adalah *ModelCheckpoint* yang berperan sebagai sistem penyimpanan otomatis untuk model terbaik. Callback ini secara terus-menerus memantau *validation loss* dan secara otomatis menyimpan snapshot model setiap kali ditemukan performa yang lebih baik. File model disimpan dengan nama yang mencakup identifikasi model spesifik untuk memudahkan identifikasi. Parameter *save best only* diaktifkan untuk memastikan hanya model dengan performa terbaik yang disimpan, menghemat ruang penyimpanan dan memudahkan proses evaluasi selanjutnya. Mode monitoring diatur pada 'min' karena tujuannya adalah meminimalkan validation loss.

3.6 Lingkungan Komputasi dan Reprodusibilitas

Bagian ini merinci spesifikasi perangkat keras dan lunak yang digunakan dalam eksperimen untuk memastikan transparansi dan memfasilitasi reprodusibilitas hasil penelitian.

3.6.1 Spesifikasi Perangkat Keras dan Lunak

Eksperimen dilakukan pada platform *cloud computing* untuk memanfaatkan akselerasi GPU. Konfigurasi utama lingkungan komputasi disajikan pada Tabel 3.4 dan 3.5.

Tabel 3.4. Konfigurasi perangkat keras

Komponen	**Spesifikasi**
Platform	*Google Colaboratory Pro+*
GPU	*NVIDIA A100 / V100*
Memori GPU	$\geq$ 16 GB GDDR6
RAM Sistem	$\geq$ 16 GB
Penyimpanan	*Google Drive Cloud Storage*

Tabel 3.5. Dependensi utama yang digunakan untuk pengembangan model *machine learning*

***Framework*/Pustaka**	**Versi**
Python	3.9+
TensorFlow	2.12+
Keras	2.12+
NumPy	1.23+
Pandas	1.5+
Scikit-learn	1.2+
Matplotlib	3.7+

3.6.2 Langkah Reprodusibilitas

Untuk menjamin bahwa hasil eksperimen dapat direplikasi, beberapa langkah penting diterapkan dalam proses implementasi. Reprodusibilitas menjadi aspek krusial dalam eksperimen berbasis pembelajaran mesin, mengingat banyak proses di dalamnya bersifat stokastik, seperti inisialisasi bobot, pemilihan batch, dan augmentasi data.

Langkah paling mendasar adalah dengan menetapkan *random seed* secara konsisten pada semua pustaka yang digunakan. Pustaka seperti Python random, NumPy, dan TensorFlow memiliki sumber acak masing-masing yang perlu dikendalikan agar eksperimen dapat menghasilkan hasil yang konsisten setiap kali dijalankan. Proses inisialisasi reprodusibilitas dimulai dengan pengaturan seed untuk modul random Python standar, dilanjutkan dengan konfigurasi seed untuk NumPy yang mengendalikan operasi matematis acak, dan kemudian pengaturan seed untuk TensorFlow yang mengatur randomness dalam operasi deep learning. Selain itu, variabel lingkungan PYTHONHASHSEED juga diatur untuk menghindari variasi akibat proses hashing internal Python. Semua pengaturan ini dilakukan dengan menggunakan nilai seed yang tetap, yaitu 42, untuk memastikan konsistensi across different runs.