

**MODEL PEMBELAJARAN DAN LAPORAN AKHIR  
PROJECT-BASED LEARNING  
MATA KULIAH DEEP LEARNING  
KELAS B**



**"KLASIFIKASI KATEGORI FASHION MENGGUNAKAN EfficientNetB0"  
DISUSUN OLEH KELOMPOK "VII" :**

- |                                    |               |
|------------------------------------|---------------|
| 1. ARDIA EVA ARDIANI               | (22083010104) |
| 2. KANESSA JASMINE PRISHEILA A.Z.S | (22083010016) |
| 3. HAUZAN HANIFAH ZAHRA            | (22083010075) |
| 4. HIKMATA TARTILA                 | (22083010082) |
| 5. HERLAMBANG AWAN IRAWAN          | (22083010101) |

**DOSEN PENGAMPU:**

Amri Muhammin, S.Stat., M.Stat., M.S ( 199507232024061002 )

PROGRAM STUDI SAINS DATA  
FAKULTAS ILMU KOMPUTER  
UNIVERSITAS PEMBANGUNAN NASIONAL "VETERAN"  
JAWA TIMUR  
2025

## **BAB I**

### **PENDAHULUAN**

#### **1.1. Latar Belakang**

Perkembangan teknologi *computer vision* dan *deep learning* telah membawa dampak dalam berbagai bidang, salah satunya pada industri *fashion* dan *e-commerce*. Peningkatan jumlah data visual berupa gambar produk pakaian menuntut adanya sistem yang mampu melakukan klasifikasi otomatis secara cepat dan tepat (Khasanah et al., 2024). Proses pengelompokan kategori pakaian secara manual tidak hanya memakan waktu, tetapi juga rentan terhadap kesalahan dan tidak konsisten, terutama ketika jumlah data terus bertambah. Klasifikasi gambar merupakan permasalahan dasar dalam *computer vision*, yang bertujuan untuk mengenali dan memberi label pada objek dalam sebuah citra berdasarkan karakteristik visual tertentu (Mohamed et al., 2021). Dalam konteks *fashion*, klasifikasi gambar pakaian bertujuan untuk mengidentifikasi jenis atau kategori pakaian seperti *dress*, *shirt*, jaket, celana, dan kategori lainnya berdasarkan citra pakaian yang diberikan. Namun, proses ini memiliki tantangan teknis yang tinggi karena adanya variasi *intra-class* yang besar, seperti perbedaan gaya, potongan, dan bahan dalam satu kategori yang sama misalnya berbagai jenis jaket, serta kemiripan yang tipis antara *shirt* dan *t-shirt*. Sistem klasifikasi ini memiliki peranan penting dalam berbagai aplikasi, antara lain otomatisasi pengelolaan katalog produk, pencarian produk berbasis gambar, sistem rekomendasi, serta peningkatan pengalaman pengguna pada *platform e-commerce*.

Pendekatan tradisional dalam pengolahan citra umumnya mengandalkan ekstraksi fitur secara manual, seperti warna, tekstur, dan bentuk, yang kemudian diklasifikasikan menggunakan teknologi *machine learning* konvensional. Namun, pendekatan ini memiliki keterbatasan dalam menangkap kompleksitas pola visual pada data gambar yang beragam. Dengan adanya teknologi *Deep Learning*, terutama metode CNN, sekarang kita tidak perlu lagi menentukan ciri-ciri gambar secara manual (Wijaya et al., 2025). Sistem ini bisa langsung mengenali pola gambar dari data aslinya secara mendalam, yang bikin hasil tebakannya jadi jauh lebih tepat. Berbagai arsitektur CNN telah dikembangkan untuk meningkatkan performa klasifikasi citra, seperti AlexNet, VGGNet, ResNet, dan Inception. Meskipun arsitektur tersebut mampu mencapai akurasi yang tinggi, sebagian diantaranya memiliki jumlah parameter yang besar dan kebutuhan komputasi yang tinggi, sehingga kurang efisien untuk implementasi praktis. Untuk mengatasi permasalahan tersebut, arsitektur

EfficientNet, yang diperkenalkan konsep *compound scaling*, yaitu teknik penskalaan jaringan secara seimbang pada dimensi kedalaman (*depth*), lebar (*width*), dan resolusi (*resolution*).

EfficientNet ini terbukti mampu menghasilkan performa yang lebih baik dengan penggunaan sumber daya komputasi yang lebih efisien dibandingkan model CNN sebelumnya. Hal ini sejalan dengan penelitian Panggabean dan Susilawati yang menunjukkan bahwa penerapan EfficientNet pada data fashion mampu menghasilkan akurasi hingga 94,29%, meskipun masih ditemukan kendala pada kategori yang memiliki kemiripan visual tinggi (Panggabean & Susilawati, 2025). Efisiensi ini menjadi faktor penentu dalam implementasi praktis di industri *e-commerce*, di mana sistem dituntut untuk mampu memproses ribuan unggahan produk baru secara *real-time* atau dijalankan pada perangkat dengan sumber daya terbatas tanpa mengorbankan akurasi prediksi.

Berdasarkan permasalahan pada latar belakang tersebut, penelitian ini dilakukan bertujuan untuk klasifikasi gambar pakaian dengan input berupa gambar pakaian dan output berupa label kategori pakaian, menggunakan metode EfficientNet dengan verisi B0. Pemilihan metode EfficientNetB0 didasarkan pada karakteristiknya yang merupakan versi paling ringan dan efisien, namun tetap memiliki performa yang kompetitif. Dengan jumlah parameter yang terhitung lebih sedikit, EfficientNetB0 ideal untuk klasifikasi gambar pakaian karena mampu menyeimbangkan antara kecepatan inferensi dan akurasi. Dengan begitu, diharapkan metode EfficientNetB0 dapat menjadi solusi yang praktis untuk diimplementasikan pada aplikasi *e-commerce* yang membutuhkan respon cepat tanpa memerlukan spesifikasi perangkat keras tingkat tinggi.

## 1.2. Rumusan Masalah

1. Bagaimana membangun model *deep learning* menggunakan arsitektur EfficientNetB0?
2. Bagaimana mengatasi tantangan visual dalam data fashion, seperti tingginya variasi dalam satu kategori (*intra-class variation*) dan kemiripan antar kategori pakaian (*inter-class similarity*)?
3. Bagaimana mengoptimalkan parameter model agar menghasilkan tingkat akurasi yang tinggi namun tetap ringan untuk diimplementasikan pada platform *e-commerce*?

## 1.3. Tujuan Penelitian

1. Mengembangkan model deep learning dengan arsitektur EfficientNetB0.

2. Mengoptimalkan kemampuan model dalam mengekstraksi fitur hierarkis untuk mengatasi kendala variasi gaya dalam satu kategori (*intra-class variation*) serta perbedaan tipis antar kategori pakaian yang serupa (*inter-class similarity*).
3. Menghasilkan model klasifikasi yang tidak hanya akurat tetapi juga efisien dalam penggunaan parameter.

#### **1.4. Manfaat Penelitian**

1. Bagi Industri Fashion & E-commerce
  - a. Mempercepat proses pengelolaan katalog dan inventaris produk tanpa perlu input manual yang memakan waktu.
  - b. Mendukung fitur pencarian berbasis gambar (*image-based search*) dan sistem rekomendasi yang lebih presisi bagi pelanggan.
  - c. Mengurangi risiko kesalahan pelabelan (*human error*) sehingga data produk menjadi lebih terorganisir dan konsisten.
2. Bagi Pengembangan Ilmu Pengetahuan
  - a. Memberikan referensi mengenai implementasi metode EfficientNetB0 dan konsep *compound scaling* dalam menyelesaikan kasus klasifikasi objek yang memiliki kompleksitas visual tinggi seperti produk fashion.
  - b. Menjadi landasan bagi penelitian selanjutnya dalam pengembangan sistem visi komputer yang ringan (*lightweight*) namun tetap memiliki performa tinggi (*high performance*).

## BAB II

### TINJAUAN PUSTAKA

#### 2.1. Teori Penunjang

##### 2.1.1. Convolutional Neural Networks (CNN)

*Convolutional Neural Networks* (CNN) adalah dasar arsitektur deep learning untuk klasifikasi citra karena kemampuannya mengekstraksi fitur visual berlapis dari data mentah, dimana operasi konvolusi mendeteksi pola lokal dari sebuah citra dengan menerapkan kernel atau filter  $\omega$  terhadap input  $x$  [6], yang secara matematis ditulis sebagai:

$$y_{i,j}^{(k)} = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} x_i m, j + n \cdot w_{m,n}^{(k)} + b^{(k)} \quad (2.1)$$

Keterangan:

$x$  : input feature map

$w^k$  : kernel/filter ke- $k$

$b^k$  : bias

$y^k$  : feature map hasil konvolusi

yang menghasilkan feature map  $y$  dari setiap filter  $k$ , sehingga model dapat belajar representasi kompleks dari tekstur dan bentuk objek. CNN sangat relevan untuk fashion karena memungkinkan *fitur hierarkis* dikenali tanpa ekstraksi manual sebelumnya.

##### 2.1.2. Transfer Learning dalam CNN

*Transfer learning* pada CNN menyediakan pendekatan praktis untuk memanfaatkan pengetahuan dari model besar yang sudah dilatih sebelumnya (misalnya pada ImageNet) dan *fine-tune* pada tugas target seperti klasifikasi pakaian, dengan parameter backbone  $\theta_{base}$  dipertahankan dan parameter head  $\theta_{head}$  dilatih ulang, ini mempercepat pelatihan dan meningkatkan *generalization* pada data dengan variasi visual tinggi. Pendekatan ini mendukung efisiensi model pada dataset fashion yang kompleks sekaligus mengatasi keterbatasan data latih [7].

##### 2.1.3 Arsitektur EfficientNet dan *Compound Scaling*

EfficientNet adalah keluarga arsitektur CNN yang dirancang dengan prinsip compound scaling, yaitu penskalaan jaringan secara seimbang terhadap tiga aspek yaitu kedalaman (*depth*), lebar (*width*), dan resolusi input (*resolution*). Alih-alih hanya memperluas satu aspek

jaringan, EfficientNet memperluas semuanya melalui koefisien skala tunggal sehingga akselerasi akurasi tetap tinggi dengan biaya komputasi relatif rendah.

$$d = \alpha^\Phi, w = B^\Phi, r = \gamma^\Phi, \text{ dengan } \alpha \cdot B^2 \cdot \gamma^2 \approx 2 \quad (2.3)$$

yang berarti peningkatan ukuran model mengikutsertakan seluruh dimensi secara proporsional dan efisien untuk meningkatkan akurasi klasifikasi tanpa lonjakan besar dalam biaya komputasi, menjadikannya unggul pada tugas klasifikasi sebuah sistem yang dituntut respons cepat seperti e-commerce [8].

#### 2.1.4 Regularisasi dan Augmentasi untuk Meningkatkan Robustness Model CNN

Agar model CNN tidak overfit pada data pelatihan, strategi seperti data augmentation, dropout, dan batch normalization umum diterapkan. Pada klasifikasi multi kelas, fungsi loss *categorical cross-entropy* digunakan untuk meminimalkan perbedaan antara distribusi prediksi  $\hat{y}$  dan label sebenarnya  $y$ , yang dituliskan sebagai:

$$\mathcal{L} = - \sum_{i=1}^C y_i \log(\hat{y}_i) \quad (2.4)$$

dengan  $C$  jumlah kelas, sehingga model *learns* untuk memaksimalkan probabilitas kelas benar sambil memperkecil probabilitas kelas lain. Aspek ini krusial ketika model mengevaluasi dan membedakan berbagai kategori pakaian dengan variasi yang tinggi [9].

#### 2.1.5 Optimasi dan Evaluasi Kinerja Model Deep Learning

Agar model tidak terjebak *overfitting*, teknik seperti *data augmentation* (rotasi, flip, zoom) penting untuk meningkatkan robustitas; sementara *optimizer* seperti Adam mempercepat konvergensi dengan update parameter berbasis momentum yang diformulasikan melalui estimasi gradien moment pertama dan kedua, dan kinerja model dievaluasi dengan metrik akurasi, precision, recall dan F1-score melalui confusion matrix untuk menilai kualitas klasifikasi tiap kelas secara komprehensif [10].

## 2.2. Penelitian Terkait

Penelitian terkait disajikan untuk meninjau metode dan hasil penelitian sebelumnya dalam klasifikasi citra fashion berbasis deep learning, khususnya penggunaan arsitektur CNN dan EfficientNet. Tinjauan ini digunakan untuk membandingkan pendekatan, dataset, serta capaian performa, sekaligus mengidentifikasi celah penelitian yang menjadi dasar pemilihan metode EfficientNetB0 pada penelitian ini.

No	Nama Peneliti	Judul	Metode	Dataset	Hasil
1.	Shimon Abert Panggabean & Susilawati	Penerapan Arsitektur EfficientNet dalam Model CNN untuk Optimalisasi Klasifikasi Gambar Fashion pada Dataset Fashion MNIST	Convolutional Neural Network (CNN) berbasis EfficientNet dengan optimizer Adam, data augmentation, dan evaluasi menggunakan accuracy, precision, recall, F1-score, serta confusion matrix.	Fashion MNIST (70.000 citra grayscale, 10 kelas pakaian)	Berdasarkan hasil eksperimen, model EfficientNet mencapai akurasi 94,29%, precision 94,26%, recall 94,26%, dan F1-score 94,25%. Kelas Trouser dan Bag memiliki performa tertinggi, sedangkan T-shirt/top dan Shirt paling sulit karena kemiripan visual.
2.	Sheikh Sadi Bandan, MD. Samiul Islam Sabbir, Md Sharuf Hossain, Khadiza Tul Kobra	Enhancing Fashion Choices: AI-Powered Style Analysis and Recommendations	Menerapkan pendekatan Deep Learning berbasis Convolutional Neural Network (CNN) untuk melakukan	Dataset yang digunakan merupakan dataset buatan sendiri yang dikumpulkan dari berbagai sumber dengan total	Berdasarkan hasil pengujian, EfficientNet-B3 memperoleh akurasi tertinggi sebesar 86%, diikuti oleh VGG19 dengan akurasi 85%, dan EfficientNet-B0

			<p>klasifikasi gambar fashion. Penulis menggunakan customized CNN framework dengan membandingkan tujuh arsitektur CNN yaitu MobileNetV2, MobileNetV3, EfficientNet-B0, EfficientNet-B3, InceptionV3, DenseNet201, dan VGG19.</p>	<p>1.000 gambar fashion. Dataset terdiri dari 10 kategori pakaian, yaitu <i>shirt, punjabi, t-shirt, blazer, sweater, saree, salwar kameez, gown, western tops, dan party wear.</i></p>	<p>dengan akurasi 80%. Sementara itu, MobileNetV3 mencapai akurasi 75%, DenseNet201 sebesar 65%, InceptionV3 sebesar 60%, dan MobileNetV2 menunjukkan performa terendah dengan akurasi 59%.</p>
3.	Uswatun Khasanah, Okta Viona Cahyanti, Ilmaya Ni'matul Fatma, Tinuk Agustin.	Klasifikasi Model Pakaian	Menggunakan metode Convolutional	Dataset yang digunakan berasal dari Kaggle, dengan total 6.007 gambar pakaian yang terbagi ke dalam 10 kategori	Hasil penelitian menunjukkan bahwa model CNN yang dikembangkan mampu mencapai akurasi sebesar 89,59% pada data uji. Lebih tinggi dibandingkan

		<p>dengan beberapa lapisan Conv2D, MaxPooling, dan fully connected layer (Dense). Proses pelatihan dilakukan menggunakan optimizer Adam, dan evaluasi performa model menggunakan metrik akurasi, presisi, recall, serta confusion matrix.</p>	<p>pakaian. Dataset dibagi menjadi 80% data latih dan 20% data uji untuk proses pelatihan dan evaluasi model.</p>	<p>metode YOLOv3 dan ResNet50 yang mencapai akurasi sekitar 86,44%. Namun demikian, model masih mengalami kesulitan dalam mengklasifikasikan kategori pakaian yang memiliki kemiripan visual, seperti kaos dan jaket. Secara keseluruhan, penelitian ini membuktikan bahwa CNN efektif dan efisien untuk klasifikasi pakaian pada konteks e-commerce.</p>
--	--	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## **BAB III**

### **METODOLOGI**

Bab ini menjelaskan metodologi penelitian yang digunakan dalam pengembangan sistem klasifikasi citra fashion pada penelitian ini. Pembahasan pada bab ini difokuskan pada dua aspek utama, yaitu dataset penelitian yang digunakan sebagai sumber data serta langkah-langkah analisis yang diterapkan dalam proses pengolahan data dan pemodelan. Langkah analisis mencakup tahapan persiapan dataset, pembagian data, serta pembangunan dan pelatihan model klasifikasi citra menggunakan arsitektur EfficientNetB0. Metodologi yang disusun pada bab ini menjadi landasan dalam melakukan pengujian dan evaluasi kinerja model yang dibahas pada bab selanjutnya.

#### **3.1. Dataset Penelitian**

Dataset yang digunakan dalam penelitian ini merupakan dataset citra fashion *DeepFashion* yang dikembangkan oleh *Multimedia Laboratory* (MMLab), *Chinese University of Hong Kong* (CUHK). Dataset ini diperoleh dari situs resmi MMLab melalui halaman *DeepFashion: Category and Attribute Prediction Benchmark* dan digunakan secara luas dalam penelitian klasifikasi dan analisis citra fashion. Dataset *DeepFashion* berisi citra pakaian dari berbagai kategori busana yang dikumpulkan dari sumber daring, seperti situs *e-commerce* dan media fashion. Setiap citra merepresentasikan satu objek pakaian dan dilengkapi dengan anotasi kategori serta atribut visual, sehingga sesuai untuk tugas klasifikasi multi-kelas berbasis *deep learning*.

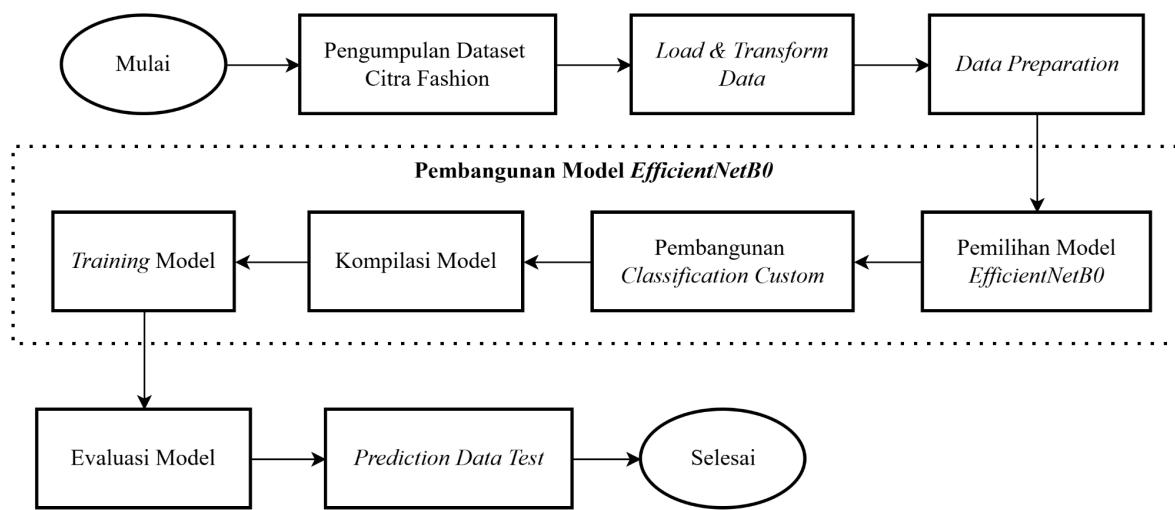
Secara keseluruhan, dataset *DeepFashion* pada *benchmark Category and Attribute Prediction* terdiri dari 289.222 citra fashion yang mencakup ribuan kategori pakaian dengan tingkat variasi visual yang tinggi. Variasi tersebut meliputi perbedaan jenis pakaian, warna, tekstur, model, pose objek, sudut pengambilan gambar, serta latar belakang. Kondisi ini menjadikan dataset *DeepFashion* sebagai dataset yang menantang dan representatif untuk menguji kemampuan model klasifikasi citra. Namun, distribusi jumlah citra pada setiap kategori dalam dataset ini tidak merata. Beberapa kategori memiliki jumlah citra yang sangat banyak, sementara kategori lainnya hanya memiliki sedikit citra. Ketidakseimbangan kelas (*class imbalance*) ini berpotensi memengaruhi kinerja model klasifikasi apabila seluruh kategori digunakan secara langsung.

Oleh karena itu, pada penelitian ini dilakukan proses seleksi dataset dengan menetapkan batas minimum jumlah citra per kategori. Kategori yang memiliki jumlah citra

kurang dari 130 gambar dieliminasi untuk menghasilkan dataset yang lebih seimbang. Setelah proses penyaringan, diperoleh dataset akhir sebanyak 9.285 citra fashion yang digunakan dalam penelitian. Dataset hasil seleksi tersebut kemudian disusun ke dalam dua skema pelabelan, yaitu 64 kelas dan 4 kelas. Skema 64 kelas merepresentasikan kategori fashion secara lebih spesifik (*fine-grained classification*), di mana setiap kelas menunjukkan jenis pakaian yang berbeda. Sementara itu, pada skema 4 kelas, kategori-kategori fashion dikelompokkan kembali ke dalam empat kelas utama yang bersifat lebih umum (*coarse-grained classification*). Pendekatan ini bertujuan untuk menganalisis pengaruh tingkat granularitas label terhadap kinerja model klasifikasi citra fashion menggunakan EfficientNet. Dataset ini selanjutnya digunakan pada seluruh tahapan penelitian, mulai dari proses persiapan data, pelatihan model, hingga evaluasi dan analisis hasil klasifikasi.

### 3.2. Langkah Analisis

Langkah analisis dalam penelitian ini disusun untuk menggambarkan alur kerja pengembangan sistem klasifikasi citra fashion menggunakan arsitektur EfficientNetB0 secara sistematis. Tahapan analisis disajikan dalam bentuk diagram alir (*flowchart*) guna memberikan gambaran menyeluruh mengenai proses penelitian, mulai dari tahap persiapan data hingga evaluasi kinerja model. Diagram alir ini digunakan sebagai acuan utama dalam pelaksanaan setiap tahapan analisis.



**Gambar 3.1.** Diagram Alir Penelitian

Berdasarkan diagram alir pada Gambar 3.1, tahapan analisis dalam penelitian dilakukan secara sistematis mulai dari pengolahan dataset hingga analisis hasil klasifikasi. Adapun penjelasan setiap tahapan adalah sebagai berikut:

## 1. Pengumpulan Dataset Citra Fashion

Tahap awal penelitian dimulai dengan pengumpulan dataset citra fashion yang diperoleh dari sumber MMLab. Dataset ini berisi citra pakaian dengan jumlah kategori yang besar dan variasi visual yang tinggi. Dataset awal memiliki distribusi kelas yang tidak seimbang, sehingga memerlukan tahapan persiapan data lebih lanjut sebelum digunakan dalam proses pemodelan.

## 2. *Load* dan Eksplorasi Dataset

Dataset citra dimuat ke dalam lingkungan Google Colab yang terintegrasi dengan Google Drive. Data yang masih dalam bentuk file terkompresi diekstraksi dan disusun dalam struktur folder yang konsisten. Tahap ini bertujuan untuk memastikan seluruh citra dapat diakses dan dibaca secara otomatis oleh sistem pada tahap analisis berikutnya.

## 3. Data *Preparation*

Tahap data *preparation* merupakan tahapan penting yang bertujuan untuk menyiapkan dataset agar layak digunakan dalam proses pelatihan model. Pada tahap ini dilakukan beberapa proses utama, meliputi:

- Pengumpulan *path* citra dan label ke dalam struktur data terorganisir.
- Eksplorasi awal dataset untuk memahami jumlah citra dan distribusi kategori.
- Penyaringan kategori berdasarkan jumlah minimum citra per kelas.
- Pembentukan dua skema label, yaitu *fine-grained* (64 kelas) dan *coarse-grained* (4 kelas).
- *Encoding* label ke dalam bentuk numerik.
- Pembagian dataset menjadi data latih, validasi, dan data uji secara *stratified*.
- Penerapan data *augmentation* pada data latih.
- Pembangunan *pipeline* dataset menggunakan `tf.data.Dataset`.

Tahapan ini bertujuan untuk mengurangi ketidakseimbangan kelas, meningkatkan kualitas data, serta mendukung proses pelatihan model yang lebih stabil.

## 4. Pemilihan Model EfficientNetB0

Pada tahap ini dipilih arsitektur EfficientNetB0 sebagai *backbone* utama model klasifikasi citra. Model ini digunakan pada kedua skenario klasifikasi (64 kelas dan 4 kelas) dengan pendekatan *transfer learning* untuk mengekstraksi fitur visual dari citra fashion.

## 5. Pembangunan *Classification Custom*

Setelah *backbone* model ditentukan, dilakukan pembangunan lapisan klasifikasi (*classification head*) yang disesuaikan dengan jumlah kelas pada masing-masing skema

klasifikasi. Struktur *classification head* dibuat berbeda antara model *fine-grained* dan *coarse-grained* untuk menyesuaikan tingkat kompleksitas permasalahan klasifikasi.

#### 6. Kompilasi Model

Model yang telah dibangun kemudian dikompilasi dengan menentukan fungsi *loss*, *optimizer*, dan metrik evaluasi. Konfigurasi kompilasi disesuaikan dengan karakteristik masing-masing skema klasifikasi untuk menjaga stabilitas proses pelatihan.

#### 7. *Training* Model

Model dilatih menggunakan data latih dan divalidasi menggunakan data validasi. Proses *training* dilakukan untuk mempelajari pola visual pada citra fashion, dengan penerapan mekanisme seperti *early stopping* untuk mencegah *overfitting*.

#### 8. Evaluasi Model

Evaluasi model dilakukan untuk menilai kinerja umum model setelah proses pelatihan. Evaluasi ini mencakup pengamatan nilai akurasi dan *loss* pada data latih dan validasi guna memastikan model tidak mengalami *overfitting* maupun *underfitting*.

#### 9. *Prediction Data Test*

Tahap *prediction data test* bertujuan untuk menganalisis kemampuan model dalam melakukan klasifikasi pada data uji. Pada tahap ini dilakukan prediksi terhadap data *test* dan dianalisis menggunakan:

- *Classification Report*, untuk mengukur performa model pada setiap kelas melalui metrik *precision*, *recall*, dan *f1-score*.
- *Confusion Matrix*, untuk mengamati pola kesalahan klasifikasi serta perbedaan struktur prediksi antara klasifikasi 64 kelas dan 4 kelas.

## **BAB IV**

### **HASIL DAN PEMBAHASAN**

Bab ini membahas hasil pengujian, evaluasi model, serta implementasi sistem klasifikasi citra fashion yang digunakan dalam penelitian. Analisis dilakukan melalui serangkaian tahapan yang dimulai dari proses pemuat dan transformasi dataset citra fashion, pemahaman dan eksplorasi data (*data understanding*), persiapan dataset, pembagian data ke dalam data latih, validasi, dan uji, hingga pembangunan dan pelatihan model *Convolutional Neural Network* berbasis EfficientNetB0. Selanjutnya, dilakukan evaluasi model untuk menilai kinerja klasifikasi menggunakan metrik akurasi, loss, serta *classification report*. Selain itu, sistem prediksi pada data uji diimplementasikan untuk mengamati kemampuan model dalam mengenali kategori fashion serta menganalisis kesalahan prediksi melalui *confusion matrix* dan visualisasi Grad-CAM sebagai bentuk interpretabilitas model.

#### **4.1. Dataset Penelitian**

Dataset yang digunakan dalam penelitian ini merupakan dataset citra fashion yang berisi berbagai jenis pakaian dengan berbagai kategori, seperti blazer, blouse, dan jenis busana lainnya dari sumber website mmlab pada [1]. Setiap citra merepresentasikan satu objek pakaian dan dikelompokkan ke dalam folder berdasarkan nama kategori, sehingga struktur data secara alami mencerminkan label kelas masing-masing. Dataset ini memiliki skala yang besar dengan variasi visual yang tinggi, baik dari segi warna, tekstur, maupun bentuk pakaian, sehingga menantang untuk tugas klasifikasi multi kelas.

Secara keseluruhan, dataset awal terdiri dari 289.229 citra yang terbagi ke dalam 5.621 kategori, namun distribusi jumlah citra pada setiap kategori tidak merata. Beberapa kategori memiliki jumlah data yang sangat banyak, sementara kategori lainnya hanya memiliki sedikit citra. Kondisi ini berpotensi menyebabkan ketidakseimbangan kelas (*class imbalance*) yang dapat memengaruhi kinerja model dalam proses pelatihan dan evaluasi.

Untuk mengatasi permasalahan tersebut, dilakukan proses seleksi dataset dengan menetapkan batas minimum jumlah citra per kategori. Kategori yang memiliki jumlah citra kurang dari 130 gambar dieliminasi dari dataset. Setelah proses penyaringan, dataset yang digunakan dalam penelitian ini berjumlah 9.285 citra yang terbagi ke dalam 4 dan 64 kategori fashion. Skema 64 label merepresentasikan kategori fashion secara spesifik, di mana setiap kelas menunjukkan jenis pakaian yang berbeda. Sementara itu, pada skema 4 label, kategori-kategori fashion yang ada dikelompokkan ulang ke dalam empat kelas utama yang

bersifat lebih umum. Pendekatan ini bertujuan untuk menganalisis pengaruh tingkat granularitas label terhadap kinerja model klasifikasi. Dataset inilah yang selanjutnya digunakan pada seluruh tahapan pemrosesan, pelatihan, dan evaluasi model klasifikasi citra.

#### 4.2. Load and Transform Data

Tahap awal penelitian diawali dengan proses pemuatan dan transformasi data citra menggunakan lingkungan Google Colab yang terintegrasi dengan Google Drive. Proses *mounting* Google Drive dilakukan untuk memudahkan akses terhadap dataset yang tersimpan secara daring. Dataset citra fashion yang digunakan berbentuk file terkompresi (ZIP), kemudian diekstraksi ke dalam direktori lokal Colab agar dapat diproses lebih lanjut oleh sistem dengan kode pada kode program 4.1.

Kode Program 4.1. Load and Transform Data

```
try:  
    from google.colab import drive  
    drive.mount('/content/drive', force_remount=True)  
except:  
    print("Drive mount failed. Restart runtime then try again.")  
  
import os  
# Folder tempat menyimpan file  
DRIVE_BASE = '/content/drive/MyDrive/DATA PJBL DEEL B'  
  
# File ZIP gambar  
DRIVE_ZIP_PATH = os.path.join(DRIVE_BASE, 'img.zip')  
  
# Folder anotasi  
ANNOT_DIR = DRIVE_BASE  
  
# Lokasi extract di storage Colab  
EXTRACT_TO = '/content/deepfashion_img'  
  
print("DRIVE_ZIP_PATH:", DRIVE_ZIP_PATH)  
print("ANNOT_DIR:", ANNOT_DIR)  
print("EXTRACT_TO:", EXTRACT_TO)
```

Tahap ini memastikan bahwa seluruh data citra tersedia dalam satu struktur folder yang konsisten sehingga memudahkan proses pembacaan data secara otomatis pada tahap berikutnya.

## 4.3. Data Preparation

### 4.3.1. Pengumpulan Data Gambar dan Label

Pada tahap ini, seluruh citra yang telah diekstraksi dikumpulkan beserta labelnya berdasarkan nama folder masing-masing kategori dengan kode program 4.2.

```
Kode Program 4.2. Pengumpulan Data dan Label

import pandas as pd
from pathlib import Path
from sklearn.model_selection import train_test_split

# Lokasi gambar setelah extract
IMG_ROOT = EXTRACT_TO + "/img"

image_paths = []
labels = []

for root, dirs, files in os.walk(IMG_ROOT):
    for f in files:
        if f.lower().endswith('.jpg', '.jpeg', '.png'):
            full_path = os.path.join(root, f)
            folder = Path(full_path).parts[-2]
            image_paths.append(full_path)
            labels.append(folder)

df = pd.DataFrame({
    'path': image_paths,
    'label_name': labels
})

print("Total images awal      :", len(df))
print("Total categories awal  :", df['label_name'].nunique())
df
```

Hasil eksplorasi awal pada gambar 4.1 menunjukkan bahwa dataset terdiri dari 289.229 citra dengan 5.621 kategori, yang mencerminkan dataset berskala besar namun memiliki distribusi kelas yang sangat tidak seimbang. Setiap citra dipetakan ke dalam sebuah *dataframe* yang berisi informasi path file dan nama label kategori, sehingga memudahkan pengelolaan data secara terstruktur.

```

Total images awal      : 289229
Total categories awal : 5621

      path          label_name
0   /content/deepfashion_img/img/Favorite_Petite_S... Favorite_Petite_Skinny_Jeans
1   /content/deepfashion_img/img/Favorite_Petite_S... Favorite_Petite_Skinny_Jeans
2   /content/deepfashion_img/img/Favorite_Petite_S... Favorite_Petite_Skinny_Jeans
3   /content/deepfashion_img/img/Favorite_Petite_S... Favorite_Petite_Skinny_Jeans
4   /content/deepfashion_img/img/Favorite_Petite_S... Favorite_Petite_Skinny_Jeans
...
289224 /content/deepfashion_img/img/Pardon_Marled_Swe... Pardon_Marled_Sweater
289225 /content/deepfashion_img/img/Pardon_Marled_Swe... Pardon_Marled_Sweater
289226 /content/deepfashion_img/img/Pardon_Marled_Swe... Pardon_Marled_Sweater
289227 /content/deepfashion_img/img/Pardon_Marled_Swe... Pardon_Marled_Sweater
289228 /content/deepfashion_img/img/Pardon_Marled_Swe... Pardon_Marled_Sweater

289229 rows × 2 columns

```

**Gambar 4.1.** Eksplorasi Data Awal

#### 4.3.2. Penyaringan Kategori

Untuk mengatasi permasalahan ketidakseimbangan kelas, dilakukan proses penyaringan kategori dengan menetapkan ambang batas minimum jumlah citra per kelas, yaitu 130 citra seperti pada kode program 4.3.

<b>Kode Program 4.3.</b> Penyaringan Kategori	
Fine-Grained (64 Class)	Coarse-Grained (4 Class)
<pre> # Menghitung jumlah gambar per kategori counts = df['label_name'].value_counts()  # Menyaring kategori yang memiliki lebih dari atau sama dengan 130 gambar valid_categories = counts[counts &gt;= 130].index  # Filter dataframe hanya untuk kategori yang valid df_filtered = df[df['label_name'].isin(valid_categories)].reset_index(drop=True)  # Menampilkan hasil </pre>	<pre> # Coarse-Grained Label Mapping (CG) # Membuat label coarse-grained (CG) df["cg_label_name"] = df["label_name"].apply(simplify_label)  # Cek distribusi label CG print(df["cg_label_name"].value_counts()) </pre>

```

print("Total      images      setelah
filter:", len(df_filtered))
print("Total  kategori      :",
df_filtered['label_name'].nunique())

# Jika ingin menggunakan df_filtered
untuk proses selanjutnya
df = df_filtered

```

Pada gambar 4.2 kategori yang tidak memenuhi kriteria tersebut dieliminasi dari dataset. Setelah proses *filtering*, jumlah data berkurang menjadi 9.285 citra yang terbagi ke dalam 4 dan 64 kategori *fashion*. Langkah ini bertujuan untuk meningkatkan stabilitas proses pelatihan dan mencegah bias model terhadap kelas dengan jumlah data yang sedikit.

```

          cg_label_name
          Blouse    5438
          Blazer    3378
          Jacket    309
          Coat      160
...  Total images setelah filter: 9285
      Total kategori      : 64           Name: count, dtype: int64

```

**Gambar 4.2.** Hasil *Filtering* Data

#### 4.3.3. Encoding Label

Label kategori yang semula berbentuk teks kemudian diubah menjadi representasi numerik menggunakan teknik label encoding pada kode program 4.4.

<b>Kode Program 4.4.</b> Encoding Label	
Fine-Grained (64 Class)	Coarse-Grained (4 Class)
<pre> label_to_idx = {name: i for i, name in enumerate(sorted(df['label_name'].un ique()))} df['label'] = df['label_name'].map(label_to_idx) df </pre>	<pre> # Encoding Coarse-Grained Labels  # Mapping label CG ke indeks numerik cg_label_map = {     name: i for i, name in enumerate(sorted(df["cg_label_name"]) .unique())) }  # Encode label df["cg_label"] = df["cg_label_name"].map(cg_label_map) </pre>

```
# Jumlah kelas CG
CG_NUM_CLASSES = len(CG_label_map)
print("CG_NUM_CLASSES:", CG_NUM_CLASSES)
```

Setiap kategori diberi indeks numerik unik agar dapat digunakan sebagai target pada proses pelatihan model klasifikasi berbasis deep learning. Proses ini merupakan langkah penting karena model hanya dapat memproses label dalam bentuk numerik. Hasil dari Encoding dijelaskan pada gambar 4.3.

	path	label_name	label
0	/content/deepfashion_img/img/Textured_Open_Fro...	Textured_Open_Front_Blazer	54
1	/content/deepfashion_img/img/Textured_Open_Fro...	Textured_Open_Front_Blazer	54
2	/content/deepfashion_img/img/Textured_Open_Fro...	Textured_Open_Front_Blazer	54
3	/content/deepfashion_img/img/Textured_Open_Fro...	Textured_Open_Front_Blazer	54
4	/content/deepfashion_img/img/Textured_Open_Fro...	Textured_Open_Front_Blazer	54
...	...	...	...

**Gambar 4.3.** Encoding Label

#### 4.3.4. Pembagian Dataset

Dataset yang telah diproses selanjutnya dibagi menjadi tiga bagian, yaitu data latih, data validasi, dan data uji. Pembagian dilakukan dengan proporsi 80% untuk data latih, serta masing-masing 10% untuk data validasi dan data uji dengan kode program 4.5.

##### Kode Program 4.5. Pembagian Dataset

```
# Data Train
train_df, temp_df = train_test_split(
    df,
    test_size=0.2,
    random_state=42,
    stratify=df['label']
)

# Data Val/Test
val_df, test_df = train_test_split(
    temp_df,
    test_size=0.5,
    random_state=42,
```

```

        shuffle=True
    )

print("UKURAN DATASET:")
print("Train  :", len(train_df))
print("Val   :", len(val_df))
print("Test  :", len(test_df))

```

Proses pembagian dilakukan secara stratified untuk menjaga distribusi kelas tetap seimbang pada setiap subset. Hasil pembagian pada gambar menghasilkan 7.428 data latih, 928 data validasi, dan 929 data uji.

#### 4.3.5. Penyimpanan Dataset

Untuk memastikan reproduksibilitas eksperimen, masing-masing subset data disimpan dalam bentuk file CSV (*manifest file*) yang berisi path citra dan labelnya dengan kode program 4.6. Penyimpanan ini memudahkan penggunaan ulang dataset tanpa harus mengulangi proses *preprocessing* dari awal.

##### Kode Program 4.6. Penyimpanan Dataset

```

train_df.to_csv(f"{EXTRACT_TO}/train_manifest.csv", index=False)
val_df.to_csv(f"{EXTRACT_TO}/val_manifest.csv", index=False)
test_df.to_csv(f"{EXTRACT_TO}/test_manifest.csv", index=False)

print("\nManifest files saved.")

```

#### 4.3.6. Data Augmentation

*Data augmentation* diterapkan pada data latih untuk meningkatkan keragaman data dan mengurangi risiko *overfitting*. Teknik augmentasi yang digunakan meliputi rotasi ringan, *zoom*, serta *flipping horizontal*. Teknik augmentasi dijelaskan pada kode program 4.7. Dengan adanya augmentasi, model diharapkan mampu mengenali variasi visual pada citra fashion meskipun berasal dari kategori yang sama.

##### Kode Program 4.7. Data Augmentation

Fine-Grained (64 Class)	Coarse-Grained (4 Class)
import tensorflow as tf	# Image Preprocessing & Data Augmentation (Coarse-Grained)

<pre> IMG_SIZE = 224 BATCH_SIZE = 32 AUTOTUNE = tf.data.AUTOTUNE  # EfficientNet preprocessor preprocess = tf.keras.applications.efficientnet.preprocess_input  #Function load &amp; preprocess image def load_image(path, label):     img = tf.io.read_file(path)     img = tf.image.decode_jpeg(img, channels=3)     img = tf.image.resize(img, (IMG_SIZE, IMG_SIZE))     img = preprocess(img)     return img, label  #Augmentasi untuk training data_augmentation = tf.keras.Sequential([     tf.keras.layers.RandomFlip("horizontal"),     tf.keras.layers.RandomRotation(0.05),     tf.keras.layers.RandomZoom(0.1), ])  def load_image_train(path, label):     img = tf.io.read_file(path)     img = tf.image.decode_jpeg(img, channels=3)     img = tf.image.resize(img, (IMG_SIZE, IMG_SIZE))     img = data_augmentation(img)     return img, label </pre>	<pre> import tensorflow as tf  CG_IMG_SIZE = 224 CG_BATCH_SIZE = 32 CG_AUTOTUNE = tf.data.AUTOTUNE  # EfficientNet preprocessor (CG) cg_preprocess = tf.keras.applications.efficientnet.preprocess_input  # Load &amp; preprocess image (validation / test) def cg_load_image(path, label):     img = tf.io.read_file(path)     img = tf.image.decode_jpeg(img, channels=3)     img = tf.image.resize(img, (CG_IMG_SIZE, CG_IMG_SIZE))     img = cg_preprocess(img)     return img, label  # Data augmentation (training only) cg_data_augmentation = tf.keras.Sequential([     tf.keras.layers.RandomFlip("horizontal"),     tf.keras.layers.RandomRotation(0.05),     tf.keras.layers.RandomZoom(0.1), ])  # Load, augment, &amp; preprocess image (training) def cg_load_image_train(path, label):     img = tf.io.read_file(path)     img = tf.image.decode_jpeg(img, channels=3)     img = tf.image.resize(img, (CG_IMG_SIZE, CG_IMG_SIZE))     img = cg_data_augmentation(img) </pre>
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

	<pre> img = cg_preprocess(img) return img, label </pre>
--	---------------------------------------------------------

#### 4.3.7. Dataset *Pipeline*

Dataset kemudian dikonversi ke dalam format tf.data.Dataset untuk meningkatkan efisiensi pemrosesan selama pelatihan. *Pipeline* ini mencakup proses pembacaan citra, *preprocessing*, *batching*, dan *prefetching* sehingga proses training dapat berjalan lebih optimal dan cepat. Pipeline ini dijelaskan pada program 4.8.

<b>Kode Program 4.8.</b> Dataset Pipeline
-------------------------------------------

```

#DataFrame → tf.data
def df_to_dataset(df, training=False):
    paths = df['path'].values
    labels = df['label'].values

    ds = tf.data.Dataset.from_tensor_slices((paths, labels))

    if training:
        ds = ds.shuffle(buffer_size=len(df))
        ds = ds.map(load_image_train, num_parallel_calls=AUTOTUNE)
    else:
        ds = ds.map(load_image, num_parallel_calls=AUTOTUNE)

    ds = ds.batch(BATCH_SIZE)
    ds = ds.prefetch(AUTOTUNE)
    return ds

train_ds = df_to_dataset(train_df, training=True)
val_ds   = df_to_dataset(val_df, training=False)
test_ds  = df_to_dataset(test_df, training=False)

print("Train batches:", len(train_ds))
print("Val batches:", len(val_ds))
print("Test batches:", len(test_ds))

```

Berdasarkan konfigurasi dari program 4.8, data latih yang berjumlah 7.428 citra terbagi menjadi 233 batch, data validasi sebanyak 928 citra terbagi menjadi 29 batch, dan data uji sebanyak 929 citra terbagi menjadi 30 batch. Perbedaan jumlah

batch pada setiap subset disebabkan oleh perbedaan jumlah data yang digunakan serta pembagian batch yang bersifat otomatis berdasarkan ukuran batch yang telah ditentukan.

#### 4.3.8. Visualisasi Data

Sebagai tahap eksplorasi, dilakukan visualisasi beberapa sampel citra dari dataset latih. Visualisasi ini bertujuan untuk memastikan bahwa proses preprocessing dan augmentasi berjalan dengan baik, serta memberikan gambaran awal mengenai variasi data yang digunakan dalam pelatihan model. visualisasi data menggunakan matplotlib dengan kode program 4.9.

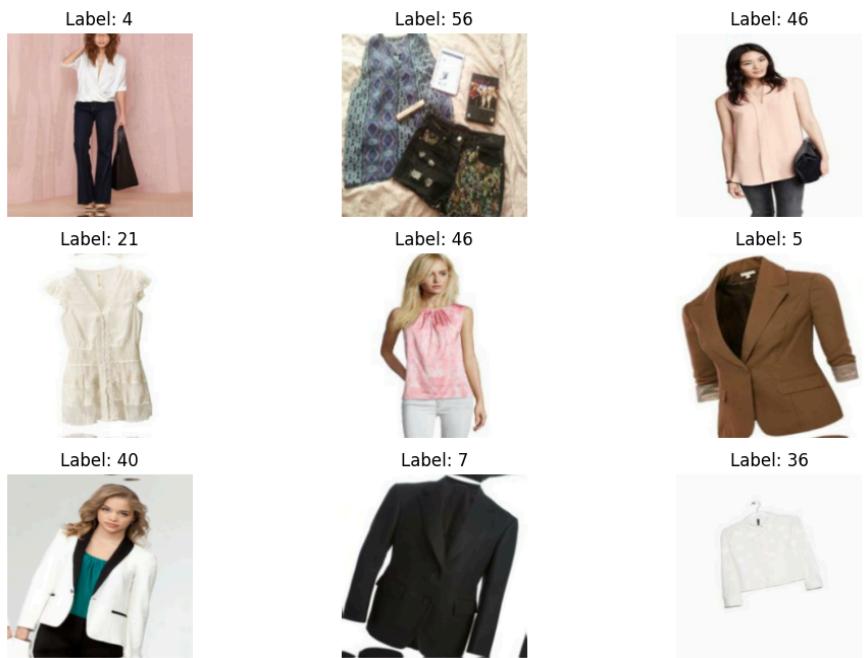
##### Kode Program 4.9. Visualisasi Data

```
import matplotlib.pyplot as plt

batch_images, batch_labels = next(iter(train_ds))

plt.figure(figsize=(12, 8))
for i in range(9):
    plt.subplot(3, 3, i+1)
    plt.imshow(batch_images[i].numpy().astype("uint8"))
    plt.title(f"Label: {batch_labels[i].numpy()}")
    plt.axis("off")
plt.show()
```

Hasil visualisasi pada Gambar 4.4, menunjukkan bahwa citra telah berhasil diproses ke ukuran yang seragam, yaitu  $224 \times 224$  piksel, sesuai dengan kebutuhan input model EfficientNetB0.



**Gambar 4.4.** Visualisasi Data

Selain itu, variasi visual seperti perbedaan warna, pola, dan bentuk pakaian dapat diamati dengan jelas pada setiap sampel citra. Hal ini menunjukkan bahwa dataset memiliki tingkat keberagaman yang tinggi, yang penting untuk melatih model agar mampu mengenali berbagai karakteristik objek fashion.

#### 4.4. Model Building

##### 4.4.1. Pemilihan Model EfficientNetB0

Pada penelitian ini digunakan arsitektur EfficientNetB0 sebagai backbone utama baik pada skenario klasifikasi *fine-grained* (64 kelas) maupun *coarse-grained* (4 kelas). Pemilihan EfficientNetB0 didasarkan pada pertimbangan keseimbangan antara kompleksitas model dan efisiensi komputasi, mengingat dataset yang digunakan berjumlah relatif besar (9.285 citra setelah proses *filtering*) namun dilatih dengan keterbatasan sumber daya. EfficientNetB0 memiliki jumlah parameter sekitar 4 juta dan telah dilatih sebelumnya pada dataset ImageNet, sehingga mampu mengekstraksi fitur visual tingkat tinggi yang relevan untuk citra fashion seperti tekstur kain, pola, dan siluet pakaian. Program pemilihan model dijelaskan pada kode program 4.10.

<b>Kode Program 4.10.</b> Pemilihan Model	
Fine-Grained (64 Class)	Coarse-Grained (4 Class)

```

import tensorflow as tf
from tensorflow.keras import layers,
models
from tensorflow.keras.callbacks import ModelCheckpoint,
EarlyStopping, ReduceLROnPlateau
from tensorflow.keras.optimizers import Adam

# Jumlah kelas dan ukuran gambar
IMG_SIZE = 224
NUM_CLASSES = df['label'].nunique()
# = 64
BATCH_SIZE = 32

# Load model pre-trained EfficientNetB0
pretrained_model = tf.keras.applications.EfficientNetB0(
    input_shape=(IMG_SIZE, IMG_SIZE, 3),
    include_top=False,
    weights='imagenet',
    pooling='max'
)

# Freeze base model
pretrained_model.trainable = False

# DataFrames → tf.data Pipeline Construction (Coarse-Grained)
def cg_df_to_dataset(df, training=False):
    paths = df["path"].values
    labels = df["cg_label"].values

    ds = tf.data.Dataset.from_tensor_slices((paths, labels))

    if training:
        ds = ds.shuffle(buffer_size=len(df))
        ds = ds.map(cg_load_image_train,
                    num_parallel_calls=CG_AUTOTUNE)
    else:
        ds = ds.map(cg_load_image,
                    num_parallel_calls=CG_AUTOTUNE)

    ds = ds.batch(CG_BATCH_SIZE)
    ds = ds.prefetch(CG_AUTOTUNE)
    return ds

cg_train_ds = cg_df_to_dataset(cg_train_df,
                               training=True)
cg_val_ds = cg_df_to_dataset(cg_val_df,
                             training=False)
cg_test_ds = cg_df_to_dataset(cg_test_df,
                             training=False)

print("CG Train batches:", len(cg_train_ds))
print("CG Val batches : ", len(cg_val_ds))
print("CG Test batches : ", len(cg_test_ds))

```

Pada model 64 kelas, EfficientNetB0 digunakan dengan konfigurasi *include\_top=False* dan *pooling='max'*. Pendekatan ini memungkinkan model untuk

menghasilkan representasi fitur global dari citra tanpa terikat pada klasifikasi ImageNet, sehingga lapisan klasifikasi dapat disesuaikan dengan jumlah kelas *fashion* yang digunakan. Sebaliknya, pada model 4 kelas, EfficientNetB0 dikonfigurasikan dengan *pooling='avg'* yang bertujuan menghasilkan fitur yang lebih stabil dan general untuk klasifikasi tingkat kategori yang lebih kasar. Perbedaan konfigurasi pooling ini mencerminkan perbedaan tujuan pembelajaran antara klasifikasi detail dan klasifikasi kategori umum. Pada skenario klasifikasi coarse-grained (4 kelas), proses pembentukan dataset melalui pipeline tf.data.Dataset menghasilkan 233 batch data latih, 29 batch data validasi, dan 30 batch data uji.

#### 4.4.2. Pembangunan *Classification Custom*

Pada model fine-grained (64 kelas), *classification head* dibangun menggunakan dua lapisan dense bertingkat dengan jumlah neuron 128 dan 256 yang masing-masing diikuti oleh *Batch Normalization* dan *Dropout* sebesar 0,45. Struktur ini dirancang untuk meningkatkan kapasitas representasi model dalam membedakan variasi antar kelas fashion yang sangat mirip secara visual, seperti perbedaan kecil antar jenis blouse atau blazer. Penggunaan *dropout* yang relatif besar menunjukkan upaya eksplisit untuk mengurangi *overfitting*, mengingat jumlah sampel per kelas minimum dibatasi pada 130 citra. Kode pembangunan *Classification Head* dijelaskan pada kode program 4.11

Kode Program 4.11. Custom Classification	
Fine-Grained (64 Class)	Coarse-Grained (4 Class)
<pre># Membuat layer custom untuk # klasifikasi inputs = layers.Input(shape=(IMG_SIZE,                            IMG_SIZE, 3)) x = pretrained_model(inputs,                       training=False) # Freeze base model x = layers.Dense(128,                  activation='relu')(x) x = layers.BatchNormalization()(x) x = layers.Dropout(0.45)(x) x = layers.Dense(256,                  activation='relu')(x) x = layers.BatchNormalization()(x)</pre>	<pre>cg_inputs = tf.keras.Input(shape=(CG_IMG_SIZE,                                    CG_IMG_SIZE, 3)) x = cg_base_model(cg_inputs,                    training=False) x = tf.keras.layers.Dense(256,                          activation="relu")(x) x = tf.keras.layers.BatchNormalization()(x) x = tf.keras.layers.Dropout(0.5)(x) cg_outputs = tf.keras.layers.Dense(CG_NUM_CLASSES,                                    activation="softmax")(x)</pre>

<pre> x = layers.Dropout(0.45)(x)  # Output layer untuk klasifikasi outputs = layers.Dense(NUM_CLASSES, activation='softmax')(x)  # Model akhir model = models.Model(inputs=inputs, outputs=outputs) </pre>	<pre> cg_model = tf.keras.Model(cg_inputs, cg_outputs) </pre>
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------

Sebaliknya, pada model *coarse-grained* (4 kelas), *classification head* dibuat lebih sederhana dengan satu lapisan dense berukuran 256 neuron, Batch *Normalization*, dan *Dropout* sebesar 0,5 sebelum lapisan output softmax. Penyederhanaan ini sejalan dengan sifat permasalahan *coarse-grained classification* yang tidak memerlukan pemisahan fitur yang terlalu detail, karena kelas-kelas seperti *Blouse*, *Blazer*, *Coat*, dan *Jacket* memiliki perbedaan visual yang lebih jelas. Jumlah parameter *trainable* pada model ini lebih sedikit, yang berkontribusi pada proses pelatihan yang lebih stabil dan konvergen lebih cepat.

#### 4.4.3. Kompilasi Model

Pada model 64 kelas, proses kompilasi dilakukan menggunakan optimizer Adam dengan learning rate sebesar  $1 \times 10^{-4}$  dan fungsi loss *sparse\_categorical\_crossentropy*. Pemilihan learning rate yang relatif kecil bertujuan untuk menjaga stabilitas pembelajaran karena backbone Efficien5Net dibekukan (*frozen*) dan hanya *classification head* yang dilatih. Metrik evaluasi yang digunakan adalah accuracy, yang relevan untuk mengukur kemampuan model dalam mengklasifikasikan setiap citra ke kelas yang tepat.

Sementara itu, pada model 4 kelas, optimizer Adam digunakan dengan learning rate yang lebih besar, yaitu  $1 \times 10^{-3}$ . Peningkatan learning rate ini dimungkinkan karena kompleksitas tugas lebih rendah dan jumlah kelas jauh lebih sedikit. Dengan konfigurasi ini, model coarse-grained mampu mempelajari batas keputusan antar kelas dengan lebih cepat tanpa mengalami instabilitas selama proses pelatihan. Perbandingan antar model ini dituliskan pada Gambar 4.5.

Fine-Grained (64 Kelas)

Layer (type)	Output Shape	Param #
input_layer_2 (InputLayer)	(None, 224, 224, 3)	0
efficientnetb0 (Functional)	(None, 1280)	4,049,571
dense (Dense)	(None, 128)	163,968
batch_normalization (BatchNormalization)	(None, 128)	512
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 256)	33,024
batch_normalization_1 (BatchNormalization)	(None, 256)	1,024
dropout_1 (Dropout)	(None, 256)	0
dense_2 (Dense)	(None, 64)	16,448

Total params: 4,264,547 (16.27 MB)  
Trainable params: 214,288 (836.75 KB)  
Non-trainable params: 4,050,339 (15.45 MB)

Coarse-Grained (4 Kelas)

Model: "functional_3"		
Layer (type)	Output Shape	Param #
input_layer_5 (InputLayer)	(None, 224, 224, 3)	0
efficientnetb0 (Functional)	(None, 1280)	4,049,571
dense_3 (Dense)	(None, 256)	327,936
batch_normalization_2 (BatchNormalization)	(None, 256)	1,024
dropout_2 (Dropout)	(None, 256)	0
dense_4 (Dense)	(None, 4)	1,028

Total params: 4,379,559 (16.71 MB)  
Trainable params: 329,476 (1.26 MB)  
Non-trainable params: 4,050,083 (15.45 MB)

Gambar 4.5. Perbandingan Model Coarse dan Fine

#### 4.4.3. Training Model

Hasil pelatihan model *fine-grained* (64 kelas) menunjukkan bahwa proses pembelajaran berjalan relatif lambat dengan peningkatan akurasi yang bertahap. Akurasi *training* meningkat dari sekitar 1,5% pada epoch awal hingga mencapai sekitar 19% pada *epoch* ke-20. Nilai loss *training* menurun secara konsisten dari 5,39 menjadi sekitar 3,16, yang mengindikasikan bahwa model mampu mempelajari pola data, namun kesulitan dalam melakukan pemisahan antar kelas yang sangat mirip. Hal ini diperkuat oleh hasil evaluasi pada data uji yang hanya mencapai akurasi 27,66%, mencerminkan tingginya kompleksitas permasalahan *fine-grained fashion classification*.

Sebaliknya, pada model *coarse-grained* (4 kelas), proses training menunjukkan performa yang jauh lebih baik. Akurasi *training* sudah mencapai lebih dari 70% pada epoch pertama dan terus meningkat hingga mendekati 89%. *Loss training* dan *validation* menurun secara stabil tanpa indikasi *overfitting* yang signifikan, berkat penggunaan *Early Stopping*. Evaluasi pada data uji menghasilkan akurasi sebesar 88,27%, yang menunjukkan bahwa pendekatan *coarse-grained classification* jauh lebih efektif dalam konteks dataset ini. perbedaan ini dijelaskan pada gambar 4.6.

*fine-grained* (64 kelas)

```

33/233 ━━━━━━━━━━ 84s 363ms/step - accuracy: 0.1410 - loss: 3.4966 - val_accuracy: 0.2457
poch 14/20
33/233 ━━━━━━━━━━ 80s 343ms/step - accuracy: 0.1631 - loss: 3.3810 - val_accuracy: 0.2446
poch 15/20
33/233 ━━━━━━━━━━ 79s 339ms/step - accuracy: 0.1548 - loss: 3.3797 - val_accuracy: 0.2532
poch 16/20
33/233 ━━━━━━━━━━ 81s 335ms/step - accuracy: 0.1682 - loss: 3.2979 - val_accuracy: 0.2532
poch 17/20
33/233 ━━━━━━━━━━ 78s 334ms/step - accuracy: 0.1757 - loss: 3.2838 - val_accuracy: 0.2759
poch 18/20
33/233 ━━━━━━━━━━ 81s 329ms/step - accuracy: 0.1870 - loss: 3.2418 - val_accuracy: 0.2672
poch 19/20
33/233 ━━━━━━━━━━ 83s 334ms/step - accuracy: 0.1872 - loss: 3.1863 - val_accuracy: 0.2716
poch 20/20
33/233 ━━━━━━━━━━ 88s 360ms/step - accuracy: 0.1940 - loss: 3.1631 - val_accuracy: 0.2769

```

### coarse-grained (4 kelas)

```

233/233 ━━━━━━━━━━ 78s 336ms/step - accuracy: 0.8426 - loss: 0.4059 - val_accuracy: 0.862
Epoch 4/20
233/233 ━━━━━━━━━━ 77s 330ms/step - accuracy: 0.8572 - loss: 0.3629 - val_accuracy: 0.869
Epoch 5/20
233/233 ━━━━━━━━━━ 82s 331ms/step - accuracy: 0.8721 - loss: 0.3369 - val_accuracy: 0.869
Epoch 6/20
233/233 ━━━━━━━━━━ 82s 332ms/step - accuracy: 0.8711 - loss: 0.3400 - val_accuracy: 0.880
Epoch 7/20
233/233 ━━━━━━━━━━ 76s 327ms/step - accuracy: 0.8807 - loss: 0.3178 - val_accuracy: 0.882
Epoch 8/20
233/233 ━━━━━━━━━━ 77s 329ms/step - accuracy: 0.8866 - loss: 0.3064 - val_accuracy: 0.867
Epoch 9/20
233/233 ━━━━━━━━━━ 82s 328ms/step - accuracy: 0.8974 - loss: 0.2850 - val_accuracy: 0.876
Epoch 10/20
233/233 ━━━━━━━━━━ 77s 329ms/step - accuracy: 0.8879 - loss: 0.2953 - val_accuracy: 0.876
Epoch 11/20
233/233 ━━━━━━━━━━ 82s 329ms/step - accuracy: 0.8976 - loss: 0.2747 - val_accuracy: 0.879
Epoch 12/20
233/233 ━━━━━━━━━━ 75s 323ms/step - accuracy: 0.8848 - loss: 0.2808 - val_accuracy: 0.868

```

**Gambar 4.5.** Perbandingan Model Coarse dan Fine

## 4.5. Model Evaluation

Evaluasi dan prediksi model dilakukan untuk membandingkan kinerja sistem klasifikasi citra fashion pada dua skenario yang berbeda, yaitu fine-grained classification (64 kelas) dan coarse-grained classification (4 kelas). Kedua skenario menggunakan arsitektur EfficientNetB0, pipeline dataset, serta strategi pelatihan yang sama, sehingga perbedaan hasil yang diperoleh dapat dikaitkan secara langsung dengan perbedaan jumlah dan granularitas label. Evaluasi model menggunakan kode program 4.12.

<b>Kode Program 4.12.</b> Model Evaluation	
Fine-Grained (64 Class)	Coarse-Grained (4 Class)
<pre> test_ds      = df_to_dataset(test_df, training=False)  test_loss,      test_acc      = model.evaluate(test_ds, </pre>	<pre> # Evaluasi model coarse-grained cg_test_ds      = cg_df_to_dataset(cg_test_df, training=False) </pre>

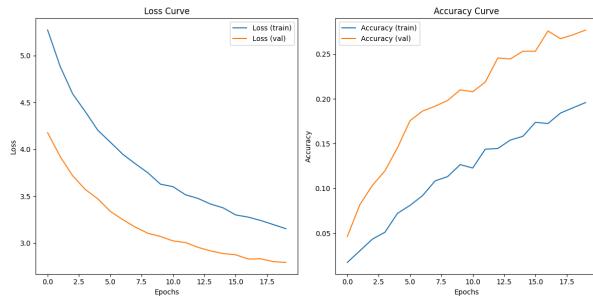
<pre> steps=len(test_ds)) print(f"Test Loss: {test_loss}") print(f"Test Accuracy: {test_acc}") </pre>	<pre> cg_test_loss,           cg_test_acc      = cg_model.evaluate(     cg_test_ds,     steps=len(cg_test_ds) )  print(f"CG    Test    Loss      : {cg_test_loss:.4f}") print(f"CG    Test    Accuracy   : {cg_test_acc:.4f}") </pre>
-------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Pada skenario fine-grained (64 kelas), hasil evaluasi menunjukkan bahwa model hanya mampu mencapai akurasi pengujian sebesar 27,66%. Nilai ini mengindikasikan bahwa model mengalami kesulitan dalam membedakan kategori fashion yang sangat spesifik dan memiliki kemiripan visual tinggi. Classification report memperlihatkan nilai precision dan recall yang rendah pada sebagian besar kelas, bahkan beberapa kelas tidak berhasil dikenali sama sekali. Pola ini mengindikasikan bahwa meskipun model mampu meminimalkan kesalahan pada data latih, proses generalisasi masih belum optimal akibat tingginya kompleksitas klasifikasi dan kemiripan visual antar kelas. Kondisi ini sejalan dengan hasil evaluasi kuantitatif yang menunjukkan akurasi pengujian yang rendah pada skenario fine-grained.

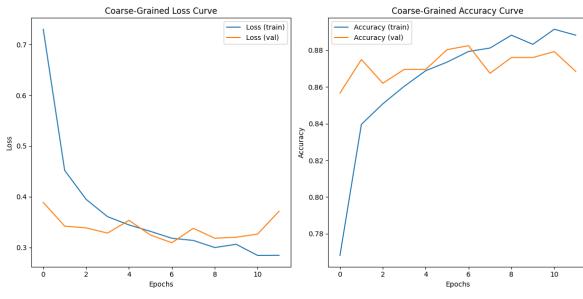
Sebaliknya, pada skenario coarse-grained (4 kelas), model menunjukkan peningkatan performa yang sangat signifikan dengan akurasi pengujian sebesar 88,27%. Penyederhanaan label ke dalam empat kelas utama memungkinkan model untuk mempelajari pola visual yang lebih konsisten dan jelas, sehingga fitur yang diekstraksi oleh EfficientNetB0 dapat dimanfaatkan secara optimal. Classification report pada skenario ini menunjukkan nilai precision, recall, dan F1-score yang tinggi, khususnya pada kelas dengan jumlah data yang lebih besar seperti *Blouse* dan *Blazer*. Jarak antara kedua kurva relatif kecil dan tidak menunjukkan divergensi yang signifikan, yang menandakan bahwa model mampu belajar secara konsisten tanpa mengalami overfitting yang berlebihan. Stabilitas kurva loss ini memperkuat hasil evaluasi yang menunjukkan performa prediksi yang tinggi pada data uji.

Visualisasi kurva loss digunakan untuk mengamati dinamika proses pembelajaran model selama pelatihan serta untuk mengidentifikasi potensi permasalahan seperti overfitting atau underfitting. Berdasarkan kurva loss yang dihasilkan pada gambar 4.6, terlihat perbedaan pola pembelajaran yang cukup jelas antara skenario fine-grained (64 kelas) dan coarse-grained (4 kelas).

fine-grained (64 kelas)



coarse-grained (4 kelas)



**Gambar 4.6.** Visualisasi Loss Curve

Secara keseluruhan, visualisasi kurva loss memperjelas bahwa perbedaan performa antara klasifikasi fine-grained dan coarse-grained tidak hanya tercermin dari nilai akurasi akhir, tetapi juga dari karakteristik proses pembelajaran model. Kurva loss pada skenario coarse-grained menunjukkan proses optimasi yang lebih efektif, sedangkan pada skenario fine-grained mengindikasikan keterbatasan model dalam menangani kompleksitas label yang tinggi.

## 4.6. Prediction Data Test

### 4.6.1. Classification Report

Analisis *classification report* dilakukan untuk mengevaluasi kinerja model secara lebih rinci melalui metrik precision, recall, dan F1-score pada skenario fine-grained (64 kelas) dan coarse-grained (4 kelas). Pada klasifikasi 64 kelas, hasil *classification report* pada tabel 4.1 menunjukkan bahwa sebagian besar kategori memiliki nilai precision dan recall yang rendah, bahkan beberapa kelas tidak terprediksi sama sekali. Kondisi ini mengindikasikan bahwa model mengalami kesulitan dalam mengenali kategori fashion yang sangat spesifik dan memiliki kemiripan visual tinggi. Rendahnya nilai recall pada banyak kelas menunjukkan bahwa cukup banyak data uji yang seharusnya masuk ke suatu kategori tertentu justru diprediksi sebagai kategori lain, sehingga menurunkan performa keseluruhan model.

**Tabel 4.1.** Classification Report 64 Kelas

Kelas	Precision	Recall	F1-Score	Support
Striped_Chiffon_Blouse	0.29	0.27	0.28	15
Classic_Denim_Blazer	0.00	0.00	0.00	14
Stripe_Print_Surplice_Blouse	0.32	0.54	0.40	13
...	...	...	...	...
Zip_Collar_Bomber	0.12	0.14	0.13	14
Petal-Sleeved_Blouse	0.11	0.14	0.12	7

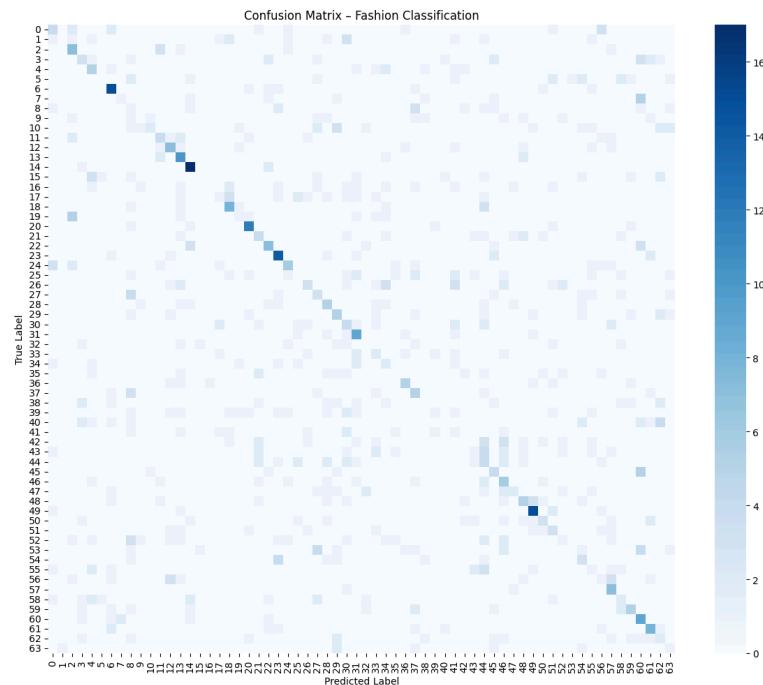
Sebaliknya, pada klasifikasi 4 kelas, *classification report* pada tabel 4.2 menunjukkan peningkatan performa yang sangat signifikan. Nilai precision, recall, dan F1-score pada sebagian besar kelas berada pada tingkat yang tinggi, khususnya pada kelas *Blouse* dan *Blazer* yang memiliki jumlah data lebih banyak. Weighted average F1-score yang tinggi menunjukkan bahwa model mampu melakukan klasifikasi secara seimbang dan konsisten. Meskipun demikian, kelas *Coat* dan *Jacket* masih menunjukkan nilai recall yang relatif lebih rendah, yang mengindikasikan adanya kesalahan prediksi pada kelas-kelas dengan karakteristik visual yang saling mendekati.

**Tabel 4.2.** Classification Report 4 Kelas

Kelas	Precision	Recall	F1-Score	Support
Blazer	0.8444	0.8669	0.8555	338
Blouse	0.9156	0.9375	0.9264	544
Coat	0.6250	0.3125	0.4167	16
Jacket	0.7059	0.3871	0.5000	31

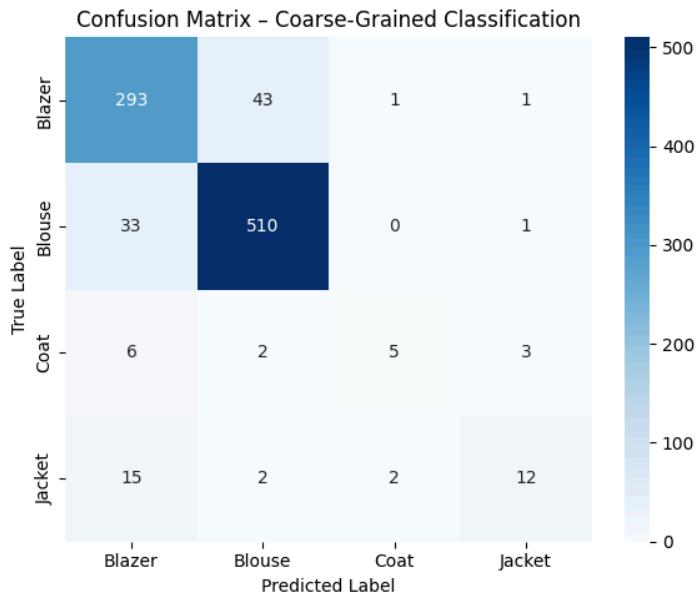
#### 4.6.2. Confusion Matrix

Analisis *confusion matrix* digunakan untuk mengamati pola kesalahan klasifikasi yang terjadi pada data uji serta hubungan antar kelas yang sering tertukar. Pada Gambar 4.7, *confusion matrix* memperlihatkan bahwa prediksi benar tidak mendominasi diagonal utama. Kesalahan prediksi tersebar luas di berbagai kelas, menunjukkan bahwa model sering kali keliru dalam membedakan kategori fashion yang memiliki perbedaan visual sangat halus. Pola ini menegaskan bahwa pada klasifikasi 64 kelas, batas keputusan antar kategori menjadi sangat sempit dan sulit dipelajari oleh model dengan keterbatasan jumlah data per kelas.



**Gambar 4.7.** Confusion Matrix 64 Kelas

Sebaliknya, pada Gambar 4.8 coarse-grained (4 kelas), *confusion matrix* menunjukkan dominasi prediksi benar pada diagonal utama, yang mengindikasikan bahwa sebagian besar citra berhasil diklasifikasikan dengan tepat. Kesalahan prediksi yang terjadi cenderung terlokalisasi pada pasangan kelas tertentu, terutama antara *Coat* dan *Jacket*, yang secara visual memiliki kemiripan dalam bentuk dan fungsi pakaian. Pola kesalahan yang terfokus ini menunjukkan bahwa model telah mempelajari batas keputusan antar kelas utama dengan baik, meskipun masih terdapat ambiguitas pada kondisi visual tertentu.



**Gambar 4.8.** Confusion Matrix 4 Kelas

Secara keseluruhan, analisis *confusion matrix* memperkuat hasil *classification report*, di mana klasifikasi 4 kelas menunjukkan performa yang lebih stabil dan terstruktur dibandingkan klasifikasi 64 kelas. Hal ini menegaskan bahwa penyederhanaan label berperan penting dalam meningkatkan keandalan sistem klasifikasi citra fashion.

## **BAB V**

### **KESIMPULAN**

Penelitian ini berhasil mengimplementasikan model klasifikasi citra fashion berbasis deep learning menggunakan arsitektur EfficientNetB0 dengan dataset DeepFashion. Model dikembangkan melalui pendekatan transfer learning dan diuji pada dua skema klasifikasi, yaitu fine-grained dengan 64 kelas dan coarse-grained dengan 4 kelas. Hasil pengujian menunjukkan perbedaan kinerja yang signifikan antara kedua skema tersebut, di mana model pada klasifikasi 64 kelas hanya mencapai akurasi sebesar 27,66% pada data uji. Rendahnya performa ini dipengaruhi oleh tingginya variasi intra-kelas serta kemiripan visual antar kategori pakaian yang bersifat spesifik, sehingga meningkatkan kompleksitas proses klasifikasi.

Sebaliknya, pada skema klasifikasi 4 kelas, model mampu mencapai akurasi data uji sebesar 88,27%, yang menunjukkan bahwa penyederhanaan granularitas label dapat meningkatkan stabilitas dan kemampuan generalisasi model. Hasil ini menegaskan bahwa EfficientNetB0 efektif dalam mengekstraksi fitur visual utama ketika permasalahan klasifikasi memiliki batas antar kelas yang lebih jelas. Dengan efisiensi parameter dan performa yang kompetitif pada klasifikasi coarse-grained, EfficientNetB0 dinilai sesuai untuk diimplementasikan pada sistem e-commerce yang membutuhkan kecepatan inferensi dan keterbatasan sumber daya, sekaligus memberikan dasar evaluasi bagi pengembangan model klasifikasi fashion dengan tingkat granularitas yang lebih kompleks di masa mendatang.

## DAFTAR PUSTAKA

- Ardana, W. M., & Kusrini, K. (2025). Optimasi algoritma convolutional neural network dengan arsitektur EfficientNet-B0 dan ResNet-50 untuk klasifikasi jenis sampah. *MALCOM: Indonesian Journal of Machine Learning and Computer Science*, 5(4).
- Edbert Thio, S., & Susilo, J. (2025). Identifikasi pemilahan sampah berbasis algoritma transfer learning CNN menggunakan MobileNetV2 dan EfficientNetB0. *bit-Tech*, 8(1).
- EfficientNet. (2025). Dalam *Wikipedia*. <https://en.wikipedia.org/wiki/EfficientNet>
- Khasanah, U., Cahyanti, O. V., Fatma, I. N., & Agustin, T. (2024, November 23). Klasifikasi model pakaian menggunakan convolutional neural network. Dalam *Prosiding Seminar Nasional AMIKOM Surakarta (SEMNASA) 2024*.
- Liu, Z., Luo, P., Qiu, S., Wang, X., & Tang, X. (2016). DeepFashion: Powering robust clothes recognition and retrieval with rich annotations. Dalam *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (hlm. xxx–xxx).
- Mohamed, A. H., Refaat, M., & Hemeida, A. M. (2021). Image classification based deep learning: A review. *Aswan University Journal of Science and Technology*, 1(2).
- Panggabean, S. A., & Susilawati. (2025, Agustus). Penerapan arsitektur EfficientNet dalam model CNN untuk optimalisasi klasifikasi gambar fashion pada dataset Fashion MNIST. *INCODING: Journal of Informatic and Computer Science Engineering*, 5(2), 281–292.
- Suri, P. A., Setiono, M. A., Andrew, A., & Fajar, M. (2025). Comparative study of CNN-based deep learning models for animal, digit, and flower image classification. *Engineering, Mathematics & Computer Science Journal*, 7(3), 321–326.
- Utami, J. W., Novita, M., & Latifa, K. (2025, Desember). Implementation and comparative analysis of CNN and transfer learning models (EfficientNetB0, MobileNetV2, and ResNet50) for rice leaf disease detection based on digital images. *Journal of Applied Informatics and Computing*, 9(6), 3862–3870.

Wijaya, A. C., Mahbubi, A. B., Januarta, M. F., Syabila, T., & Zakaria, D. (2025, Oktober). Analisis sistematis algoritma convolutional neural network untuk klasifikasi gambar bokeh dan blur: Tinjauan literatur. *Jurnal JTIK (Jurnal Teknologi Informasi dan Komunikasi)*, 9(4), 1403–1418.