PENGEMBANGAN APLIKASI ANDROID UNTUK KLASIFIKASI GAMBAR ANIME BERBASIS *DEEP LEARNING* DALAM MEMBEDAKAN GAMBAR MANUAL BUATAN ILUSTRATOR DENGAN GAMBAR BUATAN AI

Varraz Hazzandra A, 13521020[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia

***Abstract*—Kemampuan *artificial intelligence* (AI) dalam menghasilkan ilustrasi anime menimbulkan kekhawatiran hak cipta bagi ilustrator. Penelitian sebelumnya telah mengembangkan model deteksi, tetapi belum ada implementasi praktis berbasis Android. Penelitian ini mengembangkan aplikasi Android untuk klasifikasi gambar anime menggunakan model MobileNetV3Large. Model dilatih dengan 5597 gambar dengan pembagian data untuk pelatihan, validasi, dan pengujian. Untuk optimasi pada perangkat android, kuantisasi model dilakukan dan hasilnya berupa format INT8 yang unggul dengan akurasi 90.18 persen dan ukuran file 6.94 MB. Model tersebut membutuhkan waktu inferensi 0.0197 detik. Ketidakseimbangan dataset pada kategori tertentu masih menjadi kendala yang memerlukan augmentasi data. validasi data luar domain, fitur visualisasi area, serta mekanisme umpan balik pengguna menjadi saran pengembangan selanjutnya untuk peningkatan kualitas.**

Keywords—klasifikasi, gambar anime, MobileNet, artificial intelligence, ilustrator

I. PENDAHULUAN

Pesatnya perkembangan teknologi *Artificial Intelligence* (AI) melalui model seperti *Stable Diffusion*, *Illustrious*, dan *Pony* memungkinkan pembuatan gambar ilustrasi anime atau gambar anime yang menyerupai buatan illustrator secara manual. Kemampuan ini menimbulkan persoalan serius terkait hak cipta dan validasi keaslian karya bagi para ilustrator. Investigasi Nikkei Asia mengungkapkan 90 ribu gambar buatan AI yang meniru beberapa anime populer tersebar di berbagai platform [10]. Beberapa di antaranya bahkan sulit dibedakan dengan gambar anime aslinya. Sempat marak pula gambar buatan AI yang menyerupai hasil Studio Ghibli, studio anime legendaris dengan salah satu karya terkenanlnya, 'Spirited Away' [4]. Oleh karena itu, kebutuhan sistem klasifikasi gambar anime buatan AI dan bukan semakin mendesak.

Beberapa upaya klasifikasi gambar telah dilakukan, tetapi masih memiliki keterbatasan. Alat deteksi konten AI seperti UndetectableAI cenderung berfokus pada foto realistik dan kurang spesifik untuk gaya anime. Zhu dkk. mengembangkan model AniXPlore dengan akurasi tinggi, tetapi arsitekturnya terlalu kompleks untuk implementasi praktis pada perangkat seluler [14]. Johan dkk. berhasil mengimplementasikan model MobileNetV3Large dengan akurasi 94,16% pada platform *website* [5]. Namun, solusi berbasis *web* masih dinilai kurang praktis karena ketergantungan pada koneksi internet dan latensi jaringan untuk mengakses fitur solusinya.

Penelitian ini mengusulkan pengembangan aplikasi Android untuk klasifikasi gambar anime yang beroperasi secara luring (*offline*) untuk mengisi kesenjangan antara akurasi model dan praktikalitas penggunaan. Penelitian ini berfokus pada implementasi model *Convolutional Neural Network* (CNN) yang ringan dan efisien agar dapat diterapkan langsung pada perangkat (*on-device AI*). Pendekatan ini diharapkan mampu menjadi solusi validasi keaslian gambar anime yang akurat dan mudah digunakan tanpa bergantung pada infrastruktur server daring.

II. LANDASAN TEORI

A. *Anime*

Anime merupakan istilah umum karya animasi Jepang yang mencakup serial televisi, film, dan rilis video langsung [12]. Istilah ini lahir dari kata *'animeeshon'*, kata serapan dalam Bahasa Jepang dari '*animation'*. Ciri gambar anime yang membedakan dengan jenis gambar lainnya adalah gambar mengandung komponen-komponen dengan frekuensi tinggi yang lebih sedikit daripada jenis gambar lainnya. Komponen-komponen tersebut berupa tekstur yang kompleks, *stochastic noise*, dan informasi garis tepi pada frekuensi menengah hingga tinggi, terutama kontur garis. Gambar anime juga memiliki pencahayaan yang tidak realistis dan abstraksi geometris. Gambar anime terdiri dari beberapa objek yang mencuat secara semantik dengan batas yang jelas [14].

Gambar anime yang dihasilkan AI cenderung memiliki pencahayaan yang tidak biasa seperti pencahayaan gambar anime pada umumnya [11]. Selain itu, gradasi warna yang diterapkan pada gambar anime oleh AI sangat kompleks sehingga sulit diterapkan kebanyakan ilustrator. Gambar anime hasil AI juga memiliki artefak frekuensi yang tidak alami yang membuat gambar tersebut dapat dikenali sebagai buatan AI tanpa harus melihat detail yang lebih tajam pada gambar [14]

B. *Deep learning*

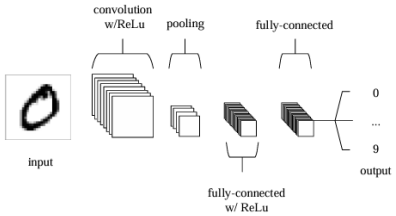
Deep learning atau pembelajaran mendalam merupakan metode pada model kecerdasan buatan yang mempelajari fungsi yang kompleks dengan kumpulan data yang besar untuk mengekstrak fitur tingkat tinggi secara otomatis melalui struktur jaringan saraf yang berlapis-lapis [2]. Model tersebut melakukan ekstraksi dengan menggunakan pembelajaran yang tidak diawasi untuk inisialisasi bobot yang saling terhubung. Inisialisasi tersebut diikuti pembelajaran yang diawasi dengan menggunakan informasi sinyal pembelajaran [6]. *Deep learning*

dibangun dari jaringan saraf buatan. Saat ini, *deep learning* merupakan salah satu area dalam *machine learning* [9].

C. Convolutional Neural Network

Convolutional *Neural Network* (CNN) atau jaringan saraf konvolusi adalah jenis *feedforward neural network* yang mampu mengekstrak fitur dari data dengan struktur konvolusi [8]. Sama halnya dengan ANN, CNN terdiri dari neuron-neuron yang mampu melakukan optimasi sendiri melalui pembelajaran yang mana tiap neuronnya menerima *input* dan melakukan operasi [11].

CNN terdiri dari tiga jenis lapisan: *convolutional layers* atau lapisan konvolusi*, pooling layers* atau lapisan penyatuan, dan *fully-connected layers* atau lapisan yang terhubung penuh.Lapisan konvolusi melakukan operasi konvolusi pada gambar sebagai input. Lapisan *pooling* menyederhanakan dimensi spasial input. Lapisan *fully-connected* menghasilkan nilai kelas untuk klasifikasi. Arstiktektur CNN dapat dilihat pada gambar II.1.



Gambar II.1 Arsitektur CNN (O'Shea dkk., 2015)

D. MobileNet

MobileNet merupakan salah satu arsitektur jaringan berbasis CNN yang dibangun khusus untuk kebutuhan *mobile* dan aplikasi *embedded vision* [3]. Arsitektur ini dibangun dengan ukuran yang kecil dan memungkinkan minimalisasi latensi yang menjadi hal krusial dalam penerapan pada media *mobile*. MobileNet merupakan hasil faktorisasi konvolusi standar menjadi konvolusi *depthwise* dan konvolusi *pointwise* yang berupa konvolusi 1x1. Konvolusi *depthwise* akan melakukan *filter* di tiap *input channel*. Lalu, hasil filter tiap kanal masukan tadi akan dikombinasi oleh konvolusi *pointwise*.

III. PENELITIAN TERKAIT

Zhu dkk. membangun sebuah model bernama AnimeDL-2M yang menghasilkan dataset gambar ilustrasi anime dalam jumlah besar [14]. Hal ini dilatari oleh sedikitnya dataset yang khusus berisi gambar ilustrasi anime. Namun, mereka juga mengajukan sebuah model bernama AniXplore, sebuah model baru yang dirancang untuk mendeteksi gaya anime pada suatu gambar. Model ini menerapkan *discrete wavelet transform* (DWT) untuk menangkap fitur *line-based,* fitur paling krusial dalam gambar gaya anime. Model ini juga menggunakan *dicsrete cosine transform* (DCT) yang menangkap struktur spasial global secara keseluruhan pada gambar anime untuk identifikasi manipulasi gambar di level objek. Model AniXplore sukses mencapai 100% dalam akurasi dan skor F1 di *image-level.* Namun, model masih terlalu kompleks diterapkan pada aplikasi *mobile.*

Johan dkk. melakukan penelitian untuk mengidentifikasi gambar anime yang dihasilkan AI [5]. Penelitian dilakukan terhadap tiga model bertipe CNN, yaitu MobileNetV2, MobileNetV3Large, dan InceptionV3. CRISP-DM digunakan sebagai metodologi penelitian. Pada tahap persiapan data, mereka melakukan *data scraping* dari situs Pixiv. Hasilnya adalah 2000 citra yang didistribusikan dengan jumlah yang sama antara label AI dan non-AI. MobileNet V3 Large mengungguli dua model lainnya. Model ini mencapai akurasi tertinggi sebesar 94,16%. Selain itu, MobileNet V3 Large hanya memakan 25 detik dalam waktu inferensi dengan ukuran model sebesar 13,57 MB. Model diaplikasikan pada *website* melalu streamlit sebagai media *deployment* sistem. Namun, *website* dinilai kurang praktis pada penggunaan sistem.

IV. ANALISIS SOLUSI

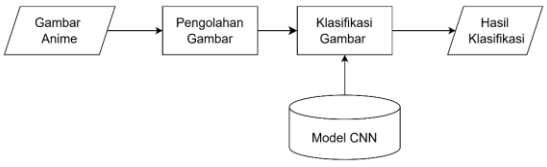
Terdapat beberapa poin yang menjadi isu berdasarkan penelitian-penelitian di atas. Kompleksitas model dan kepraktisan sistem saling terikat karena model yang kompleks menyebabkan implementasi tidak bisa dilakukan langsung pada aplikasi *mobile* dengan fitur luring.Fitur daring dinilai kurang praktis dari segi *load time* dan ketergantungan terhadap sinyal internet. Selain itu, keterbatasan jumlah dan variasi dataset menjadi tantangan karena hal tersebut membuat model rentan *overfitting*.

Dari poin-poin isu di atas, didapat solusi berupa pembangunan aplikasi android dengan model *pre-trained* yang dilatih ulang dengan dataset yang lebih banyak. Android dipilih sebagai *platform mobile* karena banyak digunakan dan praktis [13]. Pelatihan ulang dilakukan pada model *pre-trained* seperti yang dilakukan Johan dkk. [5].

V. RANCANGAN SOLUSI

Solusi yang akan dibangun memiliki alur kerja seperti di Gambar V.1. Solusi tersebut membutuhkan dua elemen, yaitu aplikasi android dan model *deep learning*. Dengan demikian, rancangan solusi terdiri dari dua bagian, yaitu pemodelan dan pengkodean aplikasi. Bagian pemodelan terdiri dari pemahaman dan pengumpulan data, pelatihan, dan pengujian. CRISP-DM digunakan sebagai acuan metodologi pada bagian ini.

Bagian pengembangan aplikasi android terdiri dari pembangunan antarmuka aplikasi android, integrasi dengan model hasil tahap *data mining*, dan pengujian aplikasi android. Hasil yang diharapkan dari rancangan solusi ini berupa aplikasi android yang dapat melakukan klasifikasi pada gambar ilustrasi anime.



Gambar V.1 Alur Kerja Solusi

VI. LINGKUNGAN IMPLEMENTASI

Berikut merupakan spesifikasi lingkungan yang digunakan.

1. Google Colab Pro (Pelatihan)
    a. GPU: NVIDIA T4
    b. System RAM: 12.7 GB
    c. GPU RAM: 15 GB
    d. Disk: 112.6 GB
2. Pustaka *Data Mining* (Pelatihan)
    a. Tensorflow: v2.19.0
    b. Keras (TF): v3.10.0
    c. Numpy: v2.0.2
    d. Pandas: v2.2.2
    e. Matplotlib: v3.10.0
    f. Seaborn: v0.13.2
    g. Sklearn: v1.6.1
    h. Python: v3.12.12
3. Laptop (Acer Swift314-510G)
    a. CPU: 2.40GHz
    b. RAM: 15.8 GB
    c. Sistem Operasi: 11th Gen Intel Core i5-1135G7 2.40GHz, 2419 Mhz, 4 Core, 8 Logical Processor
4. Ponsel (Vivo Y53s) (Pengujian)
    a. Processor: MediaTek Helio G80 (MT6769T)
    b. RAM: 8 GB
    c. Sistem Operasi: Funtouch OS 13
5. Pustaka Aplikasi (Pengembangan Aplikasi)
    a. Flutter SDK: 3.35.2
    b. Dart SDK: 3.9.0
    c. tflite_flutter: 0.12.1
    d. image: 4.5.4
    e. http: 1.5.0*
    f. file_picker: 8.3.7
    g. image_picker: 1.2.0

VII. PERSIAPAN DATA

Data dikumpulkan dari dataset yang digunakan Zhu dkk. dan Kaggle dengan dataset sudah terlabeli antara hasil AI dan ilustrator. Total gambar pada dataset yang akan digunakan adalah 5597 gambar. Pembagian dataset dilakukan dari awal dengan rasio 60:20:20 untuk data latih, validasi, dan uji [11]. Jumlah gambar per jenis data dapat dilihat pada Tabel VII.1.

Tabel VII.1

Jumlah Data Per Label

Jenis Data	Jumlah Data	
	AI	Ilustrator
Training	1678 (60%)	1680 (60%)
Validation	560 (20%)	559 (20%)
Testing	560 (20%)	560 (20%)
Total	2278	2279

Kemudian, augmentasi data dilakukan pada data latih untuk menambah variasi dan jumlah gambar. Augmentasi yang dilakukan berupa rotasi sebesar 15°, *random crop* sebesar 20%, translasi Afin pada tinggi dan lebar gambar sebesar 15%, mengaktifkan mode *horizontal flip*, *shear* sebesar ±10° dan mengaktifkan mode *nearest* pada *fill mode* untuk mengisi *pixel* kosong dari hasil augmentasi.

VIII. PELATIHAN

Model yang dilatih adalah MobileNetV3 Large. Pelatihan dilakukan terhadap total 3358 data uji dengan 1119 data validasi. Penelitian Johan dkk. menjadi acuan dalam melakukan pelatihan, termasuk *hyperparameter* dengan nilainya. Pada penelitian tersebut akurasi MobileNetV3 Large mencapai 94.16% sehingga pelatihan diharapkan mencapai akurasi paling sedikit 90% [5].

Pelatihan dilakukan dalam dua fase, yaitu fase pembekuan seluruh lapisan (*transfer learning*) dan fase pencairan sebagian lapisan model (*fine tuning*). Fase kedua mencairkan 100 lapisan pertama kecuali lapisan *BatchNormalization* pada MobileNetV3 Large. Hal ini dilakukan untuk melatih lapisan MobileNet yang dicairkan tersebut dengan dataset baru. Kedua fase melibatkan mekanisme *early stopping* dan *reduceLROnPlateau*. *Early stopping* akan menghentikan pelatihan sebelum mencapai *epoch* yang ditentukan. *ReduceLROnPlateau* mengurangi *learning rate* sampai batas *learning rate* minimum. Kedua mekanisme ini berguna untuk mencegah *overfitting* pada model dan meminimalisir penggunaan sumber daya waktu dalam pelatihan. *Hyperparameter* pelatihan model terdapat pada Tabel VIII.1.

Tabel VIII.1

Hyperparameter Training

Hyperparameter	**Variasi Nilai**
Image size	224
Channel	3
Batch size	64
Epoch	50, 50s
Dropout rate	0,4
Learning rate	5e-4, 1e-5
Momentum	0.9

Optimizer	AdamW
Loss Function	*Binary Crossentropy*
Activation Function	Gelu
Aktivasi *Output*	Sigmoid
Weight Decay	1e-2
Monitor	'val_*loss*'
Patience Early stopping	13
Patience ReduceLROnPlateau	4
Learning rate ReduceLROnPlateau	1e-6
Factor	0.2

Beberapa *hyperparameter* memiliki nilai yang berbeda dari penelitian yang dirujuk. Hal ini disebabkan oleh hasil pelatihan yang jauh di bawah hasil pengujian pada penelitian yang dirujuk, yaitu 84.54%. Perbedaan nilai *hyperparameter* terdapat pada batch size, *dropout rate*, dan *epoch*. *Batch size* diperkecil dari 128 menjadi 64 dengan harapan model akan lebih memelajari detail pada citra. *Epoch* ditambah dari 50 *epoch* menjadi 50 (dengan *early stopping*) dan 50 *epoch* untuk sesi *unfreeze*. *Dropout rate* ditambah dari 0.2 menjadi 0.4 untuk memastikan model tidak terlalu menghafal pola pada citra dengan banyak neuron yang aktif. *Patience* pada *early stopping* diubah dari 10 menjadi 13 untuk memperbanyak jumlah *epoch* yang diharapkan meningkatkan akurasi model. *Patience* pada *reduceLROnPlateau* juga ditambah dari 3 menjadi 4 dengan alasan serupa. Penambahan *layer* dilakukan dengan menambah satu *layer* dropout dengan *dropout rate* yang sama untuk mencegah *overfitting* pada model. Hasil pelatihan model terdapat pada Tabel VIII.2.

Tabel VIII.2

Hasil Pelatihan Tanpa Modifikasi (Johan dkk., 2025) dan Dengan Modifikasi

Metrik	Tanpa Modifikasi		Modifikasi	
	Best Saved (9)	*Last Epoch* (19)	*Best Saved* (74)	*Last Epoch* (79)
Akurasi (%)	85.17	88.42	91.44	94.28
Presisi (%)	84.70	89.19	90.49	94.03
Recall (%)	85.59	88.01	92.29	94.44
AUC (%)	92.89	95.88	97.76	98.60
Loss Validasi (%)	34.13	26.70	19.82	16.12
Akurasi Validasi (%)	84.54	82.57	91.51	91.24
Presisi Validasi (%)	83.98	73.35	92.49	93.41
Recall Validasi (%)	85.33	89.91	90.34	88.73
AUC Validasi (%)	91.22	91.97	96.79	96.78
Loss Validasi (%)	37.73	39.38	22.64	23.25

Perubahan *hyperparameter* dari penelitian Johan dkk. menunjukkan hasil sesuai harapan. Akurasi validasi model mencapai angka 91.51% [5]. Akurasi tersebut melampaui target minimal 90%. Pada gambar tersebut, terlihat bahwa pada sesi pertama, akurasi terbaik terdapat sebelum *epoch* ke-20 (*epoch* ke-16) dan pelatihan berhenti di *epoch* ke-29. Pada sesi kedua, akurasi terbaik didapat pada *epoch* ke-74 sebelum pelatihan akhirnya berhenti di *epoch* ke-79. Sesi pertama menunjukkan adanya potensi *overfitting* mengingat jarak yang jauh antara *epoch* dengan akurasi terbaik dengan *epoch* perhentian. Hal tersebut bisa terjadi karena *patience* yang tinggi pada mekanisme *early stopping*. Hal yang menarik adalah mekanisme tersebut tidak terjadi pada fase kedua. Hal ini dapat terjadi karena mekanisme pencairan beberapa *layer* pada MobileNetV3Large dengan tetap membiarkan lapisan *BatchNormalization* membeku. *Learning rate* juga diperkecil sehingga dapat menjadi poin perhatian. Namun, untuk memvalidasi penyebab tidak terjadinya *early stopping* pada fase kedua, dibutuhkan pengujian lebih lanjut, seperti dengan melatih model di fase pencairan dengan *learning rate* yang sama.

Pelatihan model menghasilkan bobot model yang berformat Keras (.keras). Aplikasi android hanya menerima model berformat TensorFlow Lite (.tflite). Oleh karena itu, model hasil pelatihan dikonversi menjadi format TensorFlow. Kemudian, model berformat TensorFlow diubah menjadi model berformat TensorFlow Lite. Kuantiasasi secara bersamaan diterapkan untuk menentukan presisi model. Variasi presisi tersebut berupa 32-bit (FP32), 16-bit (FP16), dan *integer* 8-bit (INT8). Presisi tersebut menentukan tipe data bobot pada model. Kuantisasi mengurangi jumlah bit untuk menyimpan model sehingga ukuran file model menjadi lebih kecil. Di sisi lain, kuantisasi juga mengurangi akurasi model.

IX. PENGEMBANGAN APLIKASI

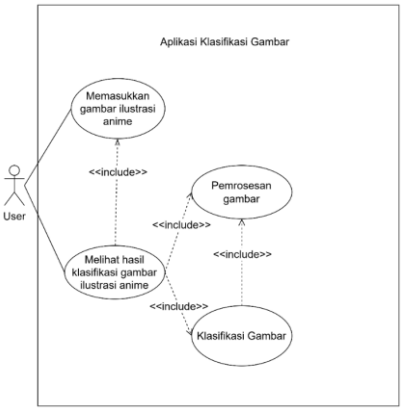
Aplikasi yang dibangun berupa *single page apps*. Laman utama aplikasi berisi fitur input gambar yang menyatu dengan penampil hasil klasifikasi. Hasil klasfikasi tidak disimpan sehingga pengguna harus memasukkan gambar saat ingin melakukan klasifikasi gambar. Kebutuhan fungsional aplikasi adalah pengguna dapat memasukkan gambar dari galeri dan pengguna dapat melihat hasil klasifikasi gambar. Kebutuhan nonfungsional gambar tertera pada Tabel IX.1. Gambar IX.1 menunjukkan diagram *use case* aplikasi klasifikasi anime dan Gambar IX.2 menunjukkan *sequence diagram* aplikasi.

Tabel IX.1

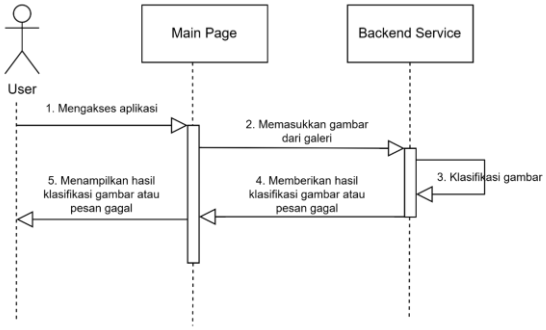
Kebutuhan Nonfungsional

Kebutuhan Non-fungsional	**Deskripsi**
Usability	Aplikasi memiliki tampilan dan alur penggunaan yang mudah dipahami

Performance	Aplikasi harus menampilkan hasil klasifikasi dalam waktu kurang dari tiga detik.
Compatibility	Aplikai berjalan lancar pada perangkat android dengan versi minimal android 7.
Security	Server aplikasi tidak menyimpan gambar *input* pengguna

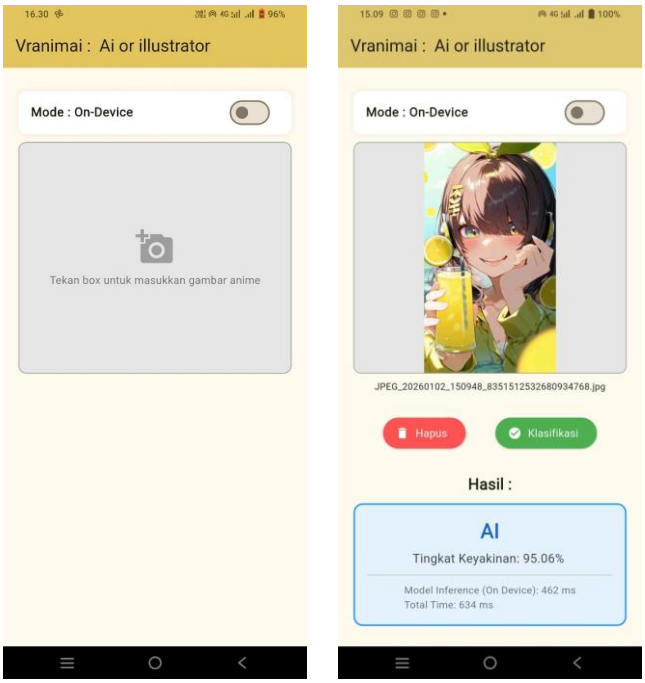


Gambar IX.1 *Use Case Diagram* Aplikasi Klasifikasi Gambar Anime



Gambar IX.2 *Sequence Diagram* Aplikasi

Aplikasi dibangun dengan arsitektur *on-device AI*. Dengan demikian, proses klasifikasi gambar yang merupakan *backbone* aplikasi dijalankan langsung di aplikasi. Namun, ukuran aplikasi akan bertambah karena memuat model langsung, tidak seperti implementasi aplikasi yang berbasis *client-server*. Untuk mencegah pertambahan ukuran file dan beban komputasi yang signiifikan, dilakukan konversi dan kuantisasi model. Aplikasi masih berupa purwarupa sehingga *deployment* dilakukan di lingkungan komputer pengguna. Gambar IX.3 menunjukkan desain antarmuka aplikasi android. Aplikasi android klasifikasi anime dikembangkan dengan ukuran 29.9 MB. Ukuran tersebut didominasi oleh pustaka yang diisi *native* Flutter (15.8 MB) dan TensorFlow Lite (6.5 MB). *Assets* menjadi direktori yang ditempati model tflite dengan ukuran 6.94 MB.



(a) (b)

Gambar IX.3 Tampilan Awal Aplikasi (a) dan Hasil Klasifikasi Gambar Anime (b)

X. PENGUJIAN

Pengujian terbagi menjadi dua bagian, yaitu pengujian terhadap aplikasi dan pengujian terhadap model. Tiap pengujian memiliki keterkaitan yang direpresentasikan dengan *traceability matrix* yang ditunjukkan oleh Tabel X.1. Kotak dengan tanda centang menunjukkan rumusan masalah sudah dipecahkan berdasarkan hasil pengujian pada tiap bagian.

Tabel X.1

Traceability Matrix Solusi

Rumusan Masalah / Pengujian	Model dapat diimplementasikan ke aplikasi android.	Aplikasi dapat melakukan klasifikasi pada gambar *input*.	Aplikasi mudah digunakan.
Model	✓	✓	
Aplikasi	✓	✓	✓

A. Pengujian Model

Pengujian model dilakukan untuk membandingkan kinerja model versi asli dan konversi yang terbagi menjadi INT8 dan FLOAT32. Model versi konversi dan kuantisasi akan digunakan pada aplikasi android sehingga perbandingan spesifik menyoroti kinerja kedua model tersebut. Model diuji dengan data latih

sejumlah 1120 gambar dengan 560 gambar AI dan 560 gambar nonAI. Metrik pengukuran yang digunakan adalah akurasi, presisi, recall, f1-score, AUC, waktu inferensi, dan ukuran Model (MB). Metrik pengukuran tersebut mengacu pada pengujian yang dilakukan Johan dkk. dalam penelitian dengan kasus yang sama [5]. Hasil pengujian tersebut ditunjukkan pada Tabel X.2.

Tabel X.2

Hasil Pengujian Model

Versi Model / Metrik Pengukuran	Basis	FLOAT32	INT8
Akurasi (%)	90.09	90.09	90.18
Presisi (%)	90.89	90.89	91.06
Recall (%)	89.11	89.11	89.11
F1-Score (%)	89.99	89.99	90.07
AUC (%)	96.95	96.95	96.94
Waktu Inferensi (s)	0.3095	0.0190	0.0197
Ukuran Model (MB)	39.84	13.80	6.94

Model basis mampu mencapai akurasi 90.09%. Akurasi ini memang masih di bawah hasil penelitian Zhu dkk. yang mencapai 100% dan Johan dkk. dengan akurasi 94.16% walau juga dengan data uji dengan jumlah gambar yang berbeda-beda, seperti 200 gambar uji per label oleh Johan dkk. [5] [14]. Model pada pelatihan dan pengujian ini memanfaatkan 5597 gambar, sedangkan model pada Johan dkk. memanfaatkan 2000 gambar dalam pelatihan dan pengujian [5]. Akurasi yang lebih rendah daripada akurasi yang dicapai model oleh Zhu dkk. diperkirakan disebabkan oleh perbedaan jumlah dan variasi dataset yang sangat signifikan yang mana mereka memanfaatkan satu juta lebih gambar yang mereka hasilkan dari dataset generator yang mereka kembangkan.

Model hasil pelatihan yang telah dikonversi ke versi INT8 menunjukkan akurasi yang mengungguli versi FLOAT32. Namun, pengujian McNemar belum diterapkan sebagaimana pengujian tersebut menjadi standar ilmiah dalam melihat signifikansi kualitas suatu model. Oleh karena itu, perbedaan performa antara kedua model belum bisa dianggap signifikan secara statistik ($p > 0,05$). Pemilihan hasil kuantisasi model dijustifikasi berdasarkan ukuran model. Model INT8 memiliki ukuran 50% lebih kecil daripada model FLOAT32. Peningkatan akurasi INT8 di luar teori umum bahwa penurunan presisi bit mengakibatkan penurunan akurasi. Namun, hal ini juga dapat terjadi akibat efek regularisasi yang terjadi saat pembulatan ke target presisi bit. Efek regularisasi dihasilkan *noise quantization*. Hal tersebut menimbulkan efek generalisasi yang lebih baik dalam mengekstrak fitur gambar [1]. Dengan demikian, model versi INT8 digunakan pada aplikasi klasifikasi gambar anime.

Meskipun model MobileNetV3Large dengan versi INT8 menunjukkan kinerja yang memuaskan pada pengujian dengan data uji yang telah disiapkan, ditemukan beberapa keterbatasan yang berpotensi memengaruhi generalisasi model pada skenario dunia nyata.

1. Ilustrasi Manga Hitam-Putih (Black & White)

   Pelatihan dan evaluasi model pada Tugas Akhir ini difokuskan pada gambar dengan banyak spektrum warna. Hal ini disebabkan oleh dataset yang didominasi oleh gambar bertipe tersebut Arsitektu model yang digunakan memanfaatkan fitur berbasis warna (*color-based features*) sebagai salah satu indikator pembeda sehingga diperkirakan akan terjadi penurunan akurasi yang signifikan ketika model dihadapkan pada gambar berwarna hitam-putih seperti gambar manga hitam-putih karena hilangnya informasi spektrum warna.

2. Sketsa dan *Line Art*

   Pelatihan model didominasi oleh gambar berjenis *finished artwork* (karya yang sudah selesai). Bias tersebut memunculkan prediksi bahwa model akan sulit mengklasifikasikan gambar yang masih dalam tahap sketsa kasar atau *line art* (*rakugaki*). Model pada Tugas Akhir dilatih dengan fitur tekstur dan *shading* sebagai fitur utama pada gambar.

3. Pola Geometris

   Gaya *chibi* dan gambar tanpa elemen karakter menjadi pola geometris yang sangat minim ditemukan pada dataset yang digunakan. Gaya *Chibi* memiliki proporsi anatomis yang berbeda jauh dari proporsi anime standar, misalnya rasio kepala terhadap tubuh yang ekstrem dan objek geometris yang minim (hanya satu objek saja dengan latar belakang monokrom). Dibutuhkan penambahan dataset, pelatihan, dan pengujian dengan jenis gambar tersebut untuk memastikan akurasi model yang digunakan

B. Pengujian Aplikasi

Pengujian dilakukan terhadap aplikasi klasifikasi gambar anime yang dibangun. Pengujian memastikan bahwa aplikasi dapat melakukan klasifikasi gambar anime memanfaatkan model yang telah diintegrasikan dengan aplikasi tersebut. Pengujian aplikasi klasifikasi gambar menerapkan pendekatan *black box testing* dengan teknik *state transition testing*. Aplikasi klasifikasi gambar tidak memiliki banyak input dan output serta hanya berisi *task* yang dapat dinilai kevalidannya dari perubahan *state* pada antarmuka. Pengujian sekaligus menguji kehandalan model *deep learning* yang digunakan aplikasi dalam melakukan klasifikasi gambar. Hasil pengujian aplikasi klasifikasi gambar anime dengan *black box testing* dapat dilihat pada Tabel X.3. Tabel X.4 menyajikan pengujian aplikasi dengan memanfaatkan model yang telah dilatih.

Tabel X.3

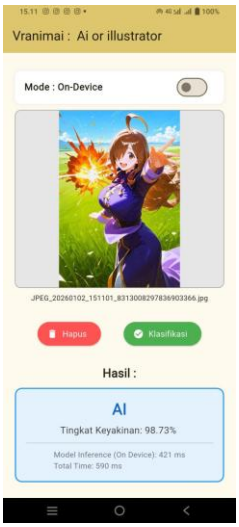
Pengujian Aplikasi Dengan *Blackbox Testing*

Task	Hasil Ideal	Hasil Pengujian	Kesimpulan
Input gambar melalui galeri dengan menekan box gambar.	Gambar dapat di-*input* melalui galeri	Gambar muncul di box *input* gambar, gambar dapat di-*input* melalui galeri	Valid
Input gambar melalui galeri dengan menekan tombol *input* gambar.	Gambar dapat di-*input* melalui galeri	Gambar muncul di box *input* gambar, gambar dapat di-*input* melalui galeri	Valid
Klik tombol penghapus gambar *input*.	Gambar *input* terhapus dari *box preview*	Gambar *input* terhapus dari *box preview*	Valid
Klik tombol klasifikasi gambar	Hasil klasifikasi muncul.	Hasil klasifikasi muncul.	Valid

Tabel X.4

Hasil Pengujian Aplikasi dan Model terhadap Gambar Input

Gambar	Label Asli	Hasil Klasifikasi
	Manual Ilustrator	AI
	AI	Manual Ilustrator
	Manual Ilustrator	Manual Ilustrator



AI AI

XI. KESIMPULAN

Aplikasi android klasifikasi gambar anime berbasis *deep learning* berhasil dikembangkan dan dapat beroperasi secara luring. Model MobileNetV3Large dipilih berdasarkan penelitian terdahulu pada kasus yang sama [5] [7]. Model yang dilatih dengan 3358 citra latih mencapai akurasi 90.09% pada tahap pengujian awal. Akurasi tersebut justru meningkat saat model ini dikuantisasi ke format TFLite INT8 dengan akurasi 90.18%. Implementasi model INT8 memiliki ukuran model hanya 6,94 MB dengan waktu inferensi 0,0197 detik. Dengan acuan ukuran tersebut, model diterapkan dalam aplikasi sehingga aplikasi berukuran total 29,9 MB. Aplikasi yang dikembangkan memiliki ukuran total 29,9 MB, berbentuk single-page application, dan telah diuji menggunakan metode black-box testing.

Terlepas dari keberhasilan pengembangan solusi, terdapat beberapa poin pengembangan untuk meningkatkan keandalan model dan fitur aplikasi. Persiapan data yang lebih ketat, seperti stratified sampling dan targeted augmentation, menjadi prioritas untuk mengatasi ketidakseimbangan dataset pada sub-kategori minoritas dan mencegah *overfitting*. Validasi model dapat diperluas terhadap generator di luar dataset (unseen generators). Penambahan fitur visualisasi prediksi seperti Grad-CAM dapat membantu analisis bagian gambar yang menjadi penentu hasil klasifikasi. Mekanisme federated learning berbasis umpan balik pengguna dibutuhkan dalam meningkatkan kualitas klasifikasi oleh model tanpa harus mengirim data ke server.

REFERENSI

[1] Askari-Hemmat, M., Hemmat, R. A., Hoffman, A., Lazarevich, I., Saboori, E., Mastropietro, O., Sah, S., Savaria, Y., & David, J.-P. (2022). QReg: On regularization effects of quantization. arXiv. https://doi.org/10.48550/arXiv.2206.12372

[2] Du, X., Cai, Y., Wang, S., & Zhang, L. (2016). Overview of *Deep Learning*. In 2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC). IEEE. https://doi.org/10.1109/YAC.2016.7804882

[3] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. arXiv:1704.04861. https://doi.org/10.48550/arXiv.1704.04861

[4] IMDb. (n.d.). Spirited Away (2001): Awards. Retrieved January 9, 2026, from https://www.imdb.com/title/tt0245429/awards/

[5] Johan, M. E., Wong, R. F., Godata, G. B., Wijaya, W., & Haezer, E. (2025). Web-based *Deep Learning* approach to identifying AI-generated anime illustration. International Journal on Informatics Visualization, 9(4), 1545–1552. http://www.joiv.org/index.php/joiv

[6] Kim, S., Choi, Y., & Lee, M. (2015). *Deep learning* with support vector data description. Neurocomputing, 165, 111–117. https://doi.org/10.1016/j.neucom.2014.09.086

[7] Kusuma, S. W., Natalia, F., Ko, C. S., & Sudirman, S. (2024). Detection of AI-generated anime images using *deep learning*. ICIC Express Letters, Part B: Applications, 15(3), 295–301. https://doi.org/10.24507/icicelb.15.03.295

[8] Li, Z., Liu, F., Yang, W., Peng, S., & Zhou, J. (2022). A survey of convolutional neural networks: Analysis, applications, and prospects. IEEE Transactions on Neural Networks and Learning Systems, 33(12), 7000. https://doi.org/10.1109/TNNLS.2022.3140444

[9] Mitchell, T. M. (1997). Machine learning. McGraw-Hill.

[10] Nikkei Asia. (2024, June 6). AI anime flood: An infringement investigation of 90,000 images. https://asia.nikkei.com/static/vdata/infographics/ai-anime-flood/

[11] O'Shea, K., & Nash, R. R. (2015). An introduction to convolutional neural networks. arXiv (Cornell) https://doi.org/10.48550/arxiv.1511.08458

[12] Öze, N., & Esengöl, G. (2025). Perception of anime cartoon characters depending on their visual traits and facial features. Dokuz Eylül Üniversitesi Sosyal Bilimler Enstitüsü Dergisi, 27(1), 82–108. https://doi.org/10.16953/deusosbil.1475098

[13] Sarkar, A., Goyal, A., Hicks, D., Sarkar, D., & Hazra, S. (2019). Android application development: A brief overview of Android platforms and evolution of security systems. In 2019 Third International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC) (pp. xx–xx). IEEE. https://doi.org/10.1109/I-SMAC47947.2019.9032440

[14] Zhu, C., Zhang, X., Sun, Y., Chang, C.-C., & Echizen, I. (2025). AnimeDL-2M: Million-scale AI-generated anime image detection and localization in diffusion era. In Proceedings of the 1st Deepfake Forensics Workshop: Detection, Attribution, Recognition, and Adversarial Challenges in the Era of AI-Generated Media (pp. 45–54). Association for Computing Machinery. https://doi.org/10.1145/3746265.3759666