BAB II
KAJIAN LITERATUR

2.1 *Artificial Intelligence* (AI)

Artificial Intelligence (AI) atau kecerdasan buatan merupakan cabang ilmu komputer yang berfokus pada pengembangan sistem yang mampu meniru kemampuan kognitif manusia, seperti memahami informasi, belajar dari data, mengenali pola, mengambil keputusan, serta memecahkan masalah. Inti dari AI adalah pembelajaran mesin (ML) sekumpulan algoritma dan metode yang menangani masalah klasifikasi, pengelompokan, dan peramalan [13]. Teknologi ini mencakup berbagai pendekatan yang saling berkaitan dalam hierarki pembelajaran mesin.

Pembelajaran mesin merupakan cabang utama kecerdasan buatan yang bekerja berdasarkan prinsip belajar contoh dengan *deep learning* sebagai bagian di dalamnya, sehingga sistem tidak lagi bergantung pada daftar aturan yang panjang untuk menyelesaikan masalah melainkan belajar langsung dari data. Dalam perkembangannya, jaringan saraf tiruan menjadi komponen penting dalam *machine learning* dan AI karena kemampuannya meniru kerja otak manusia dalam memproses informasi. Teknologi *deep learning* kini menjadi fokus utama di bidang *machine learning*, kecerdasan buatan, data *science*, dan analitik karena mampu mengekstraksi pola kompleks secara otomatis dari data yang tersedia. Kemajuan ini didorong oleh meningkatnya produksi data, daya komputasi yang semakin kuat, serta ketersediaan layanan AI, sehingga perkembangan AI secara keseluruhan sangat dipengaruhi oleh kombinasi *big data*, komputasi modern, dan algoritma yang semakin canggih [14].

2.1.1 *Machine Learning*

Machine learning merupakan salah satu cabang penting dalam *artificial intelligence* (AI) yang berfokus pada pendekatan berbasis data, di mana sistem komputer mampu belajar dari data tanpa campur tangan manusia secara langsung [15]. Pendekatan ini sangat relevan dalam menyelesaikan permasalahan dunia nyata yang umumnya melibatkan jumlah data yang besar, sehingga memerlukan bantuan komputasi untuk mengolah data tersebut menjadi pengetahuan yang bermakna bagi proses pemecahan masalah [16]. Melalui algoritma sebagai komponen utamanya, *machine learning* memungkinkan sistem melakukan prediksi serta beradaptasi secara mandiri tanpa harus diprogram secara eksplisit, yang pada akhirnya

menghasilkan perilaku sistem yang lebih cerdas. Berbagai aplikasi *machine learning* telah diterapkan secara luas di beragam bidang, menegaskan posisinya sebagai subbidang utama dalam *Artificial Intelligence* (AI). Algoritma *machine learning* dapat dikelompokkan dalam empat pembelajaran yang berbeda tergantung pada *output* yang diperkirakan dan tipe *input* [17].

1. *Supervised learning* merupakan pendekatan pembelajaran berbasis data berlabel, di mana setiap sampel memiliki pasangan *input* dan target keluaran yang jelas sehingga model mempelajari fungsi pemetaan dari *input* ke *output*. Contohnya, *input* berupa gambar buah, kemudian label keluaran berupa "apel" atau "pisang", sehingga sistem belajar mengenali karakteristik visual masing-masing kategori. Pendekatan ini efektif untuk tugas prediksi, klasifikasi, dan regresi karena hasil yang diharapkan telah ditentukan sejak awal.
2. *Unsupervised learning* merupakan pendekatan yang menggunakan data tanpa label, sehingga model tidak diberi informasi target, melainkan mengekstraksi pola, struktur, atau pengelompokan secara mandiri dari data. Contohnya, sekumpulan gambar buah tanpa label dikelompokkan otomatis berdasarkan kemiripan bentuk atau warna menjadi beberapa *cluster*, tanpa diketahui sebelumnya mana "apel" atau "pisang". Pendekatan ini umum digunakan untuk *clustering*, deteksi anomali, dan eksplorasi pola tersembunyi.
3. *Semi-supervised* learning menggabungkan sebagian kecil data berlabel dengan sejumlah besar data tidak berlabel. Model dilatih awal menggunakan data berlabel, kemudian memanfaatkan data tidak berlabel untuk memperkaya proses pembelajaran. Contohnya, hanya 100 gambar buah diberi label "apel/pisang", sementara ribuan lainnya tidak berlabel; model memanfaatkan kombinasi keduanya untuk meningkatkan performa klasifikasi. Pendekatan ini relevan ketika pelabelan data mahal atau memakan waktu [18].
4. *Reinforcement learning* merupakan pendekatan berbasis interaksi dengan lingkungan melalui mekanisme aksi dan umpan balik berupa *reward* atau penalti. Model belajar memilih tindakan yang memaksimalkan akumulasi *reward*. Contohnya, agen permainan memilih langkah tertentu; jika langkah benar mendapat poin, jika salah kehilangan poin, sehingga sistem belajar strategi optimal secara bertahap. Pendekatan ini banyak digunakan pada pengambilan keputusan berurutan dan sistem kendali.

Penggunaan *supervised learning* dalam klasifikasi sampah dipilih karena setiap citra telah memiliki label kelas yang jelas, sehingga model dapat belajar hubungan langsung antara fitur visual dan kategori yang sesuai. Pendekatan ini memungkinkan sistem untuk

melakukan prediksi kategori secara akurat berdasarkan data berlabel, memaksimalkan efektivitas pembelajaran, dan mengurangi ambiguitas yang mungkin muncul jika menggunakan metode tanpa label. Dengan *supervised learning*, proses pelatihan dapat diarahkan secara eksplisit untuk mengenali pola spesifik dalam *dataset*, sehingga performa klasifikasi menjadi lebih stabil dan terukur.

2.1.2 *Deep Learning*

Deep learning merupakan cabang utama yang berkembang dari *machine learning* dengan memanfaatkan struktur hierarkis pada jaringan saraf tiruan yang meniru sistem kerja saraf biologis dalam mengolah informasi. Pendekatan ini mempelajari fitur-fitur dasar terlebih dahulu, lalu menggabungkannya secara bertahap untuk membentuk representasi tingkat tinggi, sehingga mampu melakukan tugas seperti klasifikasi maupun regresi citra secara lebih akurat. Berbeda dengan *machine learning konvensional* yang masih banyak bergantung pada rekayasa fitur secara manual, *deep learning* mengandalkan jaringan saraf berlapis untuk mengekstraksi fitur secara otomatis langsung dari data [15].

Kemampuan *deep learning* telah menunjukkan keunggulan signifikan dalam tugas klasifikasi gambar, dengan banyak model yang diusulkan untuk pembelajaran dan klasifikasi representasi gambar [19]. *Deep learning* memanfaatkan arsitektur seperti *Convolutional Neural Networks* (CNN) untuk secara otomatis mempelajari fitur hierarkis dari data gambar mentah, sehingga mencapai akurasi yang lebih tinggi untuk tugas klasifikasi berbasis gambar [20], terutama pada *dataset* skala besar atau tidak terstruktur. Dengan mempelajari pola visual langsung dari citra, model dapat membedakan jenis sampah berdasarkan ciri yang benar-benar muncul pada data. Berikut adalah penjelasan tentang bagaimana *deep learning* berfungsi dan mengapa teknologi ini sangat efektif dalam berbagai aplikasi [21].

1. Jaringan Saraf *Deep learning*: Terdiri dari lapisan *neuron* yang saling terhubung, masing-masing lapisan berfungsi untuk mengekstraksi dan mengoptimalkan fitur dari data, meningkatkan akurasi prediksi atau klasifikasi.
2. Lapisan *Input* dan *Output*: *Layer input* menerima data mentah, sementara *layer output* menghasilkan hasil akhir. Di antaranya terdapat lapisan tersembunyi yang melakukan transformasi data secara bertahap.
3. *Forward Propagation*: Data diteruskan dari *layer input* ke *layer output* melalui lapisan-lapisan tersembunyi. Setiap *neuron* menghitung *output* menggunakan bobot dan bias, kemudian mengirim hasilnya ke *neuron* berikutnya hingga tercapai prediksi akhir.

4. *Backpropagation*: Algoritma seperti *gradient descent* digunakan untuk menghitung kesalahan antara prediksi dan nilai sebenarnya. Kesalahan ini dipropagasikan mundur melalui jaringan untuk menyesuaikan bobot dan bias, dengan tujuan meminimalkan kesalahan dan meningkatkan akurasi model.
5. Fungsi *Loss* dan Metrik Validasi: Selama proses pelatihan, parameter model dioptimasi dengan meminimalkan nilai fungsi *loss* pada data *training*, sedangkan kemampuan generalisasi model dipantau melalui *validation loss* dan *validation accuracy* pada data validasi. Berbeda dengan *validation accuracy* yang bersifat diskret (hanya mencerminkan proporsi prediksi benar/salah), *validation loss* dihitung secara kontinu berdasarkan probabilitas prediksi terhadap label sebenarnya, sehingga lebih sensitif dalam mendeteksi indikasi *overfitting*. Proses pelatihan idealnya dihentikan ketika tren *validation loss* berubah dari menurun menjadi meningkat, karena kondisi ini menandakan model mulai mengalami *overfitting* meskipun *validation accuracy* masih terlihat stabil atau bahkan meningkat. Oleh karena itu, *validation loss* secara konvensional dijadikan metrik utama dalam pemantauan mekanisme *EarlyStopping* dan *ModelCheckpoint* pada proses pelatihan *deep learning* [22].

2.1.3 *Transfer Learning*

Transfer learning adalah teknik ampuh yang digunakan dalam tugas *deep learning*. Di sini, model yang dikembangkan untuk tugas tertentu digunakan kembali sebagai titik awal untuk model pada tugas kedua. Dengan demikian, *transfer learning* menggunakan pengetahuan yang diperoleh dari model yang telah dilatih sebelumnya dan memungkinkan konvergensi yang lebih cepat dengan kinerja yang lebih baik, terutama ketika data terbatas. Di sisi lain, *fine-tuning* membawa konsep ini lebih jauh dengan menyesuaikan parameter model yang telah dilatih sebelumnya agar lebih sesuai dengan tugas baru. Kedua teknik ini sangat berguna ketika perlu melatih *deep learning* yang membutuhkan banyak data dan komputasi. Tujuan *transfer learning* untuk meningkatkan kinerja model pada tugas baru dengan memanfaatkan pengetahuan dari model yang sebelumnya telah dilatih pada tugas lain [23].

Caranya model *Convolutional Neural Network* (CNN) yang sudah dilatih dengan *dataset* standar berskala besar digunakan kembali sebagai dasar untuk menyelesaikan tugas target [24]. Model pra-terlatih tersebut berfungsi sebagai ekstraktor fitur karena telah mampu mengenali pola penting pada citra, mulai dari tepi dan tekstur sederhana hingga informasi semantik yang lebih kompleks, sehingga pengetahuan ini dapat langsung dimanfaatkan tanpa

harus melatih model dari awal [25]. Secara umum, proses *transfer learning* dilakukan melalui dua langkah utama:

1. Ekstraksi Fitur: Pada langkah ini, model yang telah dilatih sebelumnya digunakan sebagai ekstraktor fitur tetap. Bobot model yang telah dilatih sebelumnya tetap tidak berubah, dan hanya lapisan pengklasifikasi akhir yang dilatih pada tugas baru. Oleh karena itu, metode ini memanfaatkan fitur-fitur kaya yang dipelajari oleh model yang telah dilatih sebelumnya dari kumpulan data yang besar.
2. Penyempurnaan (*Fine-tuning*, bersifat opsional): beberapa lapisan teratas dari *pretrained* model dicairkan (*unfreeze*), lalu lapisan pengklasifikasi baru dan lapisan teratas dilatih bersama-sama.

Dalam penelitian ini, digunakan pendekatan *transfer learning* dengan EfficientNetB0 sebagai ekstraktor fitur tanpa proses *fine-tuning*. Seluruh bobot pada *backbone* dipertahankan (*base.trainable = False*), sedangkan lapisan akhir diganti dengan *custom classification head* yang dilatih untuk 10 kelas sampah.

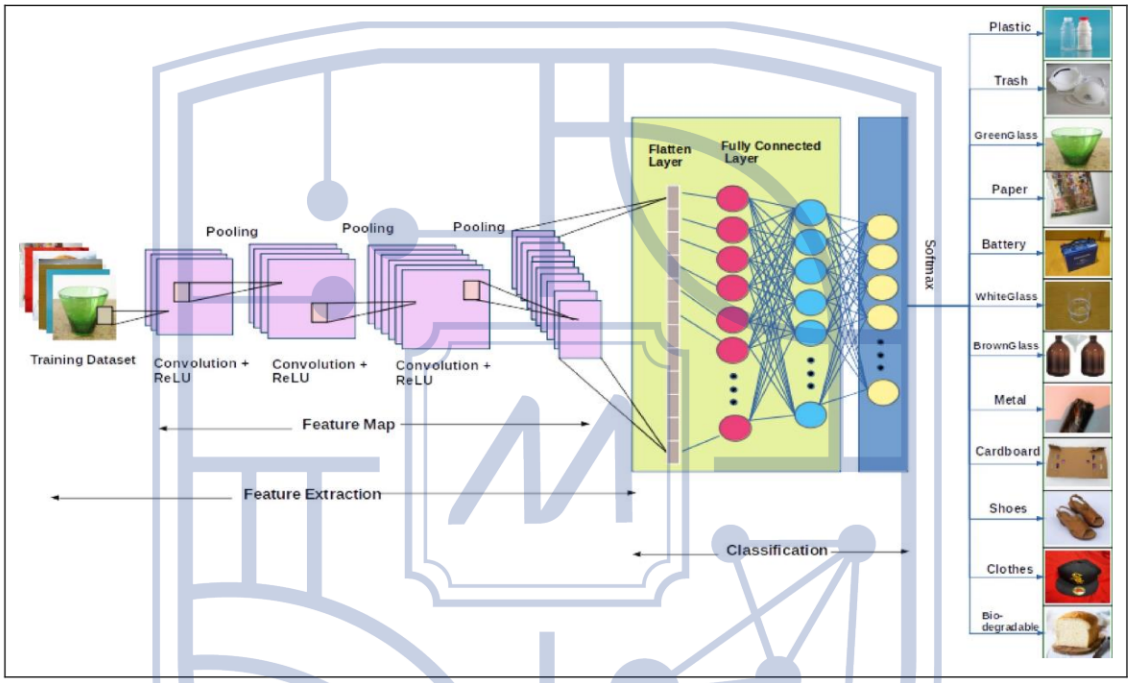
2.2 *Computer Vision*

Computer Vision (CV) merupakan kecerdasan buatan atau *Artificial Intelligence* (AI) yang menggunakan video digital atau urutan gambar untuk mengenali konten, dengan tujuan melatih komputer untuk mengekstrak informasi dari gambar sehingga pada dasarnya memungkinkan komputer untuk 'melihat' dan mengenali konten visual [26]. Teknologi *Computer Vision* mencakup domain multidisiplin yang mengintegrasikan teknik pembelajaran mesin (*machine learning*) lanjutan, pengenalan pola, dan pemrosesan gambar untuk memberdayakan komputer memahami konten visual yang terdapat dalam gambar dan video, dengan *Convolutional Neural Networks* (CNN) yang menunjukkan performa tingkat ahli dalam klasifikasi gambar, deteksi objek, dan segmentasi semantik [27]. *Model deep learning* dalam *Computer Vision* terbukti sangat efektif karena kemampuannya mengidentifikasi pola dan struktur kompleks dalam gambar, menjadikannya sangat efektif untuk berbagai tugas pemrosesan visual dan pengambilan keputusan berdasarkan data visual yang diproses [21].

2.2.1 *CNN*

Convolutional Neural Networks (CNN) telah menunjukkan kemajuan luar biasa dalam berbagai bidang *Computer Vision*, termasuk klasifikasi citra, segmentasi semantik, deteksi objek, dan rekonstruksi super-resolusi citra. CNN memiliki keunggulan dalam

pembelajaran dan ekspresi otomatis, di mana ekstraksi fitur dari data masukan asli dapat direalisasikan melalui pelatihan model CNN yang sesuai dengan aplikasi praktis [27]. Arsitektur CNN terus berkembang menjadi lebih kompleks dan beragam seiring kemajuan teknologi *deep learning*, sehingga secara bertahap menggantikan metode *machine learning* tradisional dalam tugas-tugas pengolahan citra. Sebagai pendekatan analisis citra tingkat lanjut, (CNN) telah menjadi alat yang sangat diperlukan untuk menemukan pola tersembunyi dalam fitur visual [28].

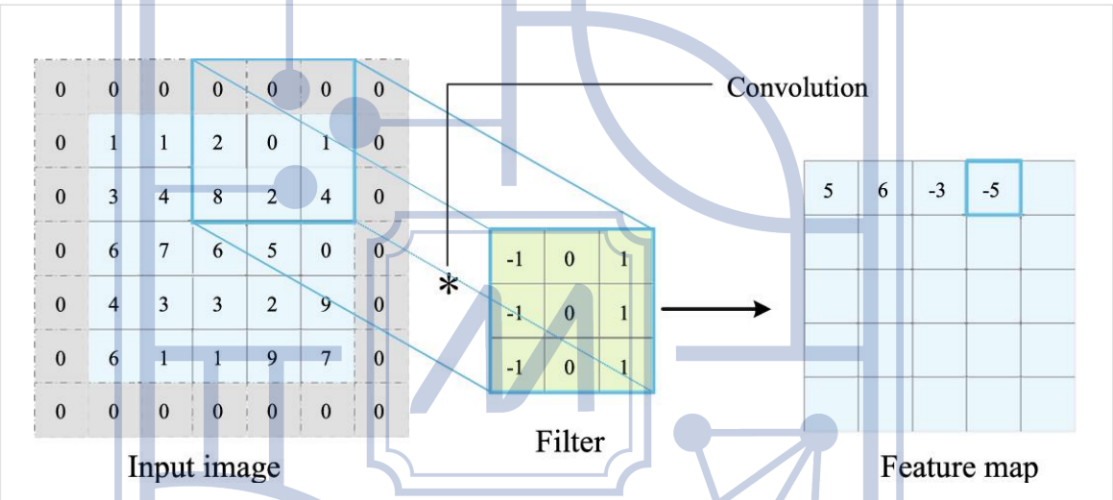


Gambar 2.1 Arsitektur *Convolutional Neural Network* (CNN) [4]

1. Arsitektur: CNN terdiri dari lapisan-lapisan yang bekerja secara bertahap. Pertama, *convolutional layer* menerapkan filter atau *kernel* yang digeser ke seluruh area citra setiap pergeseran mengalikan nilai piksel dengan nilai filter lalu menjumlahkan hasilnya menjadi satu piksel baru, sehingga terbentuk peta fitur (*feature map*) yang menangkap pola visual seperti tepi dan tekstur. Filter-filter ini tidak ditentukan secara manual melainkan dipelajari sendiri oleh model melalui *backpropagation.* Setelah konvolusi, fungsi aktivasi ReLU diterapkan dengan menghilangkan semua nilai negatif pada peta fitur, sehingga model dapat mempelajari pola yang lebih kompleks. Kemudian, *pooling layer* mengecilkan ukuran peta fitur dengan memilih nilai tertinggi dari setiap kelompok piksel (*max pooling*), sehingga hanya piksel paling penting yang dipertahankan dan komputasi menjadi lebih ringan [20]. Rangkaian konvolusi–ReLU–*pooling* ini diulang beberapa kali hingga ukuran citra cukup kecil. Selanjutnya, *flatten layer* meratakan peta

fitur yang berbentuk matriks menjadi satu baris data (vektor), lalu *fully connected layer* menghubungkan seluruh data tersebut layaknya jaringan saraf biasa. Terakhir, fungsi *softmax* menghitung dan mencocokkan data tersebut ke setiap kategori sampah, menghasilkan klasifikasi akhir berdasarkan pola yang telah dipelajari [4].

2. Mekanisme Konvolusi: Filter digeser ke seluruh area citra dengan langkah (*stride*) tertentu, dan teknik *padding* diterapkan agar ukuran citra tidak berkurang setelah konvolusi. Nilai dalam setiap filter diperbarui secara otomatis melalui *gradient descent*, Secara matematis, operasi konvolusi pada setiap posisi spasial (i, j) dari *feature map* dinyatakan sebagai berikut [29]:



Gambar 2.2 Proses *Convolutional Neural Network* (CNN) [27]

$$F(i,j) = \sum_{m}^{k-1} \sum_{n}^{k-1} P(i+m, j+n) * K(m,n) + b \quad (1)$$

Dimana:

F(i,j) = Nilai *output* pada posisi (i, j).
P = Matriks citra setelah *zero-padding*.
K = Matriks *kernel* berukuran k×k.
m dan n = Indeks iterasi pada dimensi *kernel*.
b = Nilai bias.

Ukuran *feature map* yang dihasilkan bergantung pada dimensi *input*, ukuran filter, jumlah *padding*, dan *stride* yang digunakan, dengan rumus sebagai berikut [29]:

$$W_{out} = \frac{(W - F + 2P)}{S} + 1 \quad (2)$$

Dimana:

W = Lebar (atau tinggi) citra *input*.

F = Ukuran filter.

P = Jumlah *padding*.

S = *Stride*

Dengan menerapkan *zero-padding* (P = 1) pada konvolusi 3×3 dengan *stride* 1, ukuran *feature map* yang dihasilkan akan identik dengan ukuran *input* sehingga informasi spasial pada tepi citra tetap terjaga sehingga model menemukan sendiri filter terbaik tanpa rekayasa fitur manual.

3. Keunggulan: CNN mampu mengekstraksi fitur visual secara otomatis, efisien secara komputasi karena satu filter dipakai bersama (parameter *sharing*) di seluruh area citra, serta bersifat *translation invariant* model tetap dapat mengenali objek meskipun posisinya bergeser dalam citra, berkat kombinasi konvolusi dan *pooling*.

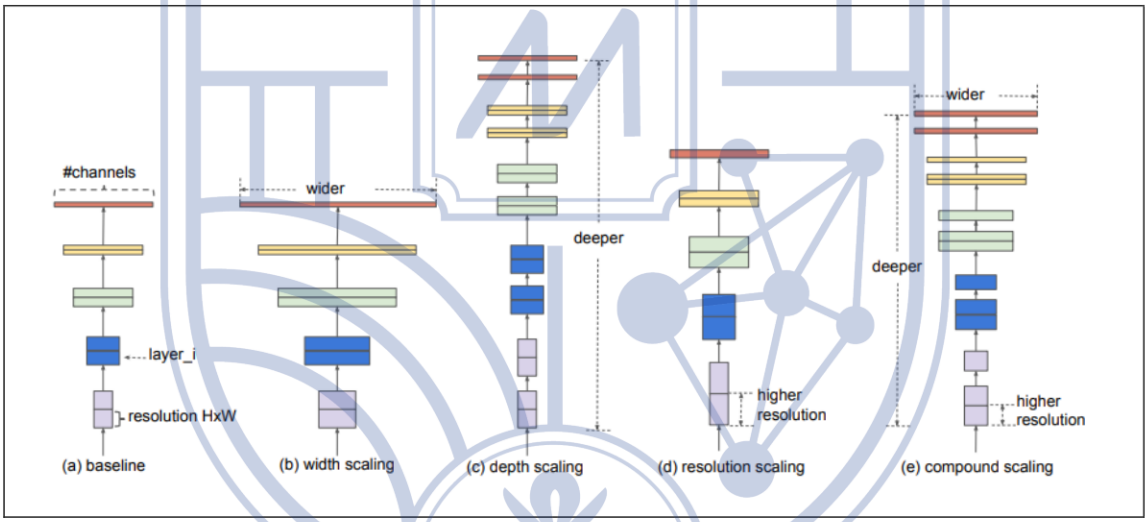
Dalam konteks klasifikasi gambar sampah, CNN menjadi pilihan yang sangat tepat karena mampu mengenali pola visual yang beragam secara otomatis, mulai dari tekstur permukaan plastik, kilap logam, hingga serat kertas dan kardus. Kemampuan ekstraksi fitur secara bertahap memungkinkan model membedakan kategori sampah yang memiliki kemiripan visual tinggi, sesuatu yang sulit dicapai metode *machine learning* konvensional. Dengan demikian, CNN memberikan fondasi yang kuat dan terbukti efektif sebagai inti sistem klasifikasi sampah otomatis berbasis citra digital.

2.2.2 *EfficientNet*

EfficientNet, merupakan keluarga model klasifikasi gambar dengan kemampuan ekstraksi fitur yang sangat baik, mampu mencapai akurasi tinggi dengan jumlah parameter yang lebih sedikit sehingga lebih ringan dalam penggunaan memori dan CPU. Efisiensi ini menjadikannya solusi praktis untuk menghasilkan kinerja klasifikasi yang kompetitif tanpa beban komputasi yang besar [30]. Model EfficientNet mencakup EfficientNetB0 hingga EfficientNetB7, dengan B0 sebagai dasar dan B7 mewakili varian terbesar dan paling ampuh. Setiap model selanjutnya dikembangkan dengan memperkuat B0 melalui rumus *compound scaling*. EfficientNetB0 adalah model pertama yang dihasilkan dari *Neural Architecture Search* (NAS), berfungsi sebagai model utama dalam garis keturunan EfficientNet, meningkatkan konfigurasinya untuk presisi dan efisiensi komputasi. EfficientNetB0, dengan hanya 5,3 juta parameter, mencapai akurasi *top-1* pada *ImageNet*,

melampaui jaringan yang lebih dalam sambil mempertahankan biaya pemrosesan yang lebih rendah [31].

EfficientNet adalah arsitektur jaringan saraf konvolusional yang diusulkan oleh *Google* pada tahun 2019, dirancang untuk mencapai kinerja dan efisiensi yang lebih tinggi dalam klasifikasi gambar dengan mengoptimalkan struktur jaringan dan pemanfaatan parameter. EfficientNet menggabungkan teknik-teknik seperti konvolusi terpisah kedalaman, konvolusi kelompok dan Pencarian Arsitektur *Neural* (NAS) [32]. EfficientNet menggunakan metode *Compound scaling* yang menyeimbangkan kedalaman, lebar, dan resolusi jaringan secara bersamaan, sehingga akurasi meningkat tanpa menambah komputasi berlebihan. Arsitekturnya dioptimalkan melalui *neural architecture search*, sehingga lebih efisien dibanding CNN biasa. EfficientNet terbukti sangat kuat untuk *transfer learning* dan mampu melakukan generalisasi dengan baik pada berbagai tugas *Computer Vision*, sehingga efektif digunakan untuk klasifikasi jenis sampah. Karena itu EfficientNet memberikan kombinasi terbaik antara akurasi tinggi, model ringan, dan *training* cepat [33].



Gambar 2.3 Penskalaan Model EfficientNet [8]

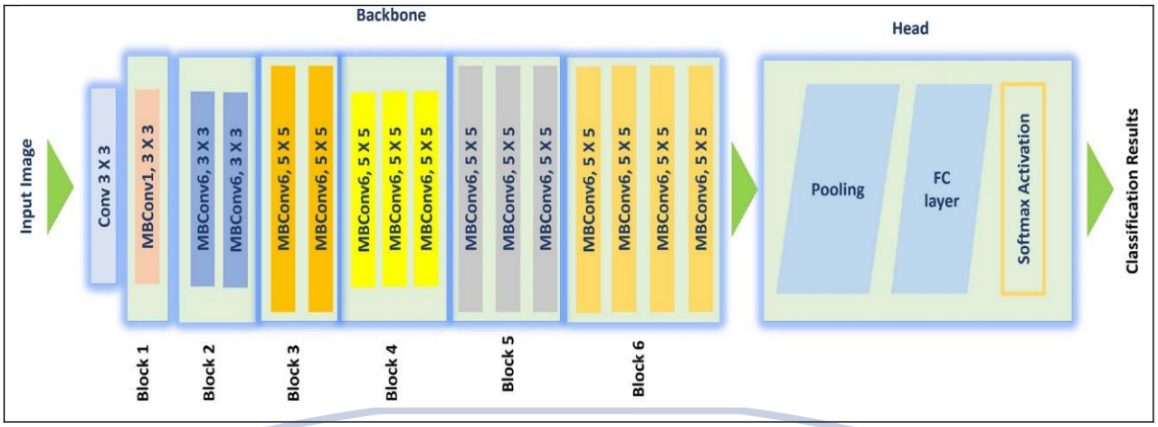
Gambar 2.3 memperlihatkan ilustrasi penskalaan model [8]:

1. Model dasar EfficientNetB0 belum menggunakan *scaling* digunakan sebagai titik awal arsitektur dasar yang sudah optimal, dengan ukuran standar, kedalaman standar, lebar standar dan resolusi *input* standar sebelum dilakukan proses peningkatan skala.
2. *Width scaling* dilakukan dengan menambah jumlah filter pada tiap *layer* sehingga jaringan menjadi lebih lebar dan mampu menampung lebih banyak informasi, namun jika diterapkan sendiri berisiko meningkatkan *overfitting*.

3. *Depth scaling* menambah jumlah *layer* agar model semakin dalam sehingga dapat mempelajari pola yang lebih kompleks, tetapi konsekuensinya waktu pelatihan lebih lama dan potensi masalah gradien meningkat.
4. *Resolution scaling* menaikkan resolusi gambar *input* sehingga detail visual lebih kaya, yang membantu model mengenali fitur dengan lebih akurat.
5. *Compound scaling* EfficientNetB1-B7 menggunakan *(compound scaling*) menjadi strategi utama EfficientNet, yaitu meningkatkan *depth, width,* dan *resolution* secara bersamaan dengan rasio terkontrol, sehingga kinerja model naik signifikan sekaligus tetap efisien secara komputasi. Metode ini memilih koefisien komposit $\phi = 0$ EfficientNetB0 dan versi EfficientNetB1-B7 $\phi = 1\text{-}7$ untuk menskalakan lebar, kedalaman, dan resolusi jaringan secara proporsional dan terstruktur.

2.2.3 *EfficientNetB0*

EfficientNetB0 merupakan varian dasar dari keluarga EfficientNet yang menjadi fondasi bagi pengembangan varian lainnya (B1 hingga B7) EfficientNetB0, telah mendapatkan banyak perhatian karena kemampuannya untuk berkinerja baik dan menggunakan sumber daya secara efisien saat mengklasifikasikan gambar. Arsitektur ini dirancang secara otomatis menemukan susunan jaringan paling optimal. Setiap *block* MBConv pada EfficientNetB0 secara bawaan menerapkan *depthwise separable convolution* sekaligus mekanisme *Squeeze-and-Excitation*, dua komponen inti yang membuat arsitektur ini mampu mengekstraksi fitur visual secara selektif dengan jumlah parameter yang jauh lebih kecil dibandingkan arsitektur CNN konvensional, yang pada dasarnya menyesuaikan ukuran dan kompleksitas model untuk menemukan keseimbangan yang sempurna. Ini berarti arsitektur ini dapat memberikan akurasi terbaik tanpa mengonsumsi terlalu banyak daya komputasi, sehingga cocok untuk model AI yang terintegrasi dengan perangkat [34]. Dengan EfficientNetB0 sebagai model dasar, yang diluncurkan dari pencarian arsitektur neural (NAS), yang memaksimalkan kepadatan arsitektur sambil memberikan efisiensi yang sangat tinggi. Meskipun hanya memiliki 5,3 juta parameter, EfficientNetB0 memberikan hasil yang tinggi dalam tugas klasifikasi gambar, hemat energi, dan bahkan melampaui model-model unggulan, seperti ResNet50 pada *dataset ImageNet*. Model ini terdiri dari *block* konvolusi *bottleneck* terbalik *mobile* (MBConv) dengan penambahan mekanisme *Squeeze-and-Excitation* (SE) pada setiap *block*, konvolusi kedalaman, dan fungsi aktivasi *Swish*, yang semuanya memungkinkan model untuk memilih representasi fitur yang lebih detail dengan biaya komputasi yang jauh lebih rendah [35].



Gambar 2.4 EfficientNetB0 *Architecture* [12]

Gambar 2.4 menunjukkan alur lengkap arsitektur EfficientNetB0 yang terbagi menjadi dua bagian utama: *Backbone* (pengekstrak fitur) dan *Head* (pengklasifikasi). Berikut penjelasan setiap tahapnya [12]:

1. *Input Image* → *Conv* 3×3. Gambar masukan berukuran 224×224×3 piksel pertama-tama dilewatkan melalui satu lapisan konvolusi standar berukuran 3×3. Tahap ini bertugas mengekstraksi fitur paling dasar dari gambar, seperti tepi dan tekstur awal, sebelum memasuki *block* inti.
2. *Block* 1–6: MBConv (*Mobile Inverted Bottleneck Convolution*) Inilah inti dari EfficientNetB0. Terdapat enam *block* berurutan, masing-masing terdiri dari beberapa lapisan MBConv dengan ukuran *kernel* yang berbeda (3×3 pada *block* awal dan 5×5 pada *block* berikutnya). Setiap *block* MBConv bekerja dengan alur berikut:
    a. Ekspansi *channel*: Jumlah *channel* diperbanyak terlebih dahulu agar jaringan dapat menangkap lebih banyak pola sekaligus. Ekspansi faktor 6 (MBConv6) berarti jumlah *channel* dikalikan 6 sebelum diproses.
    b. *Depthwise Convolution* (3×3 atau 5×5): Berbeda dari konvolusi biasa yang memproses semua *channel* sekaligus, *depthwise convolution* memproses setiap *channel* secara terpisah. Ini membuat komputasi jauh lebih ringan tanpa mengorbankan kualitas fitur yang ditangkap.
    c. *Batch Normalization* (BN) adalah teknik yang diterapkan setelah lapisan konvolusi untuk menstabilkan distribusi nilai aktivasi pada setiap *mini-batch* pelatihan. Secara formal, BN menormalkan setiap nilai fitur skalar agar memiliki rata-rata nol dan variansi satu menggunakan statistik yang dihitung dari *mini-batch*, kemudian menerapkan transformasi linier dengan parameter yang dapat dipelajari γ dan β untuk

mempertahankan kapasitas representasi jaringan. Rumus lengkap *batch normalization* adalah sebagai berikut [36]:

$$BN(x) = \gamma \cdot \left(\frac{(x - \mu)}{\sqrt{\sigma^2 + \varepsilon}}\right) + \beta \quad (3)$$

Dimana:

μ = rata-rata *mini-batch*

σ^2 = variansi *mini-batch*

ε = konstanta kecil untuk kestabilan numerik agar tidak terjadi pembagian dengan nol

γ = parameter skala

β = parameter pergeseran yang keduanya dipelajari melalui *backpropagation*.

Batch Normalization terbukti memitigasi masalah *vanishing* dan *exploding gradient* dengan menstabilkan aktivasi, memungkinkan penggunaan *learning rate* yang lebih besar, dan mengurangi ketergantungan pada inisialisasi bobot yang hati-hati. Setelah ini diterapkan fungsi aktivasi *Swish*.

d. EfficientNetB0 menggunakan fungsi aktivasi *Swish* pada setiap lapisan konvolusinya. *Swish* merupakan fungsi aktivasi adaptif yang menggabungkan *input* dengan fungsi *sigmoid*, sehingga mampu mempertahankan nilai negatif kecil dan menghasilkan gradien yang lebih halus dibandingkan ReLU. Fungsi *sigmoid* yang menjadi komponen *Swish* didefinisikan sebagai berikut [37]:

$$\sigma(x) = x \cdot \left(\frac{1}{1 + e^{\{-x\}}}\right) \quad (4)$$

Berdasarkan fungsi *sigmoid* tersebut, fungsi aktivasi *Swish* didefinisikan sebagai berikut:

$$Swish(x) = x \cdot \sigma(x) = x \cdot \left(\frac{1}{1 + e^{\{-x\}}}\right) \quad (5)$$

Dimana:

x = Nilai *input*

σ(x) = fungsi *sigmoid*

e = konstanta, e ≈ 2.71828

γ = parameter skala

β = parameter pergeseran yang keduanya dipelajari melalui *backpropagation*.

e. Mekanisme *Squeeze-and-Excitation* (SE): Seluruh peta fitur dipadatkan menjadi satu nilai per *channel*, lalu jaringan kecil menghitung seberapa penting tiap *channel*

tersebut. *Channel* yang penting diperkuat dan yang kurang relevan dilemahkan. Mekanisme ini membuat model lebih fokus pada fitur yang benar-benar bermakna.

f. Proyeksi & *Skip Connection*: Jumlah *channel* dikembalikan ke ukuran semula, lalu *input* awal dihubungkan langsung ke *output* (*skip connection*) agar informasi asli tidak hilang saat pelatihan berlangsung. Setiap *block* yang semakin dalam menghasilkan fitur yang semakin kompleks dan abstrak dari sekadar tepi dan tekstur di *block* awal, hingga pola bentuk objek yang lebih bermakna di *block* akhir.

3. *Head: Pooling → FC Layer → Softmax → Classification Results*

Setelah backbone selesai mengekstraksi fitur:

a. *Global Average Pooling (GAP)*: Setelah seluruh blok konvolusi EfficientNetB0 menghasilkan *feature map* 7×7×1280, dimensi spasial setiap saluran diratakan menjadi satu nilai skalar menggunakan operasi *Global Average Pooling*, Seluruh peta fitur yang masih berbentuk matriks dipadatkan menjadi satu vektor tunggal dengan merata-ratakan setiap peta fitur [38].

$$GAP(z) = \left(\frac{1}{(H \times W)}\right) \sum_{i=1}^{H} \sum_{j=1}^{W} x_{i,j,k} \quad (6)$$

Dimana:

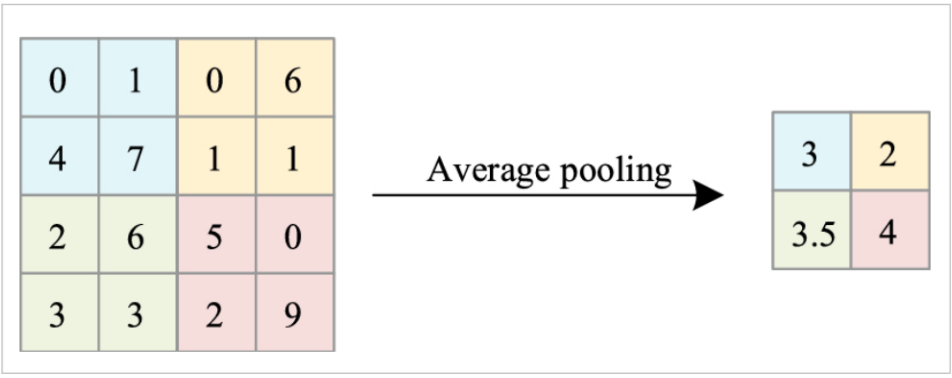
H dan W = Tinggi dan lebar *feature map*

x{i,j,k} = Nilai piksel pada posisi (i, j) pada *channel* k

z = indeks *channel output*

Pendekatan ini secara efektif mereduksi *tensor* berukuran H×W×C menjadi vektor fitur berdimensi C sehingga menghasilkan vektor fitur $f \in \mathbb{R}^{1280}$ yang selanjutnya diteruskan ke lapisan *Dense* untuk klasifikasi

b. *Fully Connected (FC) Layer:* Vektor fitur hasil GAP selanjutnya diteruskan ke lapisan *Dense* yang menerapkan transformasi linier untuk menghasilkan representasi fitur baru. Operasi transformasi linier pada lapisan *Dense* didefinisikan sebagai berikut [12], [39].



Gambar 2.5 Average *Pooling* [27]

$$z = W^1 \cdot x + b^1, \tag{7}$$

Dengan:

$$h = ReLU(z) = max(0, z) \tag{8}$$

Dimana:

W^1 = Matriks bobot yang dapat dilatih

x = vektor fitur *input* dari GAP

b^1 = vektor bias

ReLU meneruskan nilai positif secara langsung dan memotong seluruh nilai negatif menjadi nol, sehingga memperkenalkan non-linieritas tanpa menambah beban komputasi yang signifikan. Untuk mencegah *overfitting* selama pelatihan, diterapkan L2 regularisasi yang menambahkan penalti berbasis kuadrat bobot ke dalam fungsi *loss*, didefinisikan sebagai berikut [12]:

$$L\_L2 = \lambda \times \Sigma_i {w_i}^2 \tag{9}$$

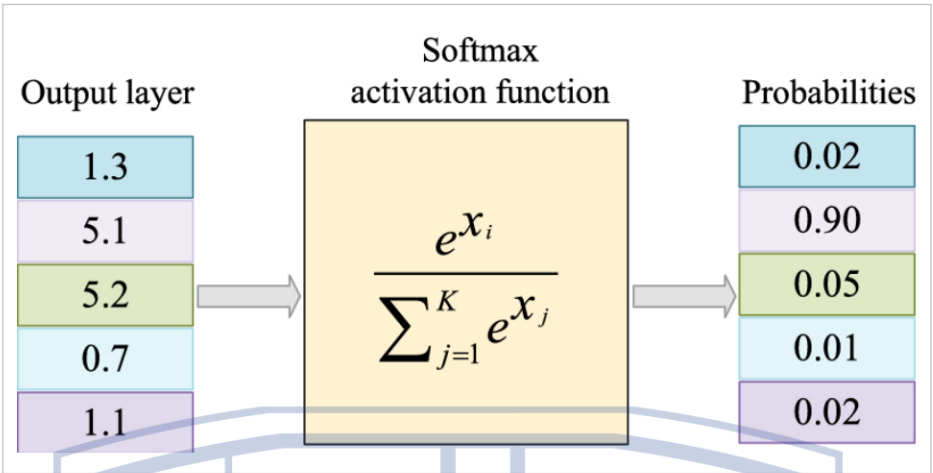
Dimana:

λ = koefisien regularisasi yang mengontrol kekuatan penalti

w_i = nilai masing-masing bobot jaringan

L2 regularisasi mendorong bobot tetap kecil dan terdistribusi merata sehingga model tidak bergantung berlebihan pada fitur tertentu, namun penalti ini hanya aktif saat pelatihan dan tidak memengaruhi nilai *output* saat inferensi.

c. *Softmax Activation*: Sebagai lapisan terakhir dalam *classification head*, vektor fitur yang telah melewati *Dropout* diteruskan ke lapisan *Dense* (10) yang menghasilkan nilai *logit* untuk setiap kelas melalui transformasi linier sebagai berikut [27]:



Gambar 2.6 *Softmax Schematic* [27]

$$Z\_out = W_2 \cdot h\_drop + b_2 \quad (10)$$

Dimana:

W_2 = matriks bobot yang dapat dilatih.

h_drop = vektor fitur *input* setelah dropout

b_2 = vektor bias

z_out = merepresentasikan skor mentah (*logit*) untuk masing-masing kelas

Nilai *logit* tersebut disebut sebagai *output layer* kemudian dikonversi menjadi distribusi probabilitas menggunakan fungsi *Softmax* sebagai berikut [27]:

$$Softmax(\hat{y}_k) = \frac{exp^{\{Z_k\}}}{\Sigma_{\{k=1\}}^{\{M\}} exp^{\{Z_k\}}} \quad (11)$$

Dimana:

Z_k = nilai *logit* untuk kelas ke-k

M = jumlah total kelas

$\hat{y}_k$ = probabilitas prediksi untuk kelas ke-k.

Softmax mengubah skor mentah menjadi nilai probabilitas dalam rentang [0, 1] di mana total seluruh probabilitas berjumlah 1. Kelas dengan nilai probabilitas tertinggi ditetapkan sebagai hasil klasifikasi akhir model.

Hasil probabilitas prediksi $\hat{y}^k$ dari *Softmax* selanjutnya digunakan untuk menghitung nilai *loss* yang mengukur kesenjangan antara prediksi model dan label *ground truth*. Pada tugas klasifikasi multi-kelas, label kelas dikonversi menjadi vektor *one-hot encoding* untuk menyesuaikan dengan fungsi loss *categorical cross-entropy*. Setiap kelas direpresentasikan sebagai vektor biner dengan satu nilai 1 pada kelas yang benar dan 0 pada kelas lainnya, didefinisikan sebagai berikut [40], [41]:

$$L_{CE} = -\sum_{k} y_k . log(\hat{y}^k) \quad (12)$$

Dimana:

y_k = label *ground truth* kelas ke-k, diperoleh dari *one-hot encoding* (bernilai 1 untuk kelas yang benar, 0 untuk kelas lainnya)

$\hat{y}^k$ = probabilitas prediksi model untuk kelas ke-k yang dihasilkan oleh *Softmax* pada Persamaan (11)

k = indeks kelas, k = 0, 1, …,

Semakin kecil nilai L_CE, semakin dekat distribusi prediksi model dengan distribusi label yang sebenarnya. Untuk meningkatkan kemampuan generalisasi model, fungsi *loss* total menggabungkan *Categorical Cross-Entropy* dengan regularisasi L2 dari Persamaan (9).

Desain ini efektif dalam memaksimalkan akurasi sekaligus meminimalkan jumlah parameter dan operasi *floating-point* (FLOPs) [42]. Untuk memperbarui bobot jaringan, digunakan *optimizer Adam*, yang bekerja dengan pendekatan adaptif berbasis gradien. Selama proses pelatihan, data diproses dalam bentuk *mini-batch* dan dilakukan secara iteratif dalam beberapa *epoch* hingga model mencapai konvergensi. Seluruh proses ini melibatkan mekanisme *forward propagation* untuk menghasilkan prediksi dan *backpropagation* untuk memperbarui parameter model. Pendekatan *Transfer learning* dengan EfficientNetB0 memanfaatkan bobot *pretrained* yang diperoleh dari pelatihan *dataset ImageNet* untuk meningkatkan *feature extraction* dengan membekukan (*freeze*) lapisan dasar (*base layers*) untuk mempertahankan representasi fitur umum yang telah dipelajari dan mengurangi waktu pelatihan pada tugas target [43].

Dengan memanfaatkan *transfer learning* memungkinkan model bekerja efektif pada *dataset* domain-spesifik yang lebih kecil, sambil menjaga efisiensi komputasi yang menjadi keunggulan EfficientNetB0. Dalam penelitian ini, EfficientNetB0 digunakan dengan konfigurasi *include_top=False* untuk menggunakan *convolutional base* sebagai *feature extractor*, yang kemudian dilengkapi dengan *custom classification head* yang disesuaikan untuk klasifikasi 10 kelas sampah organik dan anorganik [44]. Karakteristik arsitekturnya yang ringan namun *powerful* menjadikan model ini lebih praktis untuk diimplementasikan dalam sistem aplikasi berbasis *web* yang memerlukan *response time* yang cepat, konsumsi *resource* yang efisien, dan dapat diakses dalam pengelolaan sampah berkelanjutan.

2.3 *Image Preprocessing*

Pra-pemrosesan citra merupakan tahapan penting untuk memastikan data sesuai dengan kebutuhan *input* model EfficientNetB0 serta meningkatkan stabilitas dan kinerja pelatihan. Tahapan yang dilakukan meliputi:

1. *Resize Citra*

   Arsitektur EfficientNetB0 mensyaratkan seluruh citra masukan diubah ukurannya (*resize*) ke dimensi 224×224 piksel sebelum diproses oleh jaringan, sehingga setiap citra memiliki bentuk *tensor* yang konsisten sebesar 224×224×3 yang merepresentasikan tinggi, lebar, dan tiga kanal warna RGB. Keseragaman dimensi ini merupakan syarat mutlak agar proses ekstraksi fitur pada lapisan konvolusional dapat berjalan dengan benar menggunakan bobot *pretrained* yang telah dikalibrasi pada resolusi tersebut [12].

2. Normalisasi Piksel

   Sebelum citra dimasukkan ke dalam *backbone* EfficientNetB0, setiap nilai piksel mentah dalam rentang [0, 255] dinormalisasi terlebih dahulu ke rentang [−1, 1]. Ukuran ini merupakan ukuran *input* standar yang direkomendasikan untuk EfficientNetB0 menggunakan transformasi skala linier sebagai berikut [45]:

$$X\_norm = \frac{x}{127.5} - 1 \tag{13}$$

   Dimana:

   X = Nilai piksel asli.

   X_norm = Nilai piksel setelah normalisasi.

   Transformasi ini memastikan nilai batas bawah 0 menjadi −1,00 dan batas atas 255 menjadi +1,00. Normalisasi ini merupakan langkah *preprocessing* standar yang direkomendasikan untuk arsitektur EfficientNet, karena terbukti menstabilkan magnitudo gradien selama proses pelatihan dan mempercepat konvergensi model.

3. Augmentasi Data.

   Augmentasi data merupakan strategi yang efektif untuk meningkatkan generalisasi model *deep learning* dengan memperbesar keragaman data pelatihan melalui transformasi citra tanpa mengubah label (label-*preserving*). Teknik ini meliputi transformasi geometris seperti rotasi, translasi, *flipping*, dan perubahan skala, serta transformasi fotometrik seperti variasi kecerahan, yang bertujuan mensimulasikan kondisi nyata sehingga model tidak bergantung pada posisi, orientasi, ukuran, maupun pencahayaan objek tertentu. Dengan demikian, augmentasi mampu mengurangi *overfitting* yang terjadi akibat model menghafal data pelatihan. Secara teknis,

augmentasi diterapkan secara dinamis selama proses pelatihan, di mana setiap citra dimodifikasi secara acak pada setiap iterasi, sehingga model terus menerima variasi data yang berbeda. Dalam penerapannya, transformasi yang digunakan harus tetap mempertahankan identitas label dan disesuaikan dengan karakteristik *dataset* [46].

4. *Shuffling* dan Pembentukan *Batch*

   Mini-batch merupakan pendekatan optimasi dalam pelatihan model *deep learning* di mana pada setiap langkah optimasi, hanya sebagian kecil data yang dipilih secara acak (*random subsample*) dari keseluruhan data pelatihan untuk menginformasikan proses pembaruan parameter. Pendekatan ini terbukti mengurangi waktu komputasi secara drastis sekaligus membantu menghindari konvergensi ke titik pelana (*saddle points*) selama optimasi berlangsung. Pengacakan data (*shuffling*) sebelum pembentukan setiap *batch* menjadi bagian yang tidak terpisahkan dari mekanisme ini, karena memastikan subsampel yang terbentuk bersifat acak dan representatif sehingga estimasi gradien tidak bias terhadap urutan data tertentu [47].

2.4 Klasifikasi Sampah Organik dan Anorganik

Sampah merupakan sisa aktivitas manusia yang tidak digunakan lagi dan berpotensi menimbulkan dampak negatif apabila tidak dikelola dengan baik. Dalam konteks pengelolaan, sampah diklasifikasikan menjadi dua jenis utama: organik dan anorganik. Klasifikasi ini berperan penting sebagai dasar penentuan metode penanganan yang sesuai untuk setiap jenis sampah. Dengan melakukan klasifikasi sejak dari sumber, proses pengolahan dan pemanfaatan kembali bahan organik maupun anorganik dapat dilakukan secara lebih tepat dan efisien, sehingga mendukung upaya pengelolaan sampah yang berkelanjutan serta meminimalkan dampak terhadap lingkungan dan kesehatan manusia.

Sampah organik berasal dari sisa makhluk hidup seperti sisa makanan, buah, daun, dan bahan alami lainnya yang mudah terurai secara biologis. Sampah jenis ini dapat dimanfaatkan kembali melalui proses pengomposan atau pengolahan biologis lainnya seperti biogas. Proses dekomposisi sampah organik berlangsung relatif cepat dengan bantuan mikroorganisme pengurai [48], umumnya proses penguraian tersebut memerlukan waktu beberapa minggu hingga beberapa bulan bergantung pada kondisi lingkungan.

Sebaliknya, sampah anorganik merupakan sampah yang berasal dari bahan non-alami atau hasil olahan manusia seperti plastik, logam, kaca, kertas, dan kardus [49]. Sampah anorganik membutuhkan waktu sangat lama untuk terurai secara alami, bahkan ada yang tidak dapat terurai. Plastik misalnya, memerlukan ratusan hingga ribuan tahun untuk terurai

sampah plastik terdiri dari rantai karbon panjang yang sulit terurai secara alami [50], sedangkan logam dan kaca hampir tidak dapat terurai secara alami karena dapat digunakan kembali berkali-kali tanpa mengalami degradasi [51].

Perbedaan karakteristik ini menyebabkan sampah organik dan anorganik harus diklasifikasikan sejak awal dari sumbernya agar proses daur ulang dapat berjalan optimal, karena pemilahan di sumber merupakan strategi penting dalam optimalisasi aliran material dan pemanfaatan sumber daya secara berkelanjutan [52]. Dalam pengelolaan sampah, kontaminasi silang terjadi akibat ketidaktepatan pemilahan jenis sampah, yaitu tercampurnya sampah organik dan anorganik. Pencampuran ini menyebabkan perubahan karakteristik masing-masing jenis sampah, sehingga sampah organik tidak dapat diolah secara optimal dan sampah anorganik sulit didaur ulang. Akibatnya, jumlah sampah yang tidak tertangani meningkat. Sebaliknya, klasifikasi yang tepat memungkinkan sampah organik diolah secara biologis dan sampah anorganik didaur ulang dengan kualitas yang lebih baik.

2.5 *Confusion Matrix*

Confusion Matrix adalah metode evaluasi untuk mengukur kinerja model klasifikasi dengan membandingkan label aktual dan hasil prediksi dalam bentuk tabel. Baris menunjukkan kelas sebenarnya (*Actual*) dan kolom menunjukkan hasil prediksi (*Predicted*). Dari tabel ini dapat diketahui jumlah prediksi benar dan salah pada setiap kelas, sehingga pola kesalahan model dapat dianalisis dengan jelas [12].

Tabel 2.1 *Accuracy Metrics* [8]

		Actual	
		Positive	***Negative***
Predicted	*Positive*	*True Positive* (TP)	*False Positive* (FP)
	Negative	*False Negative* (FN)	*True Negative* (TN)

Dimana:

TP = Data positif yang diprediksi benar.

TN = Data negatif yang diprediksi benar.

FP = Data negatif yang salah diprediksi positif.

FN = Data positif yang salah diprediksi negatif.

Berdasarkan nilai tersebut, performa model diukur menggunakan beberapa metrik sebagai berikut [4], [12]:

1. *Accuracy* digunakan sebagai metrik dasar untuk menggambarkan kinerja umum model, namun tidak dijadikan indikator utama karena dapat memberikan nilai yang tampak tinggi pada data tidak seimbang sehingga berpotensi menimbulkan penilaian yang terlalu optimistis. Nilai *accuracy* dihitung menggunakan rumus sebagai berikut:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{14}$$

2. *Precision* mengukur jumlah positif yang diklasifikasikan dengan benar dibagi dengan jumlah contoh yang diberi label positif oleh sistem, dan *recall* mengukur jumlah contoh positif yang diklasifikasikan dengan benar dibagi dengan jumlah contoh positif dalam data. Mencari nilai *precision* dan *recall* digunakan rumus:

$$Precision = \frac{\text{TP}}{\text{TP} + \text{FP}}, \qquad Recall = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{15}$$

3. *F1-Score* merupakan kombinasi dari *precision* dan *recall* yang digunakan untuk menyeimbangkan kedua metrik, terutama pada kondisi *dataset imbalanced*. Mencari nilai *F1-Score* digunakan rumus [38]:

$$\text{F1} - Score = 2 \times \frac{Precision \cdot Recall}{Precision + Recall} \tag{16}$$

4. *Macro Average* merupakan metrik evaluasi yang menghitung rata-rata sederhana dari nilai metrik seluruh kelas secara setara tanpa mempertimbangkan jumlah sampel per kelas. *Macro Average* berlaku untuk *Precision*, *Recall*, maupun *F1-Score* dengan cara perhitungan yang identik. Rumus *Macro Average* didefinisikan sebagai berikut [53], [54]:

$$Macro - avg = \frac{1}{n}\sum_{i=1}^{n} M_{\text{i}} \tag{17}$$

Dimana:

n = jumlah total kelas

M_{i} = nilai metrik evaluasi (*Precision*, *Recall*, atau *F1-Score*) untuk kelas ke (i)

Pendekatan ini memastikan setiap kelas berkontribusi secara setara terhadap nilai akhir evaluasi, sehingga cocok digunakan pada *dataset* dengan jumlah sampel yang seimbang antar kelas.

2.6 Sistem Berbasis *Web*

Sistem berbasis *web* adalah aplikasi yang diakses melalui peramban (*browser*) tanpa memerlukan instalasi tambahan di sisi pengguna. Sistem ini menggunakan arsitektur *client-server*: pengguna berinteraksi melalui antarmuka *web* (*frontend*), sedangkan pemrosesan logika dan data dilakukan di sisi *server* (*backend*). Komponen utama dalam pengembangan sistem berbasis *web* pada penelitian ini meliputi *frontend, backend*, dan basis data.

1. *Frontend*

   Frontend adalah bagian sistem yang ditampilkan kepada pengguna melalui *browser*. Model *machine learning* dan *deep learning* pada umumnya dikembangkan menggunakan *Python* dan di-*deploy* sebagai aplikasi *web* yang dilayani dari *server backend*, sehingga dapat diakses langsung melalui *browser* tanpa instalasi di perangkat pengguna [55]. Pada penelitian ini, *frontend* dibangun menggunakan HTML (*HyperText Markup Language*) sebagai struktur halaman, CSS (*Cascading Style Sheets*) sebagai pengatur tampilan, dan JavaScript untuk mengelola interaksi pengguna serta komunikasi asinkron dengan backend melalui Fetch API. Sistem memiliki tiga halaman utama: Beranda untuk klasifikasi, Riwayat, dan Statistik.

2. *Backend*

   Backend berfungsi sebagai pengelola logika sistem, pemroses permintaan, dan penghubung antara *frontend* dengan model klasifikasi. *Flask* adalah *web framework* berbasis *Python* yang memungkinkan pengembang mempercepat dan memperluas pembangunan aplikasi *web* [56]. Pada penelitian ini, *Flask* menyediakan REST API (*Representational State Transfer*) dengan tujuh *endpoint* utama yang menangani klasifikasi gambar, penyimpanan riwayat, pengambilan statistik, dan pemeriksaan status model. *Flask-CORS* juga diterapkan untuk memungkinkan komunikasi antara *frontend* dan *backend* yang berjalan pada domain berbeda.

3. Basis Data

   Basis data digunakan untuk menyimpan seluruh hasil klasifikasi yang dilakukan pengguna. *SQLite* merupakan *database engine* yang paling banyak digunakan di dunia; *SQLite* ditemukan di hampir setiap *smartphone*, komputer, *web browser*, televisi, dan kendaraan, dengan faktor utama yang mendorong popularitasnya meliputi desain in-*process, codebase* yang mandiri, dan format berkas lintas platform [57]. *SQLite* dipilih pada penelitian ini karena tidak memerlukan *server* terpisah dan seluruh data tersimpan dalam satu berkas (*history.db*), menyimpan informasi berupa nama gambar, jenis sampah, kategori, akurasi, dan waktu.