BAB IV
ANALISA DATA DAN PEMBAHASAN

4.1 Analisis Data

Analisis data pada penelitian ini bertujuan untuk menjelaskan tahapan pengolahan dataset, proses pelatihan model, serta evaluasi hasil pengujian sistem IMAJI. Melalui analisis ini, penulis dapat menilai sejauh mana model Convolutional Neural Network (CNN) yang dikembangkan mampu mengenali pola sketsa sesuai dengan struktur dataset yang telah disusun

4.1.1 Pengolahan *Dataset*

Pengolahan *Dataset* merupakan tahap awal yang penting dalam proses pelatihan *model Convolutional Neural Network* (CNN) pada sistem IMAJI. *Dataset* pada penelitian ini dikembangkan secara mandiri, disesuaikan dengan kebutuhan aplikasi pembelajaran ilustrasi sketsa. Data terdiri dari gambar-gambar sketsa yang dibagi berdasarkan kategori objek serta urutan langkah menggambar, mulai dari bentuk dasar hingga hasil akhir yang lebih kompleks. Setiap gambar dikategorikan ke dalam empat langkah, yang merepresentasikan alur belajar pengguna dalam membentuk gambar secara bertahap.

Penulis menggunakan *Dataset* berdasarkan pembagian *Dataset* yang ada pada tabel 3.5, dengan struktur direktori yang sistematis, di mana setiap folder utama merepresentasikan satu jenis gambar (misalnya kucing, anjing dan ikan), setiap subfolder di dalamnya mewakili tahapan tertentu dalam proses menggambar, seperti `step1` hingga `step4`. Struktur ini memudahkan dalam mengelola dan mengelompokkan data berdasarkan progres menggambar. Berikut struktur direktori *Dataset* yang digunakan :

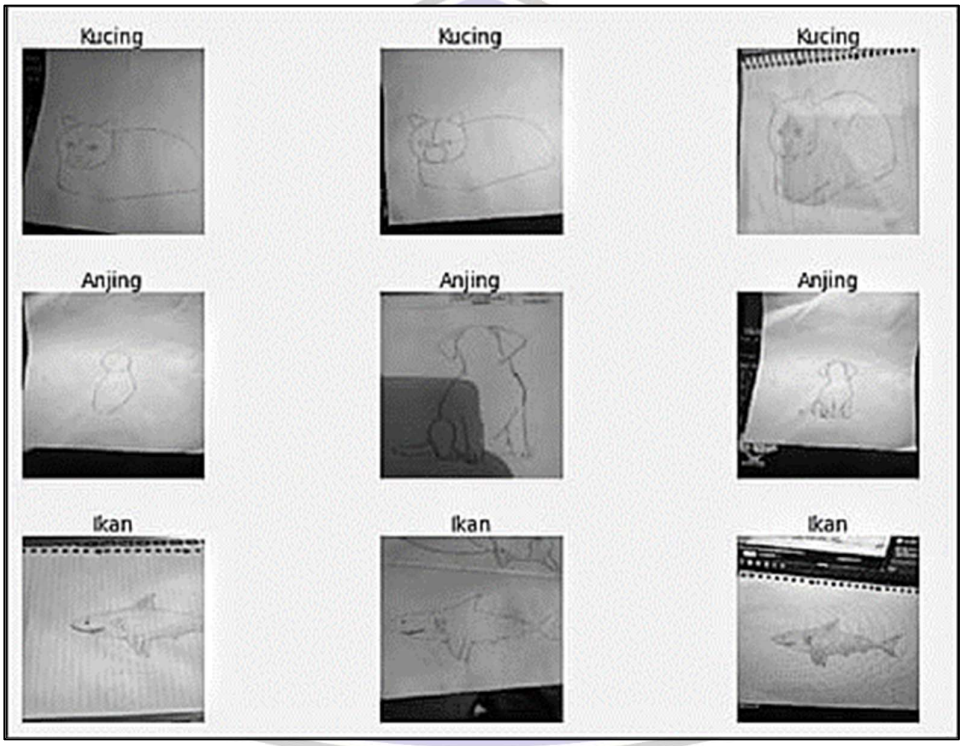
```
Dataset/
├── train/
│   ├── kucing_step1/
│   ├── .../
│   ├── kucing_step4/
│   ├── ikan_step1/
│   ├── .../
│   ├── anjing_step1/
│   └── .../
```

```
├── val/...
├── test/...
```

Dataset dibagi menjadi tiga bagian utama dengan komposisi sebagai berikut:

a. *Train*: 70%, Digunakan untuk melatih *model*.
b. *Validation*: 15%, Digunakan untuk memantau kinerja *model* selama pelatihan.
c. *Test*: 15%, Digunakan untuk menguji performa akhir *model*.

Untuk memberikan gambaran visual terhadap data yang digunakan dalam pelatihan *model* CNN, berikut ditampilkan sejumlah contoh gambar dari *Dataset* hasil pengumpulan dan klasifikasi berdasarkan kategori objek



Gambar 4.1 Contoh Gambar didalam *Dataset*

Gambar 4.1 menampilkan cuplikan data dari tiga kelas utama dalam *Dataset*, yaitu kucing, ikan, dan anjing. Setiap kelas ditampilkan sebanyak lima gambar yang diambil secara acak dari keseluruhan *Dataset*. Visualisasi ini menunjukkan variasi gaya sketsa dalam setiap kelas, baik dari segi bentuk, detail goresan, maupun sudut pandang gambar. Gambar-gambar ini dikumpulkan dan dikategorikan secara manual untuk memastikan kualitas data dan kesesuaian dengan tahapan proses menggambar.

Keberagaman pada setiap kelas ini penting untuk memastikan *model* CNN yang dilatih dapat mengenali pola dan karakteristik unik dari setiap objek, serta meningkatkan generalisasi *model* saat digunakan dalam aplikasi nyata.

4.1.2 Arsitektur *model* CNN

Model CNN yang digunakan dalam sistem IMAJI dibangun dengan memanfaatkan arsitektur *MobileNetV2*, yaitu *model pretrained* yang dirancang untuk efisiensi dan kinerja optimal pada perangkat mobile. *MobileNetV2* memiliki keunggulan dalam hal kecepatan inferensi dan ukuran *model* yang ringan, sehingga sangat cocok diimplementasikan dalam aplikasi *Android* seperti IMAJI.

Model ini dibangun menggunakan pustaka *TensorFlow* dan *Keras*, dengan pendekatan *transfer learning* dari *model MobileNetV2* yang telah dilatih sebelumnya pada *Dataset ImageNet*. Dengan metode ini, sebagian besar bobot *pretrained* digunakan ulang, dan hanya beberapa layer akhir yang disesuaikan dan dilatih ulang sesuai kebutuhan klasifikasi gambar sketsa dalam aplikasi.

Proses pembangunan arsitektur dilakukan dengan menambahkan beberapa *layer fully connected* setelah *base model MobileNetV2*. Penyesuaian dilakukan dengan menambahkan *layer GlobalAveragePooling*, Dense, *BatchNormalization*, serta *Dropout* untuk mengurangi risiko *overfitting*. Berikut potongan kode-nya:

```
base_model = MobileNetV2(weights='imagenet', include_top=False,
input_shape=(224, 224, 3))
base_model.trainable = True

for layer in base_model.layers[:-30]:
    layer.trainable = False

model = models.Sequential([
    base_model,
    layers.GlobalAveragePooling2D(),
    layer.Dense(256, activation='relu')
    batchNormalization(),
    layers.Dropout(0.3),
    layers.Dense(128, activation='relu'),
    layers.Dropout(0.3),
```

```
    layers.Dense(num_classes, activation='softmax')
])
model = Model(inputs=base_model.input, outputs=output)
model.summary()
```

Setelah arsitektur *model* berhasil dibangun, dilakukan pemeriksaan struktur *model* menggunakan fungsi *model.summary()*. Fungsi ini menampilkan detail dari setiap *layer*, bentuk output, serta jumlah parameter yang dapat dan tidak dapat dilatih.Ilustrasi arsitektur *model MobileNetV2* yang digunakan dapat dilihat pada Gambar 4.2 berikut,

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 rescaling_1 (Rescaling)     (None, 224, 224, 3)       0

 mobilenetv2_1.00_224 (Func  (None, 7, 7, 1280)        2257984
 tional)

 global_average_pooling2d_1  (None, 1280)              0
  (GlobalAveragePooling2D)

 dense_3 (Dense)             (None, 256)               327936

 batch_normalization_1 (Bat  (None, 256)               1024
 chNormalization)

 dropout_2 (Dropout)         (None, 256)               0

 dense_4 (Dense)             (None, 128)               32896

 dropout_3 (Dropout)         (None, 128)               0

 dense_5 (Dense)             (None, 12)                1548

=================================================================
Total params: 2621388 (10.00 MB)
Trainable params: 1889292 (7.21 MB)
Non-trainable params: 732096 (2.79 MB)
_________________________________________________________________
```

Gambar 4.2 Arsitektur *Model* CNN MobilNetV2

Gambar 4.2 menunjukkan struktur layer dari *model* CNN yang digunakan. *Model* terdiri dari *layer rescaling*, *MobileNetV2* sebagai *backbone*, kemudian diikuti dengan *layer dense*, *batch normalization*, dan *dropout*. *Layer output*

memiliki 12 *neuron*, sesuai dengan jumlah total kelas pada *Dataset* yang digunakan. Terlihat bahwa *model* memiliki total 2.6 juta parameter, dengan sekitar 1.88 juta parameter dapat dilatih, yang menunjukkan kombinasi antara efisiensi dan fleksibilitas untuk fine-tuning.

4.1.3 Pelatihan *Model* CNN

Pelatihan *model* CNN dilakukan untuk menghasilkan *model* kecerdasan buatan yang mampu mengenali gambar sketsa sesuai kategori dan urutan langkah menggambar. *Model* ini dilatih menggunakan *Dataset* yang telah diproses dan dibagi ke dalam data latih, validasi, dan uji. Pelatihan dilakukan dengan memanfaatkan pustaka *TensorFlow* dan *Keras*, menggunakan arsitektur yang disesuaikan dengan jenis citra dan target klasifikasi.

a. Parameter Pelatihan

Model CNN dilatih dengan sejumlah parameter penting untuk memastikan proses konvergensi berjalan optimal. Pemilihan nilai parameter ini didasarkan pada praktik umum dalam pelatihan jaringan saraf konvolusional, serta melalui percobaan awal yang telah dilakukan.

Tabel 4. 1 Parameter Pelatihan

Parameter	Nilai
Optimizer	*Adam*
Learning Rate	0.001
Loss Function	*Categorical Crossentropy*
Batch Size	32
Jumlah Epoch	20
Activation Function	*ReLU (hidden), Softmax (output)*
Dropout Rate	0.3
Input Image Size	256 x 256
Jumlah Kelas	12 (3 objek × 4 langkah)

b. Visual hasil pelatihan

Selama pelatihan *model* berlangsung, tensorflow menghasilkan log pada setiap epoch yang mencatat nilai metrik *loss*, *accuracy*, *val_loss* dan *val_accuracy*.

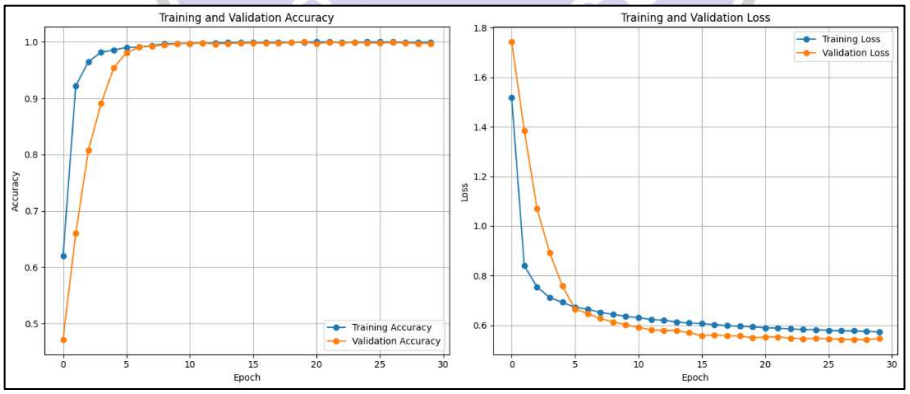
Pada Gambar 4.3 berikut Menampilkan potongan log kemajuan *loss*, *accuracy*, *val_loss* dan *val_accuracy* pada saat dilatih.

```
Epoch 1/30
188/188 ━━━━━━━━━━━━━━━━━━━━ 1385s 7s/step - accuracy: 0.4253 - loss: 2.0180 - val_accuracy: 0.4717 - val_loss: 1.7429
Epoch 2/30
188/188 ━━━━━━━━━━━━━━━━━━━━ 35s 42ms/step - accuracy: 0.9022 - loss: 0.8792 - val_accuracy: 0.6600 - val_loss: 1.3851
Epoch 3/30
188/188 ━━━━━━━━━━━━━━━━━━━━ 8s 41ms/step - accuracy: 0.9598 - loss: 0.7663 - val_accuracy: 0.8078 - val_loss: 1.0709
Epoch 4/30
...
Epoch 29/30
188/188 ━━━━━━━━━━━━━━━━━━━━ 10s 48ms/step - accuracy: 0.9992 - loss: 0.5760 - val_accuracy: 0.9972 - val_loss: 0.5423
Epoch 30/30
188/188 ━━━━━━━━━━━━━━━━━━━━ 8s 40ms/step - accuracy: 0.9996 - loss: 0.5733 - val_accuracy: 0.9972 - val_loss: 0.5460
```

Gambar 4.3 Hasil Log dari epoch

Gambar 4.3 menunjukkan potongan *Log* progres pelatihan selama 30 *epoch*. Terlihat bahwa pada epoch pertama, akurasi *model* masih rendah (sekitar 42%) dengan nilai *val_accuracy* sebesar 47%. Namun, pada epoch ke-3, akurasi meningkat signifikan menjadi 95.9%, dan *val_accuracy* mencapai 80%. Pada akhir pelatihan (*epoch* 30), *model* berhasil mencapai akurasi *training* sebesar 99.96% dan akurasi validasi sebesar 99.72%, menunjukkan bahwa *model* belajar dengan sangat baik dan mampu menggeneralisasi data validasi.

Untuk mempermudah interpretasi performa *model*, *log* pelatihan tersebut divisualisasikan dalam bentuk grafik. Gambar 4.4 memperlihatkan tren peningkatan akurasi serta penurunan nilai loss terhadap data latih dan validasi selama 30 epoch.



Gambar 4.4 Grafik Akurasi *Training Loss* CNN *MobileNetV2*

Berdasarkan Gambar 4.4, grafik akurasi menunjukkan bahwa nilai akurasi *training* meningkat dengan sangat cepat sejak awal pelatihan, mencapai lebih dari 98% pada epoch ke-5 dan tetap stabil hingga akhir pelatihan. Akurasi validasi juga mengalami peningkatan tajam dan konsisten, mendekati nilai akurasi *training*, sehingga menandakan kemampuan generalisasi *model* yang sangat baik.

Pada grafik loss, baik *training* loss maupun validation loss mengalami penurunan yang signifikan seiring bertambahnya epoch. Titik minimum loss tercapai sekitar epoch ke-16, setelah itu nilai loss cenderung stabil dengan sedikit fluktuasi pada validation loss. Secara keseluruhan, tren loss tetap menurun dan tidak menunjukkan gejala overfitting yang berarti, karena gap antara *training* loss dan validation loss sangat kecil.

Dengan demikian, *model* CNN *MobileNetV*2 yang digunakan telah berhasil melakukan pembelajaran secara efektif dan optimal, serta mampu melakukan generalisasi dengan baik pada data validasi.

4.1.4 Integrasi *Model* CNN

Setelah proses pelatihan selesai, *model* CNN yang telah dikembangkan dikonversi ke format *TensorFlow Lite* (*.tflite*) agar dapat dijalankan secara efisien pada perangkat Android. Proses konversi dilakukan menggunakan *TensorFlow Lite Converter*, kemudian *model* diintegrasikan ke dalam aplikasi IMAJI melalui *Android Studio*. Integrasi ini memungkinkan sistem untuk melakukan prediksi langsung dari input gambar pengguna secara real-time.

a. Konversi *model*

Model CNN awalnya disimpan dalam format `.h5` setelah proses pelatihan menggunakan *TensorFlow* dan *Keras*. Untuk digunakan di Android, *model* ini perlu dikonversi ke format `.tflite` dengan menggunakan *TensorFlow Lite Converter***.** Berikut adalah potongan kode untuk melakukan konversi *model*:

```
model.save("model_sketsa.h5")
converter = tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model = converter.convert()
with open("model_imaji_new.tflite", "wb") as f:
    f.write(tflite_model)
```

```
print("Model berhasil dikonversi ke model_sketsa.tflite")
```

b. Penempatan *model*

Setelah dikonversi, file `model_cnn_imaji.tflite` ditempatkan ke dalam direktori proyek Android di:

```
app/src/main/assets/
```

Penempatan *model* pada folder *assets* memungkinkan *model* diakses sebagai sumber daya statis oleh aplikasi dan dimuat saat runtime melalui *interpreter TensorFlow Lite.*

c. Implementasi *TFLite* di Android

Integrasi *model* ke dalam aplikasi IMAJI dilakukan menggunakan *API TensorFlow Lite Interpreter* di dalam *Android Studio* (dalam kasus ini, menggunakan *Kotlin*). Proses pemanggilan *model* dilakukan di salah satu file yakni `CameraFragment`, yang bertanggung jawab mengambil gambar dari kamera dan mengirimkannya ke *model* untuk dilakukan inferensi.

```
val model = FileUtil.loadMappedFile(requireContext(),
"model_imaji_new.tflite")
val options = Interpreter.Options()
// Preprocessing input bitmap -> ByteBuffer
// Menjalankan inferensi
tflite = Interpreter(model, options)
```

Model akan menerima gambar input yang telah diubah ukurannya (misalnya 256x256 piksel, *grayscale*/RGB sesuai *model*) dan mengembalikan output berupa skor probabilitas untuk masing-masing kelas gambar sketsa.

4.2 Pembahasan

Pembahasan pada penelitian ini difokuskan pada proses implementasi serta evaluasi kinerja sistem IMAJI yang telah dikembangkan. Bagian ini menjelaskan bagaimana integrasi antara model Convolutional Neural Network (CNN), teknologi Augmented Reality (AR), serta pengelolaan data dilakukan sehingga menghasilkan aplikasi pembelajaran ilustrasi sketsa yang interaktif dan adaptif terhadap kebutuhan pengguna.

4.2.1 Implementasi Sistem

Setelah *model* CNN berhasil diintegrasikan ke dalam aplikasi IMAJI, tahap implementasi mencakup penggabungan berbagai komponen utama, termasuk antarmuka pengguna (UI), teknologi *Augmented Reality* (AR), proses inferensi *model* AI, dan pengelolaan data menggunakan *Supabase PostgreSQL*. Untuk memudahkan pemahaman, berikut ini disajikan alur implementasi sistem berdasarkan tampilan-tampilan utama yang digunakan pengguna dalam aplikasi IMAJI.

a. Beranda dan Pemilihan *Template*

Gambar 4.5 Pilih *template* latihan (Beranda)

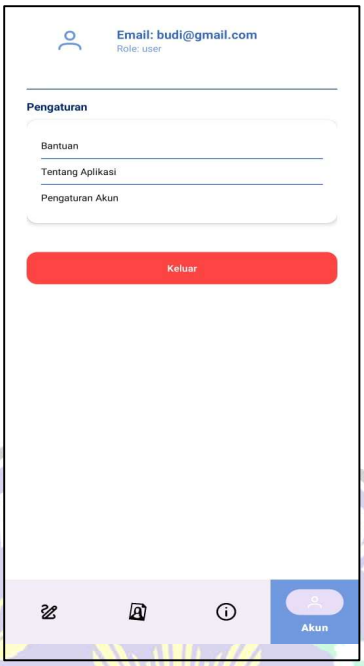
Pada Gambar 4.5, Halaman awal aplikasi menampilkan daftar *template* yang dapat dipilih oleh pengguna untuk memulai sesi latihan menggambar. Pemilihan *template* akan menentukan gambar panduan yang ditampilkan pada fitur AR.

b. Galeri Hasil Latihan

Gambar 4.6 Galeri progres pengguna

Pada Gambar 4.6, Fitur ini menampilkan riwayat latihan pengguna, termasuk gambar yang telah dicoba dan skor yang diperoleh pada setiap langkah. Pengguna dapat meninjau kemajuan mereka dari waktu ke waktu.

c. Profil dan Akun Pengguna



Gambar 4.7 Halaman Profil Pengguna

Gambar 4.7 menunjukkan aplikasi mendukung fitur autentikasi menggunakan *Supabase*, memungkinkan pengguna untuk login dan mengelola akun mereka. Halaman profil juga menampilkan informasi akun serta ringkasan skor dan jumlah latihan yang telah dilakukan.

d. Dashboard *Admin*

Gambar 4.8 Halaman Utama *Admin*

Untuk mendukung pengelolaan data latihan, *template*, dan pengguna, aplikasi menyediakan halaman khusus untuk peran *Admin*. Fitur ini hanya dapat diakses oleh akun yang memiliki hak akses khusus.

Dengan implementasi sistem yang menyeluruh ini, aplikasi IMAJI mampu memberikan pengalaman belajar menggambar berbasis teknologi AI dan AR secara real-time, efektif, dan menyenangkan.

4.2.2 Evaluasi Sistem

Evaluasi sistem dilakukan untuk menilai efektivitas dan kinerja IMAJI setelah proses implementasi. Penilaian mencakup performa *model* CNN, keakuratan umpan balik AI, serta kesesuaian antara fitur-fitur aplikasi dengan tujuan penelitian. Hasil evaluasi ini juga dibandingkan dengan beberapa penelitian terdahulu sebagai dasar validasi kualitas sistem yang dikembangkan.

a. Evaluasi *model*

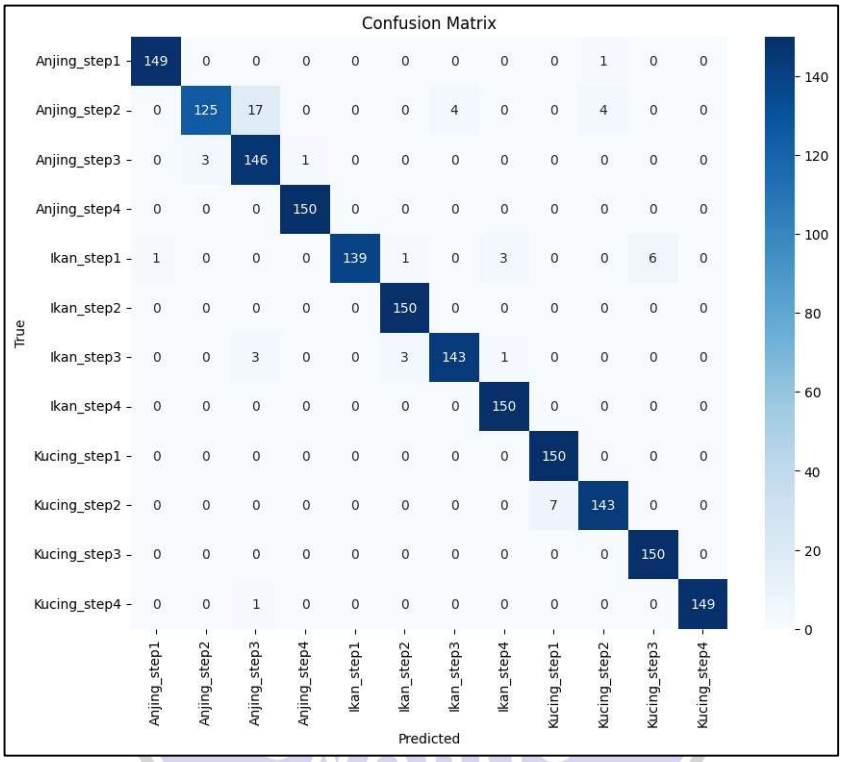
Evaluasi *model* dilakukan menggunakan data uji (test set) yang belum pernah dilihat oleh *model* selama proses pelatihan. Tujuannya adalah untuk menilai

seberapa baik *model MobileNetV2* mampu mengklasifikasikan gambar sketsa berdasarkan jenis objek (anjing, kucing, ikan) dan tahap menggambar (step 1–4). Penilaian dilakukan menggunakan beberapa metrik, yaitu akurasi, c*onfusion matrix,* dan *classification report.*

1. Akurasi *Model*:

Test Accuracy: 97%

Test Loss: 0.4847



Gambar 4.9 *Confusion Matrix* CNN *MobileNetV2*

Pada gambar 4.5, confusion matrix terlihat bahwa mayoritas prediksi berada pada diagonal utama, yang menunjukkan bahwa *model* mampu melakukan klasifikasi dengan akurasi tinggi. Beberapa kesalahan klasifikasi terjadi terutama pada kelas Anjing_step2 dan Ikan_step1, yang kemungkinan disebabkan oleh kemiripan visual antar tahap ketika sketsa mulai menyerupai bentuk akhir objek.

Classification Report:

	precision	recall	f1-score	support
Anjing_step1	0.99	0.99	0.99	150
Anjing_step2	0.98	0.83	0.90	150
Anjing_step3	0.87	0.97	0.92	150
Anjing_step4	0.99	1.00	1.00	150
Ikan_step1	1.00	0.93	0.96	150
Ikan_step2	0.97	1.00	0.99	150
Ikan_step3	0.97	0.95	0.96	150
Ikan_step4	0.97	1.00	0.99	150
Kucing_step1	0.96	1.00	0.98	150
Kucing_step2	0.97	0.95	0.96	150
Kucing_step3	0.96	1.00	0.98	150
Kucing_step4	1.00	0.99	1.00	150
accuracy			0.97	1800
macro avg	0.97	0.97	0.97	1800
weighted avg	0.97	0.97	0.97	1800

Gambar 4.10 Hasil *Classification Report*

Berdasarkan hasil *classification report* pada gambar 4.6, *model MobileNetV2* menunjukkan performa yang sangat baik dengan rata-rata nilai *precision, recall*, dan *f1-score* sebesar 0.97, baik pada perhitungan macro average maupun weighted average. Hal ini menunjukkan bahwa *model* tidak hanya akurat secara keseluruhan, tetapi juga seimbang dalam menangani setiap kelas. Hampir semua kelas memiliki nilai f1-score di atas 0.95, yang menandakan bahwa *model* mampu melakukan klasifikasi secara konsisten dan stabil terhadap berbagai variasi gambar sketsa pada data uji. Tingginya nilai metrik ini menjadi indikator kuat bahwa *model* memiliki kemampuan generalisasi yang baik, serta minim bias terhadap kelas tertentu.

Dengan performa yang tinggi dan stabil, *MobileNetV2* terbukti efektif sebagai arsitektur utama untuk klasifikasi gambar sketsa dalam aplikasi IMAJI. Selain itu, *model* ini memiliki keunggulan dari segi efisiensi ukuran dan kecepatan prediksi, sehingga cocok untuk diterapkan pada perangkat Android dengan format *model TensorFlow Lite (.tflite)*.

b. Evaluasi AI *Feedback*

Fitur AI *Feedback* dalam aplikasi IMAJI dirancang untuk memberikan saran otomatis kepada pengguna setelah menyelesaikan proses menggambar. Evaluasi dilakukan berdasarkan nilai Mean Squared Error (MSE) yang dihitung

antara gambar pengguna dengan *template* referensi. Semakin kecil nilai MSE, semakin besar tingkat kemiripan gambar.

Umpan balik diberikan melalui fungsi *showFeedbackBottomSheet(),* yang membagi hasil evaluasi menjadi tiga kategori yakni, nilai mse ketika dibawah 0.02 akan Menampilkan *Feedback* "Hebat! Gambar Anda sangat mirip contoh.", kemudian jika jika nilai mse dibawah 0.05 *Feedback* yang diberikan adalah "Proporsi sudah baik, bisa sedikit dirapikan."dan kondisi terakhir ketika mse bernilai lebih dari sama dengan 0.05 maka *Feedback*-nya "Perhatikan bentuk dan garis agar lebih sesuai contoh.".

Evaluasi ini menunjukkan bahwa sistem dapat memberikan saran koreksi yang relevan dan mudah dipahami, meskipun dalam bentuk teks. Berdasarkan hasil pengujian pada sejumlah data uji, sistem mampu membedakan tingkat kemiripan gambar dengan cukup akurat, sehingga *Feedback* yang diberikan sesuai dengan kualitas gambar pengguna. Walaupun belum menampilkan highlight visual pada kesalahan gambar, pendekatan berbasis teks ini cukup efektif dalam mendukung proses pembelajaran menggambar.

c. Evaluasi Aplikasi

Evaluasi aplikasi IMAJI dilakukan dari beberapa aspek, yaitu fungsionalitas, kecepatan respons, stabilitas sistem, dan kecocokan fitur terhadap tujuan pembelajaran sketsa. Berdasarkan hasil uji coba awal, aplikasi berhasil menjalankan seluruh proses inti dengan baik, mulai dari pengambilan gambar, pemrosesan melalui *model* CNN *MobileNetV2*, hingga penampilan hasil klasifikasi dan pemberian umpan balik secara otomatis.

Namun, dalam implementasi fitur *Augmented Reality* (AR), ditemukan kendala pada stabilitas posisi gambar *overlay*. Saat pengguna mulai menggambar, posisi *template* sering mengalami pergeseran yang disebabkan oleh tracking AR yang tidak konsisten, terutama ketika kamera bergerak, pencahayaan berubah, atau perangkat digunakan dalam posisi tidak stabil. Kemungkinan besar kendala ini diperparah oleh keterbatasan kompatibilitas perangkat, terutama pada smartphone dengan dukungan sensor atau spesifikasi AR yang rendah.

Hal ini terjadi karena sistem belum menerapkan penanda tetap (*anchor*) sebagai pengunci posisi objek dalam ruang AR, sehingga posisi *template* menjadi mudah bergeser. Sebagai solusi terhadap permasalahan ini, aplikasi menyediakan opsi mode manual (*non*-AR) sebagai alternatif. Dalam mode ini, *template* gambar ditampilkan secara statis di layar tanpa elemen AR, sehingga pengguna tetap dapat mengikuti panduan gambar dengan nyaman.

Meskipun mode manual tidak seakurat AR secara spasial, fitur ini terbukti efektif dalam menjaga kelangsungan penggunaan aplikasi dan tetap mendukung tujuan pembelajaran visual yang interaktif, terutama pada perangkat yang tidak mendukung AR secara optimal.

4.2.3 Pengujian Sistem

Pengujian dilakukan dengan pendekatan *white-box testing*, yaitu menelusuri dan menguji langsung struktur internal dari sistem aplikasi IMAJI, termasuk fungsi-fungsi kritis, percabangan, serta jalur eksekusi dalam kelas utama dalam aplikasi yakni CameraFragment.kt.

a. Panduan Gambar AR (*Overlay*) – *placeImageOnPlane()*

Fitur ini menampilkan gambar template di atas media gambar fisik dengan memanfaatkan teknologi Augmented Reality (AR). Fungsi *placeImageOnPlane()* digunakan untuk menempelkan gambar referensi secara presisi pada permukaan kertas yang telah diberi tanda silang di setiap sudutnya. Node gambar hanya ditampilkan jika tracking kamera aktif dan anchor valid, serta dikunci agar tidak berpindah setelah ditempatkan.

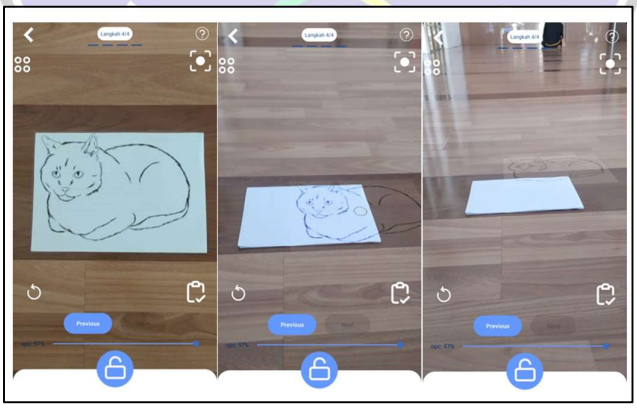
```
val anchor = hit.createAnchor()
val anchorNode = AnchorNode(anchor)
val node = TransformableNode(...).apply {
    this.renderable = renderable
    setParent(anchorNode)
}
```

Langkah pengujian dilakukan dengan menempatkan kertas pada bidang datar, kemudian menjalankan fitur AR dan mengarahkan kamera ke kertas dari sudut pandang yang bervariasi, yaitu 90°, 60°, 45°, dan 30°. Jarak antara kamera dan kertas diatur pada 30 cm, 45 cm, dan 70 cm. Tingkat pencahayaan diukur

menggunakan aplikasi lux meter dengan variasi 50 lux, 500 lux, dan 1000 lux. Selama pengujian, diamati apakah overlay tetap stabil atau mengalami pergeseran. Setiap kondisi pergeseran dicatat secara detail, termasuk besar pergeseran dan faktor penyebabnya, seperti perubahan sudut, jarak, maupun pencahayaan. Berikut tabel pengujiannya:

Tabel 4.2 Pengujian fitur Overlay

No	Sudut (°)	Jarak (cm)	Pencahayaan (lux)	Hasil	Keterangan
1	90°	30 cm	500	Stabil	Overlay presisi dan terkunci
2	60°	45 cm	1000	Stabil	Overlay presisi dan terkunci
3	60°	30 cm	100	Geser 5 cm ke kanan	Toleransi sudut mulai berkurang
4	45°	50 cm	500	Geser 13 cm ke kanan	Anchor kehilangan tracking
5	30°	70 cm	500	Geser total	Pencahayaan rendah, tracking gagal



Gambar 4.11 Pengujian *AR-overlay*

Berdasarkan hasil pengujian, juga dapat dilihat pada gambar 4.11, stabilitas overlay AR terbukti dipengaruhi oleh sudut pandang, jarak kamera, dan tingkat pencahayaan. Overlay tetap presisi dan terkunci pada sudut 90° dengan jarak 30 cm pada pencahayaan 500 lux, serta pada sudut 60° dengan jarak 45 cm pada pencahayaan 1000 lux. Namun, pada sudut 60° dengan jarak 30 cm dan pencahayaan 100 lux, overlay mulai bergeser sekitar 5 cm ke kanan, menandakan toleransi sudut berkurang pada cahaya rendah. Kondisi memburuk pada sudut 45° dengan jarak 50 cm dan pencahayaan 500 lux, di mana overlay bergeser 13 cm akibat anchor kehilangan tracking. Pada sudut 30° dengan jarak 70 cm dan pencahayaan hanya 500 lux, overlay gagal sepenuhnya karena jarak yang terlalu jauh.

b. Inferensi *Model* CNN – *runInference()*

Pengujian dilakukan untuk menilai kemampuan model CNN dalam mengenali gambar sketsa pengguna dengan tingkat kepercayaan (*confidence*) yang akurat. Proses inferensi dijalankan menggunakan fungsi *runInference()*, di mana gambar diubah menjadi format *ByteBuffer*, diproses oleh model CNN *MobileNetV2* berbasis *TensorFlow Lite*, lalu menghasilkan label prediksi beserta nilai confidence. Berikut potongan kode pada sistem:

```
val inputBuffer =
convertBitmapToByteBuffer(inputBitmap,
inputSize)

val result = Array(1) {
FloatArray(labelList.size) }
tflite.run(inputBuffer, result)

val maxIdx = result[0].indices.maxByOrNull {
result[0][it] } ?: -1
val predictedLabel = if (maxIdx != -1)
labelList[maxIdx] else ""
val confidence = result[0][maxIdx]
```

Fungsi ini digunakan untuk menjalankan proses inferensi *model* CNN berbasis *TensorFlow Lite* pada gambar sketsa yang telah dibuat oleh pengguna.

Gambar diubah menjadi format *ByteBuffer*, kemudian diproses oleh *model* untuk menghasilkan output berupa *array* skor prediksi (*logits*). Nilai tertinggi dalam array dipilih sebagai label prediksi dan disimpan dalam variabel *predictedLabel*, sedangkan *confidence score* disimpan di *confidence*.

Model menggunakan fungsi aktivasi *Softmax* untuk mengubah logit menjadi probabilitas, sebagaimana dirumuskan di rumus 2.2 dalam rumus:

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=i}^{K} e^{z_j}}$$

Contoh Penghitungan:

Diketahui,

z = [1.2, 0.5, 2.1],

nilai konstanta e = 2.718281828,

K= 3

Penyelesaian:

Karena K = 3, maka, $\sum_{j=i}^{K} e^{z_j} = e^{1.2} + e^{0.5} + e^{2.1}$

$e^{1.2}$=3.3201 $= 3.3201 + 1.6487 + 8.1662$

$e^{0.5}$=1.6487 $= 13.1349$

$e^{2.1}$=8.1662

$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=i}^{K} e^{z_j}}$, Karena ada 3 kelas maka,

1. Untuk kelas ke-1 (z_0):

$$\sigma(z_0) = \frac{e^{z_i}}{13.1349} = \frac{3.3201}{13.1349} = 0.2528$$

2. Untuk kelas ke-2 (z_1)

$$\sigma(z_0) = \frac{e^{z_i}}{13.1349} = \frac{1.6487}{13.1349} = 0.1256$$

3. Untuk kelas ke-3 (z_2)

$$\sigma(z_0) = \frac{e^{z_i}}{13.1349} = \frac{8.1662}{13.1349} = 0.6218$$

Tabel 4.3 Hasil contoh Prediksi 3 kelas

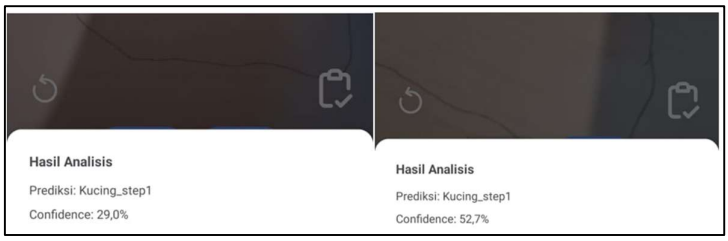
Kelas	Nilai Logit	e^z	Softmax (σ)	Probabilitas (%)
z_0	1.2	3.3201	0.2528	25.28%
z_1	0.5	1.6487	0.1256	12.56%
z_2	**2.1**	**8.1662**	**0.6216**	**62.16%**

Model memprediksi kelas ke-3 (z_2) sebagai hasil akhir dengan confidence score sebesar **62.16%.**

Pengujian inferensi model CNN dilakukan dengan menyiapkan tiga template sketsa dari dataset, kemudian masing-masing template digambar ulang menggunakan tinta hitam, tinta biru, dan pensil. Setiap hasil gambar difoto pada jarak kamera 30 cm dengan sudut 90° dan pencahayaan 500 lux, lalu diproses melalui fungsi *runInference()* untuk memperoleh nilai *confidence*. Pengujian dilanjutkan dengan kondisi pencahayaan berbeda, yaitu 200 lux dan 800 lux, guna membandingkan kinerja model pada lingkungan redup dan terang. Seluruh nilai confidence dan hasil prediksi dicatat untuk dianalisis.

Tabel 4.4 Hasil pengujian Inferensi

No	Warna Media Gambar	Pencahayaan (lux)	Confidence (%)	Hasil Prediksi	Keterangan
1	Hitam	500	52.7	Benar	Model mengenali dengan sangat baik
2	Hitam	200	29.0	Benar	Sedikit penurunan confidence pada cahaya redup
3	Biru	500	52.1	Salah	Model kurang optimal untuk tinta selain hitam
4	Biru	800	48.4	Salah	Warna di luar distribusi dataset memengaruhi prediksi
5	Pensil	500	46.9	Benar	Masih akurat untuk pensil dengan kontras tinggi



Gambar 4.12 Hasil Analisa gambar pengguna

Berdasarkan hasil pengujian, model CNN MobileNetV2 menunjukkan kinerja optimal dalam mengenali gambar tinta hitam dan pensil dengan kontras tinggi, dengan nilai confidence konsisten di atas 46% dan akurasi prediksi tetap benar meskipun pencahayaan turun hingga 200 lux (penurunan ±7%). Namun, performa menurun signifikan pada media gambar dengan tinta berwarna seperti biru, di mana nilai confidence tetap di kisaran 48–52% tetapi prediksi menjadi salah. Hal ini terjadi karena dataset pelatihan hanya mencakup tinta hitam dan pensil, sehingga distribusi warna yang terbatas membatasi kemampuan generalisasi model terhadap variasi media gambar di luar data latih.

c. Evaluasi Kemiripan (MSE) – *calculateMSE(img1, img2)*

Pengujian fitur evaluasi kemiripan dilakukan untuk mengukur seberapa mirip hasil gambar pengguna dengan template menggunakan metode *Mean Squared Error* (MSE)**.** MSE menghitung rata-rata selisih kuadrat antara nilai piksel gambar pengguna dan template. Semakin kecil nilai MSE, semakin tinggi tingkat kemiripan. Berikut potongan kode dari sistem:

```
val r1 = (pixels1[i] shr 16) and 0xFF
val r2 = (pixels2[i] shr 16) and 0xFF
...
sum += (r1 - r2)^2 + (g1 - g2)^2 + (b1 - b2)^2

return sum / (size * 3) / 255.0
```

Fungsi ini membandingkan dua gambar berdasarkan perbedaan nilai RGB dari masing-masing piksel. Total selisih kuadrat dihitung lalu dinormalisasi ke

rentang 0–1. Hasilnya dipakai untuk menilai kemiripan gambar sketsa dengan *template*. Perhitungan dilakukan dengan menggunakan rumus 2.1 berikut:

$$MSE = \frac{1}{n.3}\sum_{i=1}^{n}[(Ri - R'i) + (Gi - G'i) + (Bi - B'i)^2] \div 225$$

Contoh Penghitungan:

Diberikan tiga piksel perbandingan:

Tabel 4.5 Contoh nilai piksel

Piksel	R	G	B	R'	G'	B'
P1	100	120	140	102	118	138
P2	200	180	160	198	179	161
P3	50	60	70	48	59	69

Penyelesaian:

$$MSE = \frac{1}{n.3}\sum_{i=1}^{n}[(Ri - R'i) + (Gi - G'i) + (Bi - B'i)^2] \div 225$$

$$= (Ri - R'i) + (Gi - G'i) + (Bi - B'i)^2$$

$= (2^2+2^2+2^2)+(2^2+1^2+1^2)+(2^2+1^2+1^2)$

$= 4+4+4+4+1+1+4+1+1 = 24$

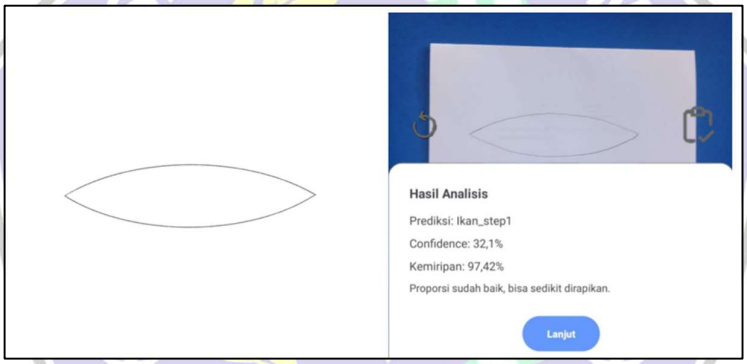
$$MSE = \frac{24}{3.3.225} = \frac{24}{2025} = 0.0119$$

Nilai mse = 0.0119, yang artinya gambar dari pengguna sangat mirip dengan panduan.

Pengujian fitur ini dilakukan dengan membandingkan hasil gambar pengguna dengan template acuan. Gambar dibuat menggunakan tinta hitam, biru, atau pensil, difoto pada jarak ±30 cm, dan diuji pada pencahayaan 200, 500, dan 800 lux. Fungsi *calculateMSE*(img1, img2) menghitung selisih nilai RGB setiap piksel, di mana nilai MSE yang lebih kecil menunjukkan kemiripan lebih tinggi. Hasil MSE digunakan sebagai dasar pemberian umpan balik otomatis.

Tabel 4.6 Hasil pengujian MSE

No	Nilai MSE	Pencahayaan (lux)	Hasil	Keterangan
1	0.0148	500	Sangat Mirip	Garis sesuai template
2	0.0234	200	Cukup Baik	Sedikit meleset pada detail
3	0.0456	800	Perlu Perbaikan	Beberapa proporsi meleset
4	0.0567	500	Tidak Mirip	Banyak garis tidak sesuai
5	0.0198	600	Sangat Mirip	Hampir identik dengan template



Gambar 4.13 Perbandingan Template dengan Hasil

Berdasarkan hasil pengujian MSE, terlihat bahwa tingkat pencahayaan dan ketelitian pengguna memengaruhi nilai kemiripan gambar secara signifikan. Pada pencahayaan optimal (500–600 lux), sistem mampu menghasilkan nilai MSE sangat rendah (≤ 0.02) yang menunjukkan kemiripan tinggi. Namun, pada pencahayaan redup (200 lux) atau terlalu terang (800 lux), nilai MSE cenderung meningkat sehingga kategori turun menjadi "cukup baik" atau "perlu perbaikan". Selain itu, MSE hanya memberikan gambaran global perbedaan piksel tanpa mempertimbangkan lokasi kesalahan, sehingga meskipun nilai tinggi, tidak terlihat secara langsung bagian gambar yang keliru. Keterbatasan ini membuat penilaian kurang detail dalam mengarahkan pengguna untuk memperbaiki hasil gambar

d. Saran Koreksi Sketsa Otomatis – *showFeedbackBottomSheet()*

Fitur ini memberikan umpan balik otomatis kepada pengguna berdasarkan nilai Mean Squared Error (MSE) dan confidence dari model CNN MobileNetV2. Nilai MSE digunakan untuk mengukur tingkat kemiripan gambar dengan template, sedangkan confidence menunjukkan tingkat keyakinan CNN dalam mengenali gambar. Kombinasi kedua nilai tersebut dipetakan ke dalam tiga kategori:

1. "Hebat! Gambar Anda sangat mirip contoh." - sangat mirip,
2. "Proporsi sudah baik, bisa sedikit dirapikan." - cukup baik,
3. "Perhatikan bentuk dan garis agar lebih sesuai contoh."- perlu perbaikan.

Berikut implementasi kode pada sistem:

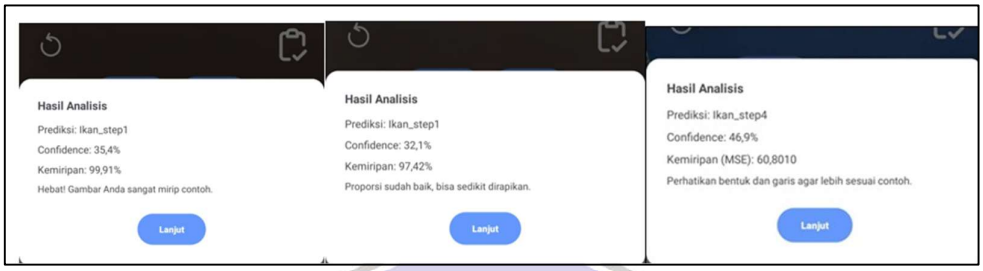
```
val Feedback = when {
    mse < 0.02 -> "Hebat! Gambar Anda sangat mirip contoh."
    mse < 0.05 -> "Proporsi sudah baik, bisa sedikit dirapikan."
    else -> "Perhatikan bentuk dan garis agar lebih sesuai
contoh."
}
```

Pengujian dilakukan dengan menyiapkan template gambar sebagai acuan, meminta pengguna menggambar ulang menggunakan tinta hitam, biru, atau pensil, lalu memotretnya dari jarak ±30 cm pada variasi pencahayaan 200, 500, dan 800 lux. Sistem kemudian menghitung nilai MSE menggunakan *calculateMSE*(img1, img2) serta mengambil nilai *confidence* dari CNN untuk menentukan kategori umpan balik yang ditampilkan pada *Bottom Sheet*, yang selanjutnya diverifikasi kesesuaiannya dengan kondisi gambar sebenarnya.

Tabel 4.7 Hasil Pengujian AI Feedback

No	Nilai MSE	Confidence (%)	Kategori Feedback	Deskripsi Sistem
1	0.0148	35.4	Sangat Mirip	Garis sesuai template, hasil optimal.
2	0.0234	32.1	Cukup Baik	Detail sedikit meleset, namun proporsi utama benar.
3	0.0456	46.9	Perlu Perbaikan	Banyak proporsi dan garis yang tidak sesuai.

No	Nilai MSE	Confidence (%)	Kategori Feedback	Deskripsi Sistem
4	0.0198	82.1	Sangat Mirip	Hampir identik dengan template, perbedaan minimal.



Gambar 4.14 Hasil fitur AI feedback

Pengujian pada fitur Saran Koreksi Sketsa Otomatis menunjukkan bahwa kombinasi nilai MSE dan confidence model CNN dapat menghasilkan kategori umpan balik yang sesuai, yaitu *sangat mirip*, *cukup baik*, dan *perlu perbaikan*. Hasil menunjukkan bahwa nilai MSE yang rendah cenderung diikuti confidence yang tinggi, menghasilkan prediksi *sangat mirip*, sedangkan nilai MSE yang tinggi dengan confidence moderat atau rendah menghasilkan kategori *perlu perbaikan*. Meskipun demikian, akurasi penentuan kategori masih bergantung pada kualitas pencahayaan dan posisi kamera saat pengambilan gambar.

e. *Mode Manual - switchToManualOverlay()*

Fitur mode manual pada aplikasi IMAJI dirancang untuk memberikan fleksibilitas kepada pengguna dalam mengatur posisi, ukuran, dan rotasi gambar panduan tanpa bantuan teknologi AR. Pengujian dilakukan untuk memastikan bahwa setiap interaksi yang diberikan pengguna dapat direspons dengan baik oleh sistem, sehingga proses pembelajaran tetap berjalan meskipun AR tidak digunakan. Berikut potongan kode yang mengimplementasikan fungsi *switchToManualOverlay()* dalam sistem:

```
private fun switchToManualOverlay() {
    binding.transformOverlay.visibility = View.VISIBLE
    binding.transformOverlay.setLocked(false)

    val imageUrl = getImageUrl(templateName, currentStep)
```

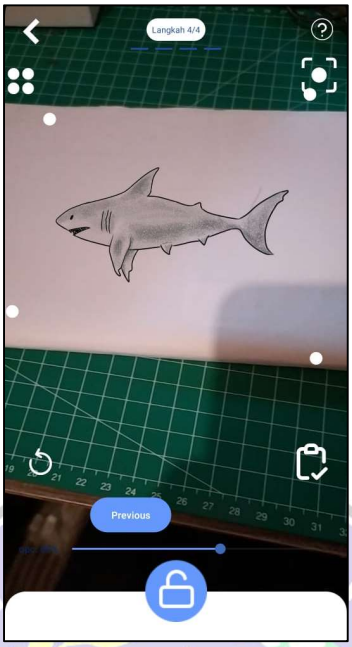
```
        Glide.with(requireContext())
            .asBitmap()
            .load(imageUrl)
            .into(binding.transformOverlay)

        binding.btnSwitchToManual.visibility = View.GONE
    }
```

Proses pengujian dilakukan dengan mengaktifkan mode manual melalui tombol navigasi, kemudian mencoba beberapa aksi seperti menggeser gambar panduan ke arah horizontal dan vertikal, memperbesar dan memperkecil (zoom in/zoom out), serta melakukan rotasi. Setiap aksi diamati untuk menilai sejauh mana sistem merespons input pengguna. Hasil pengujian interaksi pengguna dengan sistem ditunjukkan pada Tabel 4.8 berikut:

Tabel 4. 8 Hasil Pengujian Mode Manual

No	Aksi Pengguna	Respons Sistem	Keterangan
1	Geser gambar ke kiri/kanan	Berhasil	Gerakan responsif dan akurat
2	Geser gambar ke atas/bawah	Berhasil	Tidak ada delay pergerakan
3	Zoom in/zoom out	Berhasil	Skala proporsional
4	Rotasi gambar	Berhasil	Rotasi lancar tanpa distorsi
5	Navigasi ke step berikutnya	Gagal	Fitur hanya menampilkan 1 step



Gambar 4.15 Implementasi Mode Manual

Dari hasil pengujian tersebut, dapat disimpulkan bahwa fitur mode manual berfungsi dengan baik dalam penyesuaian posisi, ukuran, dan rotasi gambar panduan. Seluruh aksi utama berjalan lancar dan responsif. Namun, masih terdapat keterbatasan yaitu fitur belum mendukung navigasi antar langkah (*step*), sehingga pengguna hanya dapat menampilkan satu gambar panduan dalam mode ini.

f. Simpan dan *Upload* Hasil – *finishDrawing()* dan *Retrofit* API

Fitur ini berfungsi untuk mengunggah hasil gambar pengguna ke server Supabase melalui API *Retrofit*. Proses ini memastikan setiap hasil karya tersimpan secara terpusat di server, sehingga dapat diakses kembali untuk evaluasi maupun dokumentasi. Integrasi dengan Supabase juga memudahkan pengelolaan dan pencatatan data hasil gambar secara terstruktur. Berikut implementasi kode pada sistem:

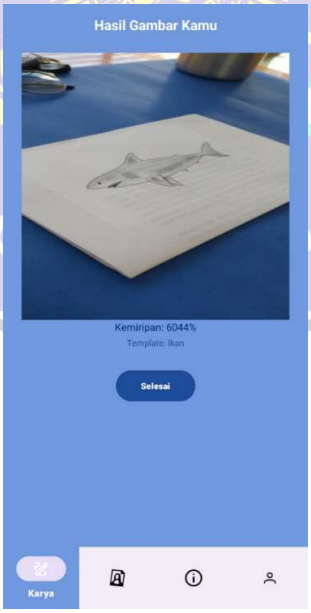
```
val imageUrl = uploadImageToSupabase(bitmap, userId)
...
val result = LatihanResult(...)
private fun finishDrawing(originalBitmap: Bitmap, mse:
Double)...
```

```
val response =
RetrofitClient.api.saveLatihanResult(listOf(result))
```

Pengujian dilakukan dengan menyelesaikan gambar di aplikasi, lalu menekan tombol "Simpan & *Upload*". Sistem diuji pada berbagai kondisi jaringan, seperti koneksi Wi-Fi stabil dan koneksi seluler yang tidak stabil. Setelah proses selesai, file di Supabase diperiksa untuk memastikan unggahan berhasil. Hasil pengujian menunjukkan bahwa pada koneksi stabil proses berjalan lancar, sedangkan pada koneksi tidak stabil, unggahan dapat tertunda atau gagal sehingga memerlukan mekanisme retry.

Tabel 4. 9 Hasil Pengujian fitur *Upload* Hasil

No	Kondisi Jaringan	Status Upload	Keterangan
1	Wi-Fi stabil (50 Mbps)	Berhasil	Proses cepat dan tanpa error
2	Seluler stabil (10 Mbps)	Berhasil	Kecepatan upload normal
3	Seluler tidak stabil	Gagal	Upload gagal, perlu retry
4	Tanpa koneksi internet	Gagal	gagal upload



Gambar 4.16 Hasil gambar dan upload gambar pengguna

Fitur simpan dan upload bekerja dengan baik pada jaringan stabil, namun pada jaringan tidak stabil atau tanpa koneksi internet proses upload gagal meskipun penyimpanan lokal tetap berhasil. Diperlukan fitur retry otomatis agar unggahan dapat dilanjutkan saat koneksi membaik.

g. *Upload Template - uploadTemplateToSupabase()*

Fitur ini memungkinkan admin menambahkan template gambar baru ke server Supabase tanpa perlu memperbarui aplikasi. Dengan begitu, konten latihan dapat diperbarui secara fleksibel dan langsung tersedia bagi semua pengguna. Proses ini dilakukan melalui fungsi uploadTemplateToSupabase(), yang mengunggah file template ke bucket penyimpanan Supabase dengan struktur direktori yang telah ditentukan. Berikut kode implementasi pada sistem:

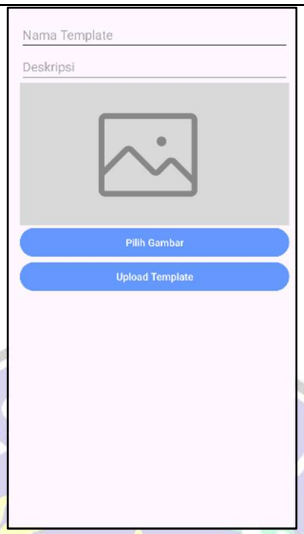
```
val imageUrl = uploadTemplateToSupabase(bitmap, templateName)
...
val template = TemplateUploadRequest(
    name = templateName,
    category = category,
    imageUrl = imageUrl,
    steps = steps
)
val response = RetrofitClient.api.uploadTemplate(template)
```

Pengujian dilakukan dengan memilih file template dari perangkat admin, menjalankan fungsi unggah, lalu memverifikasi bahwa file berhasil tersimpan di Supabase pada path yang benar. Setelah itu, aplikasi pengguna diperiksa untuk memastikan template baru muncul di daftar latihan. Hasil pengujian menunjukkan bahwa proses berjalan lancar pada koneksi internet stabil, sementara pada koneksi lambat proses unggah memerlukan waktu lebih lama namun tetap berhasil.

Tabel 4. 10 Hasil Pengujian *Upload Template*

No	Jenis File	Ukuran File	Status Upload	Keterangan
1	PNG	120 KB	Berhasil	Template langsung muncul di aplikasi
2	JPG	500 KB	Berhasil	Tidak ada penurunan kualitas visual
3	PNG	2 MB	Berhasil	Waktu upload lebih lama ±5 detik

No	Jenis File	Ukuran File	Status Upload	Keterangan
4	PNG	5 MB	Gagal	Melebihi batas ukuran file Supabase



Gambar 4.17 Upload *template* baru (*Admin*)

Fitur ini berjalan dengan baik untuk file berukuran kecil hingga menengah. Namun, ukuran file yang terlalu besar (lebih dari batas Supabase) menyebabkan kegagalan upload. Optimalisasi kompresi file sebelum upload diperlukan untuk menghindari masalah ini.

h. Hasil Pengujian fitur

Pada Tabel 4.11, dijabarkan Hasil Uji berdasarkan skenario pengujian fungsionalitas sistem IMAJI yang sebelumnya telah dilakukan.

Tabel 4.11 Hasil pengujian Sistem

No.	Fitur yang Diuji	Jumlah Uji Coba	Jumlah Berhasil	Jumlah Gagal	Keterangan
1	Panduan Gambar AR (*Overlay*)	5	2	3	*overlay* tampil stabil pada bidang kertas
2	Inferensi *Model* CNN	5	3	2	2 gambar tidak dikenali dengan benar oleh *model*

3	Evaluasi Kemiripan (MSE)	5	5	0	Nilai MSE berhasil dihitung dan akurat pada setiap pasangan gambar
4	Saran Koreksi Sketsa Otomatis	4	4	0	*Feedback* muncul sesuai rentang nilai MSE
5	Mode Manual *Overlay*	5	4	1	*Template* muncul dan dapat diatur posisi secara manual
6	Simpan dan Upload Hasil	4	2	2	Data hasil gambar berhasil diunggah ke *Supabase* (*storage* dan *database*)
7	*Upload Template* (*Admin*)	4	3	1	*Admin* berhasil menambahkan *template* baru
Total		**32**	**23**	**9**	

Penghitungan akurasi sistem:

$$Akurasi\ Sistem = \frac{23}{32} \times 100\% = 71.9\%$$

Berdasarkan pengujian terhadap 32 skenario uji coba, sebanyak 23 skenario berhasil dan 9 mengalami kegagalan atau ketidaksesuaian output. Fitur yang paling memerlukan perbaikan adalah Panduan Gambar AR (Overlay) yang sensitif terhadap sudut pandang dan pencahayaan, serta Saran Koreksi Sketsa Otomatis yang kadang memberikan kategori tidak sesuai dengan nilai MSE yang dihitung. Dengan akurasi keseluruhan 71.9%, sistem IMAJI dinilai cukup efektif, meskipun masih ada ruang perbaikan pada stabilitas AR dan konsistensi evaluasi otomatis.