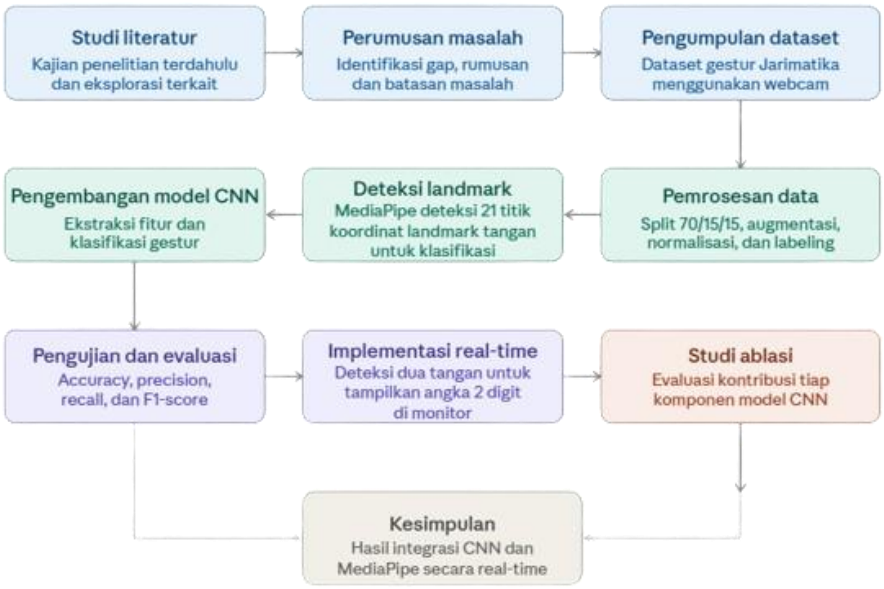
BAB III

METODOLOGI PENELITIAN

3.1 Tahapan Penelitian

Dalam penelitian ini, terdapat beberapa tahapan yang dilakukan. Dimulai dari Studi Literatur, Perumusan Masalah, Pengumpulan Dataset, kemudian Prapemrosesan Data, Deteksi *landmark* tangan menggunakan MediaPipe, Pengembangan Model, Pengujian dan Evaluasi Model, Implementasi *Real-Time* dan kesimpulan.



Gambar 3. 1 Tahapan Penelitian

Tahap penelitian dilakukan untuk mengkaji teori, algoritma, dan penelitian terdahulu yang relevan guna membangun fondasi pemahaman terkait pengenalan gestur tangan Jarimatika.

3.1.1 Permasalahan Penelitian

Perancangan tahap ini dilakukan identifikasi *gap* penelitian dari literatur yang akan diterapkan menjadi rumusan masalah penelitian, tujuan penelitian, dan batasan masalah. Masalah yang akan diselesaikan dalam penelitian ini yaitu dengan mengembangkan sistem deteksi gestur jari tangan jarimatika dengan mengintegrasikan CNN dan MediaPipe.

3.1.2 Pengumpulan Dataset

Tahap pengumpulan dataset dilakukan dengan mengambil citra gestur Jarimatika menggunakan *webcam* secara mandiri. Dataset mencakup variasi posisi tangan dan latar belakang yang berbeda-beda untuk meningkatkan kemampuan generalisasi model. Dataset terdiri dari 18 kelas gestur, di mana tangan kanan merepresentasikan nilai satuan dan tangan kiri merepresentasikan nilai puluhan. Distribusi kelas dataset ditunjukkan pada Tabel 3.1.

Tabel 3. 1. Distribusi Dataset Gestur Jarimatika

Gestur	Total Kelas	Citra per kelas	Total Citra
Gestur Tangan Kanan (Satuan)	9	600	5.400
Gestur Tangan Kiri (Puluhan)	9	600	5.400
Total	18	600	10.800

Berdasarkan Tabel 3.1, setiap kelas memiliki jumlah citra yang seimbang yaitu 600 citra sehingga model tidak condong (*bias*) terhadap kelas tertentu selama pelatihan. Total keseluruhan dataset yang dikumpulkan adalah 10.800 citra dari 18 kelas gestur Jarimatika. Selanjutnya, dataset dibagi menjadi tiga subset yaitu data latih, data validasi, dan data uji dengan rasio 70:15:15 sebagaimana ditunjukkan pada Tabel 3.2.

Tabel 3. 2 Pembagian Dataset

Subset	Rasio	Jumlah Citra
Data Latih	70%	7.560
Data Validasi	15%	1.620
Data Uji	15%	1.620
Total	100%	10.800

Data latih digunakan untuk melatih bobot model, data validasi digunakan untuk memantau performa model selama pelatihan berlangsung, dan data uji digunakan untuk mengevaluasi performa akhir model secara independen terhadap data yang belum pernah dilihat sebelumnya.

3.1.3 Prapemrosesan Data

Tahap prapemrosesan dilakukan untuk memastikan data siap digunakan dalam pelatihan model. Prapemrosesan yang diterapkan meliputi augmentasi data, normalisasi, dan pelabelan kelas. Augmentasi data diterapkan pada data latih untuk meningkatkan variasi dataset dan kemampuan generalisasi model. Augmentasi yang digunakan meliputi rotasi citra hingga 15 derajat, perubahan kecerahan dalam rentang 0,8–1,2, dan *zoom* sebesar 5%. Augmentasi yang digunakan secara ringan agar variasi yang dihasilkan tetap realistis dan tidak mengubah karakteristik gestur secara berlebihan. Data validasi dan data uji tidak dikenai augmentasi untuk menjaga objektivitas evaluasi.

Normalisasi piksel dilakukan dengan membagi nilai piksel citra dengan 255 seperti pada persamaan (4.5) sehingga nilai piksel berada pada rentang [0, 1]. Normalisasi ini diterapkan pada seluruh subset data latih, validasi, maupun uji untuk menstabilkan proses pelatihan dan mempercepat konvergensi model. Pelabelan kelas dilakukan secara otomatis berdasarkan nama folder pada masing-masing subset dataset. Setiap folder mewakili satu kelas gestur dengan label sesuai

angka Jarimatika yang direpresentasikan, yaitu (01, 02, 03, 04, 05, 06, 07, 08, 09) untuk gestur satuan dan (10, 20, 30, 40, 50, 60, 70, 80, 90) untuk gestur puluhan.

3.1.4 Deteksi *Landmark* Tangan dengan MediaPipe

Pada tahap ini, MediaPipe *Hands* digunakan untuk mendeteksi dan melacak posisi tangan secara *real-time* dari citra yang ditangkap *webcam*. Proses deteksi dilakukan dengan pengguna memposisikan tangan di depan *webcam* pada jarak sekitar 30–60 cm dari kamera dengan pencahayaan yang cukup. Tangan harus terlihat jelas dalam *frame* tanpa terhalang objek lain agar MediaPipe *Hands* dapat mendeteksi seluruh 21 titik *landmark* secara akurat. Sistem mendukung deteksi hingga dua tangan secara bersamaan dalam satu *frame*, yang memungkinkan pengenalan bilangan dua digit Jarimatika dilakukan dalam satu waktu. MediaPipe *Hands* mampu mendeteksi 21 titik koordinat (*landmark*) pada setiap tangan, mencakup *keypoint* pada setiap ruas jari, ujung jari, dan telapak tangan. Deteksi dilakukan dengan nilai *minimum detection confidence* dan *minimum tracking confidence* masing-masing sebesar 0,7 untuk memastikan hanya tangan yang terdeteksi dengan keyakinan tinggi yang diproses lebih lanjut.

Koordinat *landmark* yang diperoleh digunakan untuk menghitung batas *bounding box* area tangan berdasarkan nilai minimum dan maksimum koordinat pada sumbu X dan Y terdapat pada persamaan (4.1)–(4.4). Area tangan yang telah dibatasi kemudian diekstraksi sebagai ROI untuk dimasukkan ke model CNN.

MediaPipe *Hands* menyediakan informasi *multi_handedness* yang digunakan untuk membedakan tangan kanan dan tangan kiri secara otomatis. Informasi ini penting

dalam sistem Jarimatika karena tangan kanan merepresentasikan nilai satuan dan tangan kiri merepresentasikan nilai puluhan, sehingga sistem dapat menggabungkan hasil klasifikasi kedua tangan menjadi bilangan dua digit secara otomatis.

3.1.5 Pengembangan Model CNN

Berbeda dengan algoritma klasifikasi konvensional, model berbasis *deep learning* seperti CNN mengekstraksi fitur sekaligus melakukan klasifikasi citra dalam satu proses yang terintegrasi (Sely Wita & Yanti Liliana, 2022). Pada penelitian ini, model CNN dikembangkan untuk mengklasifikasikan citra ROI hasil ekstraksi area tangan ke dalam 18 kelas gestur Jarimatika, yaitu sembilan kelas gestur satuan yang direpresentasikan oleh tangan kanan dan sembilan kelas gestur puluhan yang direpresentasikan oleh tangan kiri.

Model terdiri dari tiga blok konvolusi yang masing-masing diikuti oleh *Batch Normalization* dan *Max Pooling*, dilanjutkan dengan lapisan *fully connected* untuk klasifikasi akhir sebagaimana ditunjukkan pada Tabel 3.3.

Tabel 3. 3 Arsitektur CNN

Layer	Tipe	Filter	Kernel	Aktivasi	Output Shape
1	Conv2D	32	3x3	ReLU	126x126x32
2	BatchNormalization	-	-	-	126x126x32
3	MaxPooling2D	-	2x2	-	63x63x32
4	Conv2D	64	3x3	ReLU	61x61x64
5	BatchNormalization	-	-	-	61x61x64
6	MaxPooling2D	-	2x2	-	30x30x64
7	Conv2D	128	3x3	ReLU	28x28x1288
8	BatchNormalization	-	-	-	28x28x128
9	MaxPooling2D	-	-	-	14x14x128

10	Flatten	-	-	-	25.088
11	Dense	128	-	ReLU	128
12	Dropout(0.5)	-	-	-	128
13	Dense	18	-	Softmax	18

Model menerima input citra berukuran 128×128 piksel (RGB). Jumlah filter ditingkatkan bertahap dari 32, 64, hingga 128 agar model dapat mempelajari fitur yang semakin kompleks. *Batch Normalization* diterapkan setelah setiap lapisan konvolusi untuk menstabilkan distribusi nilai aktivasi, sedangkan *Dropout* 0,5 pada lapisan *Dense* berfungsi sebagai regularisasi untuk mencegah *overfitting*. Lapisan keluaran menggunakan aktivasi *softmax* sesuai persamaan (2.4) untuk menghasilkan distribusi probabilitas pada 18 kelas gestur.

Pelatihan dilakukan selama 30 *epoch* dengan *batch size* 16 menggunakan optimizer *Adam* (*learning rate* 0,001) dan *loss categorical cross-entropy*. Tiga *callback* diterapkan sebagaimana ditunjukkan pada Tabel 3.4.

Tabel 3. 4 Konfigurasi *Callback* Pelatihan

Callback	**Parameter**	**Fungsi**
ModelCheckpoint	monitor: val_loss	Menyimpan bobot model terbaik secara otomatis
ReduceLROnPlateau	factor: 0,5, patience: 3	Mengurangi *learning rate* jika val_loss stagnan
EarlyStopping	patience: 5	Menghentikan pelatihan jika val_loss tidak membaik

Augmentasi data diterapkan pada data latih meliputi rotasi hingga 15 derajat, perubahan kecerahan dalam rentang 0,8–1,2, dan *zoom* sebesar 5% untuk meningkatkan variasi dan kemampuan generalisasi model.

3.1.6 Pengujian dan Evaluasi Model

Tahap pengujian dan evaluasi dilakukan untuk mengukur performa model CNN yang telah dilatih terhadap data yang belum pernah dilihat sebelumnya. Pengujian dilakukan menggunakan data uji sebanyak 1.620 citra (90 citra per kelas) yang terpisah dari data latih maupun data validasi. Proses pengujian dilaksanakan menggunakan skrip Python.

Evaluasi kinerja model dilakukan menggunakan empat metrik utama yaitu *accuracy*, *precision*, *recall*, dan F1-*score*. *Accuracy* mengukur proporsi prediksi yang benar terhadap seluruh data uji. *Precision* mengukur proporsi prediksi positif yang benar. *Recall* mengukur proporsi data aktual yang berhasil dikenali dengan benar oleh model. F1-*score* merupakan rata-rata harmonik antara *precision* dan *recall* yang digunakan sebagai metrik keseimbangan antara keduanya. Keempat metrik dihitung menggunakan persamaan berikut:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (3.\ 1)$$

$$Precision = \frac{TP}{TP+FP} \quad (3.\ 2)$$

.

$$Recall = \frac{TP}{TP+FN} \quad (3.\ 3)$$

$$F1\ Score = 2 \times \frac{Precision \times Recall}{Precision+Recall} \quad (3.\ 4)$$

TP (*true positive*) adalah jumlah data positif yang diprediksi benar, TN (*true negative*) adalah jumlah data negatif yang diprediksi benar, FP (*false positive*) adalah jumlah data negatif yang salah diprediksi sebagai positif, dan FN (*false negative*) adalah jumlah data positif yang salah diprediksi sebagai negatif. Karena

penelitian ini menggunakan klasifikasi multikelas dengan 18 kelas gestur, nilai *precision*, *recall*, dan F1-*score* dihitung untuk setiap kelas dan dirata-ratakan menggunakan *macro average*.

Selain keempat metrik tersebut, analisis *confusion matrix* juga dilakukan untuk mengevaluasi distribusi hasil prediksi model secara visual. *Confusion matrix* menampilkan jumlah prediksi benar dan salah untuk setiap kelas sehingga dapat diidentifikasi kelas-kelas yang paling sering mengalami kesalahan klasifikasi.

Pengujian kecepatan sistem dilakukan untuk mengukur kemampuan sistem dalam memproses *frame* secara *real-time* menggunakan metrik *Frames Per Second* (FPS) sesuai persamaan (4.6). Pengujian dilakukan pada dua kondisi yaitu deteksi satu tangan dan deteksi dua tangan secara bersamaan untuk menganalisis pengaruh jumlah tangan terhadap beban komputasi sistem.