BAB 2
LANDASAN TEORI

2.1 Penelitian Terdahulu

Penelitian ini didasarkan pada beberapa studi terdahulu yang membahas deteksi citra wajah *AI-generated*. Seiring berkembangnya model generatif seperti *diffusion model*, berbagai pendekatan telah dikembangkan untuk membedakan citra wajah asli dan citra wajah sintetis. Dalam konteks tersebut, arsitektur EfficientNet telah digunakan dalam sejumlah penelitian deteksi citra sintetis karena efisiensi parameter dan performa klasifikasinya yang kompetitif. Studi-studi tersebut menjadi landasan dalam memahami karakteristik citra sintetis, pendekatan pelatihan model, serta strategi evaluasi yang relevan dengan penelitian ini. Penelitian terdahulu yang menjadi rujukan utama dirangkum pada Tabel 2.1.

Tabel 2.1. Ringkasan Penelitian Terdahulu

Penulis	**Judul**	**Metode**	**Hasil**
GJA Porcile, J Gindi, S Mundra, JR Verbus, H Farid	Finding AI-Generated Faces in the Wild	Menggunakan EfficientNet-B1 *pretrained* ImageNet dengan data latih 60.000 citra dan pengujian pada dataset yang lebih luas.	TPR 98% pada pengujian *in-engine*, 84,5% pada pengujian *out-of-engine*, dan FPR hanya 0,5%. Model juga tetap *robust* hingga resolusi 128×128.
Lanjut di halaman berikutnya			

Tabel 2.1 Ringkasan Penelitian Terdahulu (lanjutan)

Penulis	**Judul**	**Metode**	**Hasil**
M Tan, QE Le	Rethinking Model Scaling for Convolutional Neural Networks	Menggunakan EfficientNet berbasis CNN dengan pendekatan *compound scaling* dan pembagian standar dataset ImageNet, sekitar 1,28 juta data *training* dan 50 ribu data validasi.	Varian awal hingga menengah EfficientNet menunjukkan peningkatan akurasi yang konsisten dari 77,1% pada B0 menjadi 81,6% pada B3 dengan jumlah parameter yang tetap efisien. EfficientNet-B1 juga melampaui akurasi ResNet-152 dengan parameter yang lebih sedikit.
B Babu Vimala, T Baiju, E Srinivasan, S Saravanan, U Mathivanan, *et al.*	Detection and Classification of Brain Tumor Using Hybrid Deep Learning Models	Menggunakan lima varian EfficientNet *pretrained* ImageNet, yaitu EfficientNet-B0 hingga EfficientNet-B4, dalam skenario *transfer learning* dan *fine-tuning* untuk klasifikasi multikelas citra MRI otak.	Varian EfficientNet-B0 hingga EfficientNet-B3 menunjukkan performa kompetitif, dengan EfficientNet-B2 sebagai varian yang paling menonjol. Hasil tersebut menunjukkan bahwa rentang varian awal hingga menengah layak digunakan untuk analisis komparatif.

Penelitian berjudul *Finding AI-Generated Faces in the Wild* mengusulkan metode deteksi wajah AI-generated dalam kondisi dunia nyata (in-the-wild) dengan

memanfaatkan arsitektur Convolutional Neural Network berbasis EfficientNet-B1 [11]. Dalam studi tersebut, dataset yang digunakan berukuran besar yang mencakup lebih dari 225.000 citra, termasuk 120.000 wajah asli dan 105.900 wajah hasil generasi AI dari berbagai model seperti StyleGAN, DALL-E, dan Stable Diffusion [11]. Model dilatih menggunakan 60.000 data (30.000 wajah asli dan 30.000 wajah AI-generated) dan diuji pada data yang lebih luas untuk mengevaluasi kemampuan generalisasi [11]. Berdasarkan hasil pengujian, model tersebut memperoleh tingkat deteksi sebesar 98,0% pada data yang serupa dengan data pelatihan dengan tingkat kesalahan (false positive rate) sebesar 0,5%, namun performa menurun menjadi 84,5% ketika diuji pada data dari model generatif yang tidak digunakan saat pelatihan [11]. Selain itu, model menunjukkan ketahanan terhadap variasi kualitas citra, dengan akurasi sebesar 91,9% pada resolusi 128×128 dan 88,0% pada kualitas JPEG rendah [11]. Namun, fokus utama studi tersebut berada pada perancangan metode deteksi yang tahan terhadap variasi data, sehingga belum membahas secara khusus pengaruh konfigurasi pembagian data *training* dan *testing* terhadap performa model [11].

Penelitian berjudul *Rethinking Model Scaling for Convolutional Neural Networks* memperkenalkan EfficientNet sebagai arsitektur *Convolutional Neural Network* (CNN) yang dikembangkan dengan pendekatan *compound scaling*, yaitu strategi penskalaan model yang mempertimbangkan dimensi kedalaman (*depth*), lebar (*width*), dan resolusi (*resolution*) secara bersamaan [1]. Berbeda dengan pendekatan konvensional yang hanya meningkatkan satu dimensi, metode ini terbukti lebih optimal dalam meningkatkan performa model [1]. Pada rentang varian awal hingga menengah, EfficientNet telah menunjukkan peningkatan performa yang konsisten, yaitu mulai dari EfficientNet-B0 dengan akurasi 77,1%, EfficientNet-B1 sebesar 79,1%, EfficientNet-B2 sebesar 80,1%, hingga EfficientNet-B3 sebesar 81,6% pada dataset ImageNet [1]. Peningkatan tersebut dicapai dengan jumlah parameter yang tetap efisien, yaitu sekitar 5,3 juta pada B0, 7,8 juta pada B1, 9,2 juta pada B2, dan 12 juta pada B3 [1]. Bahkan, EfficientNet-B1 telah mampu melampaui akurasi ResNet-152 yang memiliki sekitar 60 juta parameter, sehingga menunjukkan bahwa kekuatan utama EfficientNet sudah terlihat jelas pada varian awal-menengah [1]. Temuan tersebut menunjukkan bahwa pendekatan *compound scaling* mampu mendukung peningkatan akurasi sekaligus menjaga penggunaan komputasi tetap efisien. Dengan karakteristik tersebut, EfficientNet menjadi salah satu arsitektur yang relevan digunakan pada berbagai tugas pengolahan citra, termasuk deteksi wajah berbasis *AI-generated*.

Penelitian berjudul *Detection and Classification of Brain Tumor Using Hybrid Deep Learning Models* mengevaluasi lima varian EfficientNet pretrained ImageNet, yaitu EfficientNet-B0 hingga EfficientNet-B4, untuk klasifikasi multi-kelas citra MRI otak berbasis *transfer learning* dan *fine-tuning* [14]. Studi tersebut memanfaatkan dataset CE-MRI Figshare yang mencakup 3064 citra MRI dari 233 pasien, lalu membandingkan performa beberapa varian EfficientNet dalam berbagai skenario eksperimen [14]. Pada pengujian utama setelah augmentasi data, EfficientNet-B0 mencapai *accuracy* 97,43%, EfficientNet-B1 sebesar 98,66%, EfficientNet-B2 sebesar 99,06%, EfficientNet-B3 sebesar 98,08%, dan varian dengan kompleksitas tertinggi pada eksperimen tersebut mencapai 97,90% [14]. Hasil ini menunjukkan bahwa peningkatan performa pada keluarga EfficientNet tidak selalu berlangsung secara linear terhadap peningkatan ukuran model, karena varian awal hingga menengah justru memberikan hasil yang lebih kompetitif dibandingkan varian yang lebih kompleks [14]. Selain itu, pada validasi lintas dataset eksternal, EfficientNet-B2 masih mencapai *accuracy* 92,23%, *precision* 92,11%, *recall* 92,11%, *specificity* 95,96%, dan *F1-score* 92,02% [14]. Penelitian tersebut juga menegaskan bahwa pemilihan varian EfficientNet perlu mempertimbangkan ukuran dataset, sumber daya komputasi, kedalaman model, jumlah parameter, dan *batch size* [14]. Meskipun penelitian ini berada pada domain klasifikasi citra medis, bukan deteksi wajah *AI-generated*, studi tersebut tetap relevan karena menunjukkan bahwa varian awal hingga menengah EfficientNet layak digunakan sebagai objek analisis komparatif dalam penelitian berbasis klasifikasi citra.

2.2 Wajah AI-Generated

Perkembangan kecerdasan buatan (*Artificial Intelligence*) telah mendorong kemajuan model generatif yang mampu menghasilkan citra sintetis dengan tampilan menyerupai citra hasil pengambilan kamera [6, 15, 16]. Model generatif mempelajari pola dan distribusi data pelatihan, kemudian memanfaatkan pengetahuan tersebut untuk membentuk data baru yang memiliki karakteristik visual serupa [15].

Wajah *AI-generated* merupakan citra wajah manusia yang dibentuk melalui proses generatif berbasis kecerdasan buatan tanpa pengambilan gambar secara langsung menggunakan kamera. Citra tersebut tidak selalu merepresentasikan individu yang benar-benar ada, tetapi merupakan hasil sintesis pola visual, seperti

struktur wajah, tekstur kulit, rambut, ekspresi, dan pencahayaan, yang dipelajari oleh model generatif [11, 16].

Wajah *AI-generated* berbeda dari *deepfake*. Citra *deepfake* umumnya dibentuk dengan memanipulasi atau mengganti identitas wajah seseorang pada citra maupun video. Sebaliknya, wajah *AI-generated* dapat membentuk identitas baru yang tidak berasal dari individu tertentu [17, 18]. Pembentukan wajah sintetis juga dapat dilakukan melalui masukan berupa *prompt* teks yang mendeskripsikan karakteristik subjek, seperti usia, ekspresi, ciri fisik, latar, dan kondisi pencahayaan.

Citra wajah sintetis dapat dihasilkan melalui beberapa pendekatan, antara lain *Generative Adversarial Network* (GAN), *diffusion model*, dan model autoregresif [2, 15, 19]. GAN menggunakan proses pembelajaran antara generator dan diskriminator, sedangkan *diffusion model* membentuk citra melalui proses penghilangan *noise* secara bertahap [15, 19]. Sementara itu, pendekatan autoregresif membentuk keluaran secara bertahap berdasarkan konteks masukan dan representasi yang telah dihasilkan sebelumnya [2].

Peningkatan kemampuan model generatif membuat citra wajah sintetis memiliki struktur, pencahayaan, dan tekstur yang semakin realistis. Kondisi tersebut menyebabkan perbedaan visual antara wajah asli dan wajah hasil generasi AI semakin sulit dikenali melalui pengamatan manusia. Meskipun demikian, citra sintetis masih dapat mengandung pola atau karakteristik distribusi tertentu yang dapat dipelajari menggunakan CNN [5, 7, 11]. Oleh karena itu, pendekatan komputasional diperlukan untuk membantu mendeteksi citra wajah hasil generasi AI.

2.2.1 Karakteristik Wajah *AI-generated* pada Domain Frekuensi

Pada citra diam, domain spasial merepresentasikan nilai intensitas piksel berdasarkan posisinya pada citra, sedangkan domain frekuensi merepresentasikan perubahan intensitas tersebut sebagai komponen-komponen frekuensi. Melalui transformasi Fourier, komponen frekuensi rendah umumnya berkaitan dengan perubahan intensitas yang berlangsung secara perlahan, seperti bentuk global dan pencahayaan. Sebaliknya, komponen frekuensi tinggi berkaitan dengan perubahan yang lebih cepat, seperti tepi, tekstur halus, dan derau. Transformasi tersebut tidak menghasilkan informasi baru, tetapi menyusun kembali informasi citra sehingga pola periodik atau ketidakteraturan spektral yang sulit diamati secara langsung pada domain spasial dapat lebih mudah dianalisis [20].

Perbedaan antara wajah asli dan wajah hasil generasi AI dapat teramati pada domain frekuensi, meskipun karakteristiknya bergantung pada model generatif yang digunakan. Convertini *et al.* menerapkan *two-dimensional Discrete Fourier Transform* (2D-DFT) untuk membandingkan wajah asli dari dataset FFHQ dan CelebA dengan wajah sintetis yang dihasilkan menggunakan StyleGAN dan StyleGAN2 [20]. Hasil penelitian tersebut menunjukkan adanya perbedaan karakteristik spektral, terutama pada komponen frekuensi menengah dan tinggi. Corvi *et al.* juga menemukan bahwa citra yang dihasilkan oleh beberapa keluarga model, termasuk GAN, *diffusion model*, dan VQ-GAN, dapat menunjukkan artefak pada domain Fourier serta perbedaan kandungan sinyal frekuensi menengah hingga tinggi dibandingkan citra asli [21]. Temuan tersebut menunjukkan bahwa domain frekuensi dapat menyediakan fitur pembeda tambahan ketika perbedaan visual pada domain spasial semakin sulit dikenali.

Meskipun demikian, karakteristik frekuensi tidak dapat dianggap sebagai tanda mutlak yang selalu terdapat pada seluruh citra hasil generasi AI. Chandrasegaran *et al.* menunjukkan bahwa ketidaksesuaian spektrum frekuensi tinggi pada citra sintetis dapat dikurangi melalui perubahan pada operasi *upsampling* generator [22]. Penelitian Karageorgiou *et al.* juga menunjukkan bahwa artefak spektral dapat berbeda antargenerator sehingga memengaruhi kemampuan generalisasi detektor berbasis frekuensi [23]. Oleh karena itu, informasi frekuensi diposisikan sebagai petunjuk forensik pelengkap dan bukan sebagai bukti tunggal bahwa suatu citra merupakan hasil generasi AI. Penelitian ini menggunakan citra RGB pada domain spasial sebagai masukan EfficientNet dan tidak menerapkan transformasi Fourier secara eksplisit. Dengan demikian, kajian domain frekuensi digunakan sebagai landasan teoritis mengenai alternatif karakteristik pembeda antara citra asli dan sintetis, bukan sebagai metode yang diterapkan dalam eksperimen.

2.2.2 4o Image Generation

4o Image Generation diperkenalkan oleh OpenAI pada 25 Maret 2025 sebagai kemampuan pembangkitan citra yang terintegrasi secara langsung dalam model multimodal GPT-4o [2]. Sistem tersebut dapat menerima masukan berupa teks maupun citra untuk menghasilkan atau memodifikasi gambar. Kemampuannya mencakup pembentukan citra fotorealistis, transformasi citra, penerapan instruksi terperinci, dan penyisipan teks ke dalam gambar [2].

Berbeda dengan DALL-E yang menggunakan pendekatan *diffusion*, 4o Image Generation dijelaskan sebagai model autoregresif yang terintegrasi dalam arsitektur GPT-4o [2]. Integrasi tersebut memungkinkan proses pembangkitan citra memanfaatkan pemahaman bahasa, konteks percakapan, dan instruksi visual dalam satu sistem multimodal. Namun, rincian teknis mengenai susunan arsitektur serta mekanisme pembentukan representasi visualnya belum dipublikasikan secara lengkap [3].

Kemampuan 4o Image Generation telah dianalisis melalui sejumlah penelitian empiris. Chen *et al.* mengevaluasi GPT-4o pada lebih dari 20 tugas yang dikelompokkan ke dalam empat kategori, yaitu *text-to-image*, *image-to-image*, *image-to-3D*, dan *image-to-X* [3]. Luasnya cakupan pengujian tersebut menunjukkan bahwa GPT-4o dapat digunakan untuk berbagai bentuk pembangkitan dan transformasi visual.

Pengujian kuantitatif oleh Yan *et al.* menunjukkan bahwa GPT-4o memperoleh skor keseluruhan sebesar 0,84 pada tolok ukur GenEval, yang menjadi skor tertinggi di antara model pembangkitan citra yang dibandingkan dalam penelitian tersebut [4]. Pada pengujian penyuntingan citra menggunakan Reason-Edit, GPT-4o memperoleh skor 0,929, sedangkan metode SmartEdit memperoleh skor 0,572. Selisih sebesar 0,357 tersebut memperlihatkan keunggulan GPT-4o dalam mengikuti instruksi dan melakukan penyuntingan visual pada tolok ukur yang digunakan.

Evaluasi OpenAI juga menunjukkan perkembangan dalam keragaman representasi manusia dibandingkan DALL-E 3. Pada evaluasi kelompok berdasarkan ras, frekuensi keluaran heterogen meningkat dari 80% pada DALL-E 3 menjadi 100% pada 4o Image Generation. Pada evaluasi warna kulit untuk citra individu, nilainya meningkat dari 61% menjadi 96% [2]. Selain itu, skor kesesuaian atribut pada evaluasi historis dan realistis meningkat dari 92% menjadi 97%. Hasil tersebut menunjukkan peningkatan keragaman dan kesesuaian representasi manusia, meskipun angka tersebut tidak secara langsung mengukur tingkat fotorealisme citra.

Seiring dengan berkembangnya 4o Image Generation, kemampuan pembangkitan citra mengalami peningkatan dalam aspek fotorealisme, kepatuhan terhadap instruksi, dan pengendalian komposisi visual [2, 4]. Kemampuan tersebut memungkinkan pembentukan wajah sintetis dengan detail tekstur, struktur wajah, ekspresi, serta pencahayaan yang semakin menyerupai citra hasil fotografi. Tingginya kualitas visual tersebut menyebabkan pembedaan antara wajah asli

dan wajah hasil generasi AI semakin sulit dilakukan hanya melalui pengamatan manusia. Meskipun demikian, citra sintetis masih dapat mengandung karakteristik distribusi dan pola visual tertentu yang dapat dipelajari melalui pendekatan berbasis CNN [5, 7, 11]. Oleh karena itu, analisis komputasional diperlukan untuk mengenali perbedaan halus antara citra wajah asli dan citra wajah hasil generasi AI.

2.3 Convolutional Neural Network

Convolutional Neural Network (CNN) adalah salah satu arsitektur *deep learning* yang banyak digunakan untuk memproses data berstruktur grid, termasuk citra dua dimensi [6, 24]. CNN mengekstraksi fitur melalui operasi konvolusi yang dilakukan secara berlapis, sehingga mampu membentuk representasi fitur secara hierarkis [24, 25]. Dalam bentuk matematis, proses konvolusi dua dimensi dihitung menggunakan Persamaan 2.1.

$$S(i,j) = \sum_m \sum_n X(i+m, j+n) K(m,n) \tag{2.1}$$

Pada implementasi jaringan saraf modern, operasi yang disebut konvolusi umumnya dihitung sebagai *cross-correlation*, yaitu tanpa membalik posisi kernel. Meskipun demikian, operasi tersebut tetap disebut sebagai lapisan konvolusi dalam literatur *deep learning* [8]. Pada Persamaan 2.1, X menyatakan citra masukan, K menyatakan kernel atau filter konvolusi, dan $S(i,j)$ menyatakan nilai hasil konvolusi pada posisi (i,j). Persamaan tersebut menunjukkan bentuk dasar operasi konvolusi pada satu kanal masukan. Pada citra berwarna seperti RGB, operasi konvolusi dilakukan pada masing-masing kanal warna, kemudian hasil dari seluruh kanal dijumlahkan untuk menghasilkan satu nilai pada *feature map*. Dengan demikian, operasi konvolusi pada citra RGB tetap mengikuti prinsip yang sama, tetapi melibatkan kontribusi dari kanal merah, hijau, dan biru. Melalui operasi tersebut, CNN dapat mengekstraksi pola lokal pada citra, seperti bentuk, tepi, dan tekstur, yang kemudian digunakan untuk membentuk representasi fitur pada lapisan-lapisan berikutnya.

Secara umum, CNN terdiri atas *convolutional layer*, fungsi aktivasi, *pooling layer*, dan *fully connected layer* atau lapisan klasifikasi [8, 24]. CNN dapat bekerja dengan baik pada pengenalan citra karena memiliki tiga karakteristik utama, yaitu konektivitas lokal (*local connectivity*), penggunaan bobot bersama (*weight*

sharing), dan pembentukan representasi fitur secara hierarkis [6, 8].

Konektivitas lokal memungkinkan setiap kernel memproses wilayah kecil pada citra yang disebut *local receptive field*. Karakteristik ini sesuai dengan struktur citra karena pola visual, seperti tepi, sudut, tekstur, dan perubahan warna, terbentuk dari hubungan antarpiksel yang berdekatan. Kernel kemudian digeser ke seluruh posisi citra menggunakan bobot yang sama. Mekanisme *weight sharing* tersebut membuat pola yang sama dapat dikenali pada lokasi yang berbeda sekaligus mengurangi jumlah parameter dibandingkan jaringan yang menghubungkan setiap piksel secara langsung ke seluruh neuron. Operasi konvolusi juga memiliki sifat *translation equivariance*, yaitu pergeseran posisi pola pada citra akan menghasilkan pergeseran respons yang sesuai pada *feature map*. Selanjutnya, operasi *pooling* atau penggunaan *stride* membantu meningkatkan toleransi model terhadap perubahan kecil pada posisi objek [6, 8].

Ketika beberapa lapisan konvolusi disusun secara bertingkat, CNN membentuk representasi fitur secara hierarkis. Lapisan awal umumnya merespons pola sederhana, seperti tepi, warna, dan orientasi. Lapisan menengah menggabungkan pola tersebut menjadi tekstur dan bagian objek, sedangkan lapisan yang lebih dalam membentuk representasi yang lebih kompleks untuk mendukung keputusan klasifikasi [6, 25]. Seluruh kernel dipelajari secara otomatis melalui proses pelatihan, sehingga fitur yang terbentuk disesuaikan dengan pola yang paling relevan terhadap tugas klasifikasi dan tidak perlu ditentukan secara manual.

Dalam konteks deteksi citra wajah *AI-generated*, karakteristik tersebut memungkinkan CNN mempelajari pola pembeda pada tekstur, tepi, warna, dan hubungan spasial antara bagian-bagian wajah. Pola tersebut dapat bersifat halus dan tidak selalu mudah dikenali melalui pengamatan manusia, tetapi masih dapat menghasilkan respons tertentu pada *feature map* CNN [5, 7]. Meskipun demikian, kemampuan CNN juga dipengaruhi oleh rancangan arsitektur, kualitas data, dan kapasitas model. Peningkatan kapasitas CNN dapat dilakukan dengan menambah kedalaman, lebar, atau resolusi masukan, tetapi peningkatan hanya pada satu dimensi belum tentu menghasilkan keseimbangan antara akurasi dan kebutuhan komputasi. Permasalahan penskalaan tersebut menjadi dasar pengembangan arsitektur EfficientNet [1].

2.3.1 EfficientNet

EfficientNet merupakan salah satu keluarga arsitektur *Convolutional Neural Network* yang dikembangkan untuk memperoleh keseimbangan antara akurasi klasifikasi dan efisiensi komputasi [1]. Sebagai bagian dari CNN, EfficientNet tetap menggunakan lapisan konvolusi untuk mengekstraksi pola lokal dan membentuk representasi fitur secara hierarkis. Perbedaannya terletak pada penggunaan blok konvolusi yang efisien serta strategi untuk meningkatkan kapasitas jaringan secara lebih terstruktur.

Pada arsitektur CNN, kapasitas model dapat ditingkatkan melalui penambahan kedalaman (*depth*), lebar (*width*), atau resolusi masukan (*resolution*). Kedalaman berkaitan dengan jumlah lapisan atau pengulangan blok, lebar menunjukkan jumlah kanal pada *feature map*, sedangkan resolusi menentukan tingkat detail spasial citra yang diterima model. Penskalaan yang hanya berfokus pada salah satu dimensi dapat menghasilkan kapasitas jaringan yang kurang seimbang karena peningkatan pada satu bagian belum tentu didukung oleh bagian lainnya. Permasalahan tersebut menjadi dasar pengembangan *compound scaling* pada EfficientNet, yang mengatur peningkatan ketiga dimensi secara terkoordinasi [1].

Inovasi utama EfficientNet adalah pendekatan *compound scaling*, yaitu mekanisme penskalaan yang meningkatkan kedalaman, lebar, dan resolusi secara bersamaan dengan proporsi yang terkoordinasi. Melalui mekanisme tersebut, ketika varian model diperbesar, jumlah blok ditambah untuk memperdalam proses ekstraksi fitur, jumlah kanal ditingkatkan untuk memperluas keragaman fitur yang dapat dipelajari, dan resolusi masukan ditingkatkan agar model memperoleh detail spasial yang lebih tinggi. Peningkatan ketiga dimensi dilakukan secara seimbang agar penambahan kapasitas model tetap sejalan dengan kebutuhan komputasinya [1].

Pengembangan EfficientNet diawali dengan pencarian arsitektur dasar EfficientNet-B0 menggunakan *Neural Architecture Search* (NAS). Proses tersebut menghasilkan jaringan dasar yang mempertimbangkan keseimbangan antara akurasi dan kebutuhan komputasi. Setelah EfficientNet-B0 diperoleh, mekanisme *compound scaling* digunakan untuk membentuk varian yang lebih besar, mulai dari EfficientNet-B1 hingga EfficientNet-B7 [1, 26].

Secara arsitektural, EfficientNet terdiri atas beberapa bagian utama, yaitu *stem convolution*, rangkaian *Mobile Inverted Bottleneck Convolution* (MBConv),

dan *top convolution* [1]. *Stem convolution* berperan sebagai tahap awal untuk memproses citra masukan dan mengekstraksi pola visual dasar, seperti tepi, tekstur, dan bentuk sederhana [1, 8]. Fitur yang dihasilkan kemudian diproses melalui rangkaian MBConv sebagai komponen utama dalam EfficientNet [1]. Blok ini pertama kali diperkenalkan pada arsitektur MobileNetV2 [1, 27] dan memanfaatkan prinsip *inverted residual* serta *depthwise separable convolution* untuk meningkatkan efisiensi ekstraksi fitur dengan jumlah parameter yang lebih rendah dibandingkan konvolusi standar [27]. Pada EfficientNet, MBConv tidak digunakan sebagai satu blok tunggal, melainkan disusun secara berulang sesuai konfigurasi masing-masing varian model [1].

Di dalam MBConv, proses ekstraksi fitur dilakukan melalui beberapa tahapan, yaitu *expansion convolution*, *depthwise convolution*, *batch normalization*, aktivasi Swish, *squeeze-and-excitation*, *projection convolution*, dan *skip connection* [1, 27]. *Expansion convolution* digunakan untuk memperluas jumlah kanal fitur sebelum proses konvolusi utama dilakukan, sedangkan *depthwise convolution* memproses setiap kanal secara terpisah agar ekstraksi fitur lebih efisien [27]. *Batch normalization* berfungsi menjaga distribusi aktivasi agar tetap stabil selama pelatihan berlangsung, sehingga model dapat dilatih dengan lebih stabil [28]. Aktivasi Swish digunakan karena memiliki karakteristik non-linear yang efektif pada jaringan dalam dan menjadi salah satu komponen yang digunakan pada EfficientNet [1, 29].

Selain itu, EfficientNet mengintegrasikan mekanisme *squeeze-and-excitation* (SE) untuk meningkatkan sensitivitas model terhadap kanal fitur yang relevan [1, 30]. Melalui mekanisme ini, model dapat menekankan kanal fitur yang lebih informatif dan menurunkan kontribusi kanal yang kurang relevan [30]. Setelah itu, *projection convolution* digunakan untuk mengembalikan representasi fitur ke dimensi yang lebih ringkas, sedangkan *skip connection* menjaga informasi dari lapisan sebelumnya tetap diteruskan, sehingga proses pembelajaran menjadi lebih stabil [1, 27]. Rangkaian proses tersebut kemudian dilanjutkan ke *top convolution* untuk membentuk representasi fitur akhir sebelum masuk ke bagian klasifikasi [1].

Dalam berbagai tugas klasifikasi citra skala besar, EfficientNet menunjukkan rasio akurasi terhadap jumlah parameter yang tinggi serta efisiensi komputasi yang baik [1]. Kemampuannya dalam mengoptimalkan representasi fitur pada berbagai skala resolusi memungkinkan model untuk menangkap detail tekstur dan struktur citra secara lebih komprehensif. Dengan karakteristik tersebut, EfficientNet menjadi arsitektur CNN modern yang relevan untuk berbagai kebutuhan pengolahan

citra [1, 11, 12, 13].

A Varian EfficientNet-B0, EfficientNet-B1, EfficientNet-B2, dan EfficientNet-B3

Keluarga EfficientNet terdiri atas beberapa varian model yang dibentuk melalui pendekatan *compound scaling*, dimulai dari varian dasar EfficientNet-B0 hingga varian yang lebih besar seperti EfficientNet-B7 [1]. Setiap kenaikan varian menunjukkan peningkatan kapasitas representasi fitur melalui penskalaan yang terkoordinasi pada dimensi kedalaman, lebar, dan resolusi jaringan. Pada rentang varian awal hingga menengah, perubahan skala tersebut sudah dapat menunjukkan perbedaan karakteristik model, dari varian dengan beban komputasi paling rendah sampai varian dengan kapasitas representasi yang lebih besar.

Dalam penelitian asli EfficientNet, varian awal-menengah seperti EfficientNet-B1 dan EfficientNet-B3 dilaporkan mampu menghasilkan kinerja klasifikasi yang kompetitif dengan penggunaan parameter yang lebih efisien dibandingkan beberapa arsitektur CNN berukuran lebih besar [1]. Pada konteks yang lebih dekat dengan deteksi wajah sintetis, Porcile *et al.* [11] menunjukkan bahwa EfficientNet-B1 memiliki efektivitas tinggi dalam mendeteksi wajah *AI-generated*, sedangkan EfficientNet-B0 juga telah digunakan secara efektif pada tugas deteksi *deepfake* [12, 13]. Selain itu, EfficientNet-B3 juga sering muncul sebagai varian dengan performa tinggi pada studi komparatif, sementara varian B0, B1, dan B2 tetap berada pada rentang performa yang berdekatan [31].

Secara metodologis, rentang EfficientNet-B0 hingga EfficientNet-B3 dipilih untuk merepresentasikan peningkatan kapasitas arsitektur secara bertahap dengan kebutuhan komputasi yang masih dapat dikelola pada dataset berukuran sedang. Dalam literatur *deep learning*, model dengan kapasitas parameter yang semakin besar cenderung memiliki risiko *overfitting* yang lebih tinggi ketika dilatih pada jumlah sampel yang terbatas, karena model dapat lebih mudah menyesuaikan diri pada detail spesifik data latih dibandingkan mempelajari pola fitur yang lebih umum [8]. Dengan karakteristik tersebut, EfficientNet-B0, EfficientNet-B1, EfficientNet-B2, dan EfficientNet-B3 digunakan dalam penelitian ini untuk merepresentasikan peningkatan bertahap kompleksitas arsitektur dalam keluarga EfficientNet sekaligus memberikan rentang varian yang cukup representatif untuk menganalisis pengaruh skala model terhadap performa deteksi citra wajah *AI-generated* secara sistematis.

Perbedaan konfigurasi bawaan EfficientNet-B0 hingga EfficientNet-B3

ditunjukkan pada Tabel 2.2. Ukuran *feature map* menunjukkan dimensi representasi fitur terakhir yang dihasilkan oleh jaringan konvolusi pada konfigurasi bawaan setiap varian. Jumlah parameter menunjukkan banyaknya nilai bobot yang membentuk arsitektur model serta berkaitan dengan kapasitas model dan kebutuhan memori. Sementara itu, FLOPs (*floating-point operations*) menunjukkan perkiraan jumlah operasi komputasi yang diperlukan untuk memproses satu citra melalui satu kali proses propagasi maju pada resolusi masukan bawaan. Oleh karena itu, FLOPs dapat digunakan sebagai indikator kebutuhan komputasi suatu model, meskipun tidak secara langsung menunjukkan waktu pemrosesan karena waktu aktual juga dipengaruhi oleh perangkat keras dan implementasi yang digunakan.

Tabel 2.2. Perbandingan konfigurasi EfficientNet-B0 hingga EfficientNet-B3 [1]

Varian	Jumlah blok MBConv	Resolusi masukan bawaan	Ukuran *feature map* akhir	Parameter	FLOPs
EfficientNet-B0	16	224×224	$7 \times 7 \times 1280$	5,3 juta	0,39 miliar
EfficientNet-B1	23	240×240	$8 \times 8 \times 1280$	7,8 juta	0,70 miliar
EfficientNet-B2	23	260×260	$9 \times 9 \times 1408$	9,2 juta	1,0 miliar
EfficientNet-B3	26	300×300	$10 \times 10 \times 1536$	12 juta	1,8 miliar

Berdasarkan Tabel 2.2, peningkatan skala dari EfficientNet-B0 hingga EfficientNet-B3 diikuti oleh peningkatan jumlah parameter dan FLOPs. Jumlah parameter meningkat dari 5,3 juta pada EfficientNet-B0 menjadi 12 juta pada EfficientNet-B3, sedangkan kebutuhan komputasinya meningkat dari 0,39 miliar menjadi 1,8 miliar FLOPs. Peningkatan tersebut terjadi karena penskalaan kedalaman dan lebar menambah jumlah blok serta kanal jaringan, sementara peningkatan resolusi memperbesar ukuran data spasial yang harus diproses. Dengan demikian, peningkatan kapasitas representasi pada varian yang lebih besar disertai dengan konsekuensi berupa kebutuhan memori dan komputasi yang lebih tinggi [1].

EfficientNet-B0 merupakan arsitektur dasar yang diperoleh melalui proses *Neural Architecture Search*. Varian ini memiliki 16 blok MBConv dan merupakan model dengan jumlah parameter serta kebutuhan komputasi paling rendah di antara varian yang digunakan dalam penelitian ini. Pada konfigurasi bawaannya, EfficientNet-B0 menghasilkan *feature map* akhir berukuran $7 \times 7 \times 1280$. Karena sifatnya yang ringan, EfficientNet-B0 dapat digunakan sebagai model dasar untuk membandingkan pengaruh peningkatan skala pada varian berikutnya. Relevansi varian ini terhadap deteksi citra sintetis juga ditunjukkan oleh penggunaannya sebagai pengekstraksi fitur pada arsitektur gabungan EfficientNet dan *Vision Transformer* untuk deteksi *deepfake* [13]. Selain itu, EfficientNet-B0 telah

dievaluasi secara langsung bersama varian EfficientNet lainnya pada dataset deteksi *deepfake* [12].

EfficientNet-B1 merupakan hasil penskalaan dari EfficientNet-B0 dengan penambahan kedalaman jaringan dan peningkatan resolusi masukan bawaan. Varian ini memiliki 23 blok MBConv dan menghasilkan *feature map* akhir berukuran $8 \times 8 \times 1280$. Peningkatan tersebut menyebabkan jumlah parameter EfficientNet-B1 menjadi 7,8 juta dengan kebutuhan komputasi sebesar 0,70 miliar FLOPs, lebih tinggi dibandingkan EfficientNet-B0. Meskipun jumlah kanal pada *feature map* akhirnya sama dengan EfficientNet-B0, EfficientNet-B1 memiliki lebih banyak pengulangan blok sehingga dapat membentuk representasi fitur yang lebih bertingkat. Pada konteks yang lebih dekat dengan penelitian ini, EfficientNet-B1 juga telah menunjukkan efektivitas tinggi dalam mendeteksi wajah *AI-generated* pada kondisi dunia nyata [11].

EfficientNet-B2 memiliki jumlah blok MBConv yang sama dengan EfficientNet-B1, yaitu 23 blok, tetapi menggunakan jaringan yang lebih lebar serta resolusi masukan bawaan yang lebih tinggi. Peningkatan lebar tersebut menghasilkan *feature map* akhir dengan 1408 kanal, yaitu berukuran $9 \times 9 \times 1408$. Meskipun memiliki jumlah blok MBConv yang sama dengan EfficientNet-B1, peningkatan lebar jaringan dan resolusi masukan menyebabkan jumlah parameter EfficientNet-B2 meningkat menjadi 9,2 juta dengan kebutuhan komputasi sebesar 1,0 miliar FLOPs. Dengan demikian, perbedaan utama antara EfficientNet-B1 dan EfficientNet-B2 tidak hanya terletak pada resolusi, tetapi juga pada jumlah kanal yang digunakan untuk merepresentasikan fitur. EfficientNet-B2 telah dievaluasi bersama varian EfficientNet lainnya pada tugas deteksi *deepfake* [12]. Selain itu, studi komparatif lintas domain menunjukkan bahwa varian pada rentang awal hingga menengah, termasuk EfficientNet-B2, tetap menghasilkan performa yang kompetitif dalam klasifikasi citra [31].

EfficientNet-B3 meningkatkan kedalaman dan lebar jaringan menjadi 26 blok MBConv dengan *feature map* akhir berukuran $10 \times 10 \times 1536$. Varian ini memiliki jumlah parameter dan kebutuhan komputasi terbesar di antara empat varian yang digunakan, yaitu 12 juta parameter dan 1,8 miliar FLOPs. Hal tersebut menunjukkan bahwa EfficientNet-B3 menyediakan kapasitas representasi fitur yang lebih tinggi, tetapi disertai kebutuhan memori dan komputasi yang lebih besar [1]. Dalam beberapa studi komparatif, EfficientNet-B3 juga sering muncul sebagai varian dengan performa yang tinggi [31]. Oleh karena itu, penggunaan EfficientNet-B3 penting untuk melihat apakah peningkatan ukuran model benar-benar diikuti

oleh peningkatan performa deteksi citra wajah *AI-generated*.

B Transfer Learning

Transfer learning adalah strategi pembelajaran yang menggunakan kembali representasi atau pengetahuan dari model yang telah dilatih pada dataset besar untuk diterapkan pada tugas baru yang berbeda tetapi masih memiliki kemiripan domain [8, 32, 33, 34]. Pada tugas klasifikasi citra, strategi ini biasanya diterapkan dengan memanfaatkan model *pretrained* pada dataset besar seperti ImageNet, lalu mengadaptasikannya ke tugas klasifikasi yang lebih khusus [35].

Keunggulan utama *transfer learning* adalah kemampuannya untuk membuat pelatihan menjadi lebih efisien dan membantu meningkatkan kinerja model, khususnya ketika dataset penelitian berukuran lebih terbatas dibandingkan dataset yang digunakan pada tahap *pretraining* [32, 33, 34]. Dengan memanfaatkan bobot awal yang telah mempelajari pola visual umum, seperti tepi, tekstur, bentuk, dan struktur objek, model tidak perlu mempelajari seluruh representasi fitur dari awal [35].

Dalam penelitian ini, EfficientNet diterapkan dengan strategi *transfer learning* menggunakan bobot awal dari model yang telah dilatih pada ImageNet. Bagian *base model* berfungsi sebagai pengekstraksi fitur utama, sedangkan lapisan klasifikasi tambahan digunakan untuk menyesuaikan model terhadap tugas klasifikasi biner, yaitu membedakan citra wajah *real* dan citra wajah *fake*. Pendekatan ini dipilih karena lebih efisien secara komputasi dan sesuai untuk eksperimen yang membandingkan beberapa varian arsitektur pada dataset berukuran sedang [11, 34].

C Fungsi Aktivasi ReLU

Rectified Linear Unit (ReLU) merupakan fungsi aktivasi yang umum diterapkan pada jaringan saraf modern [36, 37]. Fungsi ini digunakan agar jaringan saraf dapat mempelajari hubungan yang bersifat non-linear dengan mekanisme yang sederhana, yaitu mempertahankan nilai masukan yang bernilai positif dan mengubah nilai masukan yang bernilai nol atau negatif menjadi nol [8]. Dengan karakteristik tersebut, penggunaan ReLU membantu jaringan membentuk representasi pola yang lebih kompleks dibandingkan operasi linear biasa.

ReLU banyak digunakan karena efisien secara komputasi dan mampu

membantu menekan permasalahan *vanishing gradient* yang umumnya terjadi pada fungsi aktivasi seperti sigmoid atau *tanh* ketika jaringan semakin dalam [8, 37]. Selain itu, sifatnya yang hanya mengaktifkan sebagian neuron juga mendukung proses pembelajaran yang lebih efektif, karena tidak semua neuron memberikan respons pada setiap masukan [36].

Dalam arsitektur jaringan saraf, ReLU umumnya diterapkan pada lapisan tersembunyi (*hidden layer*) atau *dense layer* untuk membantu model mempelajari representasi fitur yang lebih spesifik sebelum hasil pemrosesan diteruskan ke lapisan berikutnya.

D Fungsi Aktivasi Sigmoid

Pada tugas klasifikasi biner, sigmoid merupakan fungsi aktivasi yang sering diterapkan pada lapisan keluaran [8]. Fungsi sigmoid digunakan untuk memetakan nilai masukan ke nilai keluaran dalam rentang 0 hingga 1, sehingga dapat dipahami sebagai probabilitas. Dengan karakteristik tersebut, sigmoid banyak digunakan ketika model perlu memberikan tingkat keyakinan terhadap dua kemungkinan kelas yang saling eksklusif.

Fungsi sigmoid memetakan nilai masukan yang sangat kecil ke nilai keluaran yang mendekati 0, sedangkan nilai masukan yang sangat besar akan dipetakan ke nilai keluaran yang mendekati 1. Nilai di antara keduanya akan menghasilkan keluaran pada rentang tengah, sehingga perubahan pada masukan tetap dapat tercermin sebagai perubahan tingkat keyakinan model terhadap suatu kelas [8]. Dalam konteks klasifikasi biner, keluaran yang mendekati 1 menunjukkan keyakinan yang lebih kuat terhadap kelas positif, sedangkan keluaran yang mendekati 0 menunjukkan keyakinan yang lebih kuat terhadap kelas negatif.

Karena sifatnya yang menghasilkan keluaran berbentuk probabilitas, sigmoid banyak diterapkan pada lapisan keluaran model klasifikasi biner. Nilai probabilitas tersebut kemudian dapat dibandingkan dengan suatu nilai ambang (*threshold*) untuk menentukan label klasifikasi akhir.

E GlobalAveragePooling2D dan Dropout

Selain memanfaatkan *base model* EfficientNet, penelitian ini juga menggunakan lapisan *GlobalAveragePooling2D* dan *Dropout* pada bagian klasifikasi tambahan. *GlobalAveragePooling2D* berfungsi untuk menyederhanakan

dimensi keluaran *feature map* dengan mengambil nilai rata-rata pada setiap kanal fitur. Pendekatan ini membantu menghasilkan representasi fitur yang lebih ringkas serta mengurangi jumlah parameter dibandingkan penggunaan *fully connected layer* secara langsung [32, 38].

Sementara itu, Dropout merupakan metode regularisasi yang dilakukan dengan mematikan sebagian neuron secara acak selama proses pelatihan [8, 39]. Tujuan utama teknik ini adalah mengurangi ketergantungan model terhadap neuron tertentu, sehingga membantu menekan risiko *overfitting* dan meningkatkan kemampuan generalisasi model terhadap data baru [39]. Dalam arsitektur klasifikasi, kombinasi *GlobalAveragePooling2D*, *Dense layer*, dan *Dropout* sering diterapkan agar model tidak terlalu bergantung pada seluruh neuron selama pelatihan, tetapi tetap mampu menjalankan tugas klasifikasi dengan baik.

2.3.2 Gradient-weighted Class Activation Mapping

Gradient-weighted Class Activation Mapping (Grad-CAM) merupakan metode interpretabilitas visual yang digunakan untuk menunjukkan wilayah citra yang memberikan kontribusi terhadap prediksi kelas tertentu pada model CNN. Grad-CAM memanfaatkan gradien kelas target dan aktivasi fitur pada lapisan konvolusi untuk menghasilkan peta lokalisasi yang bersifat spesifik terhadap kelas [40]. Metode ini diterapkan setelah model selesai dilatih tanpa memerlukan perubahan arsitektur maupun pelatihan ulang. Penelitian terkini mengenai metode berbasis *Class Activation Mapping* juga menempatkan peta aktivasi sebagai pendekatan untuk menjelaskan hubungan antara representasi fitur internal dan keputusan model [41, 42]. Dalam penelitian ini, Grad-CAM ditujukan untuk membantu mengamati apakah prediksi citra wajah *real* dan *fake* didukung oleh wilayah wajah yang relevan atau justru dipengaruhi oleh bagian lain, seperti latar belakang.

Penentuan Grad-CAM diawali dengan memilih kelas target yang akan dijelaskan dan lapisan konvolusi yang digunakan sebagai sumber *feature map*. Lapisan konvolusi terakhir umumnya dipilih karena masih mempertahankan informasi spasial sekaligus telah memuat representasi fitur tingkat tinggi yang berkaitan dengan hasil klasifikasi [40, 41]. Selanjutnya, gradien skor kelas target dihitung terhadap setiap kanal pada *feature map* dan dirata-ratakan pada seluruh posisi spasial untuk menentukan tingkat kepentingan setiap kanal. Bobot kepentingan tersebut digunakan untuk menggabungkan seluruh kanal *feature map*,

sedangkan kontribusi negatif dihilangkan agar visualisasi berfokus pada fitur yang mendukung kelas target. Peta yang dihasilkan kemudian dinormalisasi, diperbesar mengikuti ukuran citra masukan, dan ditumpangkan pada citra asli dalam bentuk *heatmap*.

Wilayah dengan intensitas Grad-CAM yang lebih tinggi menunjukkan kontribusi positif yang relatif lebih besar terhadap kelas target, tetapi intensitas tersebut bukan nilai probabilitas dan tidak membuktikan bahwa wilayah tersebut merupakan penyebab pasti dari prediksi model. Grad-CAM menghasilkan penjelasan untuk satu citra atau prediksi tertentu sehingga hasilnya tidak dapat langsung digunakan untuk menggambarkan perilaku model secara keseluruhan [43]. Selain itu, kualitas suatu penjelasan visual tidak cukup dinilai hanya berdasarkan tampilan *heatmap*, karena konsistensi dan keterkaitannya dengan keputusan model juga perlu dipertimbangkan [42]. Oleh karena itu, Grad-CAM digunakan sebagai analisis kualitatif untuk melengkapi nilai prediksi dan metrik evaluasi, bukan sebagai metode klasifikasi atau metrik performa model.

2.4 Data Splitting

Pembagian data (*data splitting*) merupakan tahapan fundamental dalam proses pembelajaran terawasi (*supervised learning*), di mana dataset dipisahkan ke dalam beberapa subset yang memiliki fungsi berbeda selama proses pengembangan dan evaluasi model [8]. Secara umum, subset tersebut mencakup data pelatihan (*training set*), data validasi (*validation set*), dan data pengujian (*testing set*) [32, 44, 45]. Data pelatihan berperan sebagai sumber utama bagi model untuk mempelajari pola pada dataset, sedangkan data validasi digunakan selama pelatihan untuk mengamati kinerja model dan mendukung penyesuaian konfigurasi. Adapun data pengujian disimpan terpisah untuk menilai kemampuan model pada data yang tidak digunakan dalam proses pengembangan [8, 46].

Pembagian data bertujuan menghasilkan pengukuran performa model yang lebih adil dan tidak hanya mencerminkan keberhasilan model pada data pelatihan. Apabila pemisahan data tidak dilakukan dengan tepat, model dapat mengalami *overfitting*, yaitu keadaan ketika model terlalu mengikuti karakteristik data pelatihan hingga kinerjanya menurun saat menghadapi data baru [6, 8]. Oleh karena itu, data pengujian perlu dipisahkan dari proses pelatihan maupun penyesuaian model agar evaluasi akhir benar-benar merefleksikan kemampuan generalisasi model [44].

Pendekatan yang umum dipakai dalam pembagian data adalah *holdout method*, yaitu pendekatan di mana sebagian data dipisahkan sejak awal sebagai data pengujian akhir dan tidak digunakan selama proses pengembangan model [44, 45]. Dalam kerangka ini, data yang tersisa digunakan untuk membangun dan memvalidasi model, sedangkan *test set* disimpan untuk evaluasi akhir. Pendekatan ini banyak digunakan karena relatif sederhana, efisien secara komputasi, dan tetap mampu memberikan gambaran performa model apabila pembagian data dilakukan secara tepat [32, 45]. Pada penelitian ini, prinsip *holdout method* diterapkan dengan membagi dataset ke dalam subset *train-validation* dan subset *testing*, kemudian sebagian dari subset *train-validation* dialokasikan sebagai data validasi.

Dalam penerapannya, tidak terdapat satu rasio pembagian data yang bersifat mutlak untuk semua penelitian. Namun demikian, rasio 70:30 dan 80:20 merupakan dua konfigurasi yang umum digunakan sebagai kompromi antara kebutuhan data pelatihan dan kebutuhan evaluasi performa [32, 44, 45]. Rasio 70:30 menyediakan proporsi data pengujian yang lebih luas, sehingga penilaian performa model dapat dilakukan menggunakan lebih banyak sampel dan estimasi generalisasi cenderung lebih stabil. Sebaliknya, rasio 80:20 menyediakan proporsi data pelatihan yang lebih luas, sehingga model memiliki peluang lebih besar untuk mempelajari representasi fitur dari data yang tersedia [32, 45]. Dengan demikian, rasio 70:30 lebih menekankan kekuatan evaluasi, sedangkan rasio 80:20 lebih menekankan ketersediaan data untuk proses pembelajaran model.

Sejumlah penelitian terdahulu juga menunjukkan bahwa rasio 70:30 dan 80:20 sering digunakan dalam eksperimen klasifikasi citra. Bichri *et al.* [47], misalnya, melaporkan bahwa kedua rasio tersebut termasuk konfigurasi yang konsisten dalam menghasilkan performa klasifikasi yang tinggi pada beberapa dataset kustom. Sementara itu, rasio 70:30 juga digunakan dalam penelitian deteksi *deepfake* karena menyediakan pembagian yang cukup proporsional antara data untuk pelatihan dan data untuk pengujian [48]. Meskipun demikian, temuan-temuan tersebut tetap perlu diverifikasi kembali pada konteks dataset dan model yang digunakan dalam penelitian ini, karena performa suatu rasio pembagian data dapat dipengaruhi oleh karakteristik dataset, jumlah data, serta arsitektur model yang digunakan.

Selain menentukan rasio pembagian data, aspek penting lain dalam proses ini adalah menjaga distribusi kelas pada setiap subset. *Stratified split* merupakan pendekatan pembagian data yang menjaga komposisi kelas pada setiap subset agar tetap serupa dengan distribusi kelas pada dataset awal [44, 45, 49]. Pendekatan ini

penting pada tugas klasifikasi biner atau multi-kelas karena membantu memastikan bahwa setiap subset tetap merepresentasikan distribusi data secara adil. Pada kasus klasifikasi biner seperti penelitian ini, *stratified split* digunakan untuk mengontrol komposisi yang sepadan antara kelas *real* dan kelas *fake* pada subset *training*, *validation*, dan *testing*. Dengan demikian, hasil pelatihan dan evaluasi tidak terlalu dipengaruhi oleh ketimpangan distribusi kelas pada salah satu subset, sehingga performa model yang diperoleh menjadi lebih reliabel.

Dalam eksperimen *deep learning*, data validasi umumnya dibentuk dari sebagian data pengembangan model setelah pembagian utama antara data *training* dan data *testing* dilakukan. Metode ini memungkinkan model tetap memiliki data pelatihan yang memadai, sekaligus menyediakan subset khusus untuk memantau performa selama pelatihan, menyesuaikan hiperparameter, dan mendukung mekanisme seperti *early stopping* [32, 46]. Proporsi data validasi yang umum digunakan dalam literatur berkisar antara 10% hingga 20% dari data pelatihan, dengan pertimbangan bahwa proporsi tersebut cukup untuk memantau kecenderungan *overfitting* tanpa mengurangi jumlah data pelatihan secara signifikan [32, 45, 50].

Dalam konteks penelitian ini, rasio 70:30 dan 80:20 digunakan sebagai pembagian utama antara subset *train-validation* dan subset *testing*, kemudian 10% dari subset *train-validation* dialokasikan kembali sebagai data validasi. Pemilihan proporsi validasi sebesar 10% dilakukan dengan mempertimbangkan ukuran dataset yang digunakan serta kebutuhan untuk menjaga jumlah data pelatihan tetap memadai. Dengan pendekatan tersebut, model memperoleh data pelatihan untuk mempelajari pola visual, data validasi untuk memantau proses pelatihan, serta data pengujian yang terpisah untuk evaluasi akhir. Oleh karena itu, penggunaan *holdout method* yang dipadukan dengan *stratified split* dan pembentukan data validasi dari subset *train-validation* memungkinkan analisis pengaruh rasio pembagian data terhadap performa model dilakukan secara lebih terstruktur dan objektif.

2.5 Optimizer dan Loss Function

Dalam proses pelatihan jaringan saraf tiruan, model memerlukan dua komponen penting, yaitu *optimizer* dan *loss function*. *Loss function* berperan sebagai ukuran kesalahan antara hasil prediksi model dan label sebenarnya, sedangkan *optimizer* digunakan untuk memperbarui bobot model agar nilai kesalahan tersebut semakin kecil dari satu iterasi ke iterasi berikutnya [8, 32].

Kombinasi kedua komponen ini sangat menentukan arah pembelajaran dan kecepatan konvergensi model selama pelatihan.

2.5.1 Optimizer Adam

Adam (*Adaptive Moment Estimation*) adalah algoritma optimasi yang umum diterapkan pada pelatihan model *deep learning* karena memanfaatkan prinsip *momentum* dan *adaptive learning rate* [51]. Secara umum, Adam memperbarui parameter model dengan memanfaatkan informasi gradien saat ini sekaligus riwayat gradien sebelumnya, sehingga proses pembelajaran dapat berlangsung lebih stabil dan efisien [51].

Melalui mekanisme tersebut, Adam tidak hanya mempertimbangkan arah perubahan gradien, tetapi juga menyesuaikan besar langkah pembaruan untuk setiap parameter secara adaptif. Karakteristik ini membuat Adam mampu bekerja dengan baik pada berbagai permasalahan optimasi, terutama ketika model memiliki banyak parameter dan distribusi gradien yang berbeda-beda pada setiap bagian jaringan [51]. Dibandingkan metode yang lebih sederhana seperti *stochastic gradient descent* standar, Adam umumnya membantu proses pelatihan mencapai kondisi konvergen dengan lebih stabil, terutama pada fase awal pelatihan.

Karena sifatnya yang adaptif, Adam menjadi salah satu optimizer yang paling umum digunakan dalam pelatihan jaringan saraf modern, termasuk pada skenario *transfer learning*. Penggunaan optimizer ini membantu proses pembaruan bobot berlangsung secara lebih efisien, terutama ketika model memerlukan penyesuaian parameter tanpa harus melakukan perubahan yang terlalu besar pada setiap iterasi.

2.5.2 Loss Function Binary Crossentropy

Pada tugas klasifikasi biner, salah satu *loss function* yang paling umum digunakan adalah *binary crossentropy*. Fungsi ini digunakan untuk mengukur seberapa besar perbedaan antara probabilitas prediksi model dan label aktual pada dua kelas [8]. Dalam konteks klasifikasi biner, model umumnya menghasilkan keluaran berupa nilai probabilitas antara 0 hingga 1, sehingga diperlukan fungsi *loss* yang mampu mengevaluasi kesesuaian antara nilai probabilitas tersebut dan label sebenarnya.

Binary crossentropy bekerja dengan memberikan penalti terhadap prediksi

yang jauh dari label aktual. Apabila model memberikan probabilitas yang mendekati label sebenarnya, nilai *loss* akan semakin kecil. Sebaliknya, apabila probabilitas prediksi model berbeda jauh dari label aktual, nilai *loss* akan meningkat. Karakteristik ini membuat *binary crossentropy* sesuai digunakan bersama lapisan keluaran sigmoid, karena keduanya berkaitan dengan prediksi berbasis probabilitas pada rentang 0 hingga 1.

Fungsi *loss* ini banyak digunakan pada model klasifikasi biner karena dapat membantu proses pelatihan dalam menyesuaikan bobot model agar prediksi yang dihasilkan semakin mendekati label aktual. Dengan demikian, *binary crossentropy* tidak hanya berperan sebagai ukuran kesalahan, tetapi juga menjadi dasar bagi optimizer dalam memperbarui parameter model selama proses pelatihan.

Dalam penelitian ini, *binary crossentropy* digunakan karena tugas yang dilakukan merupakan klasifikasi biner antara citra wajah *real* dan citra wajah *fake*. Dengan menggunakan fungsi *loss* ini, proses pelatihan diarahkan untuk meminimalkan kesalahan probabilistik pada prediksi dua kelas tersebut, sehingga model dapat menghasilkan keputusan klasifikasi yang lebih sesuai terhadap label data.

2.6 Confusion Matrix dan Metrik Evaluasi

Confusion matrix adalah metode evaluasi untuk melihat kinerja model klasifikasi dengan membandingkan label aktual dan hasil prediksi pada data uji [44, 45]. Matriks ini menyajikan hasil klasifikasi dalam bentuk tabel terstruktur sehingga memungkinkan analisis terhadap jenis kesalahan prediksi yang dilakukan model.

Pada kasus klasifikasi biner, confusion matrix terdiri atas empat komponen utama, yaitu *True Positive* (TP), *True Negative* (TN), *False Positive* (FP), dan *False Negative* (FN). *True Positive* menyatakan jumlah data positif yang diprediksi benar sebagai positif, sedangkan *True Negative* menyatakan jumlah data negatif yang diprediksi benar sebagai negatif. *False Positive* terjadi ketika data negatif diprediksi sebagai positif, dan *False Negative* terjadi ketika data positif diprediksi sebagai negatif [32].

Dalam penelitian ini, citra wajah *fake* ditetapkan sebagai kelas positif, sedangkan citra wajah *real* ditetapkan sebagai kelas negatif. Dengan demikian, *True Positive* menunjukkan citra *fake* yang dikenali dengan benar, sedangkan *False Negative* menunjukkan citra *fake* yang keliru diprediksi sebagai *real*.

Berdasarkan komponen tersebut, beberapa metrik evaluasi dapat dihitung untuk mengukur performa model secara kuantitatif. Akurasi (*accuracy*) digunakan untuk menunjukkan seberapa besar bagian prediksi yang benar dari seluruh data uji. Perhitungan akurasi ditunjukkan pada Persamaan 2.2.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.2}$$

Pada Persamaan 2.2, nilai akurasi diperoleh dari jumlah prediksi benar, yaitu *True Positive* (TP) dan *True Negative* (TN), dibandingkan dengan seluruh data yang dievaluasi. Metrik ini digunakan untuk melihat tingkat keberhasilan model dalam melakukan klasifikasi.

Selain akurasi, metrik lain yang juga diterapkan adalah *precision*. Metrik ini menunjukkan ketepatan model ketika memberikan prediksi positif. Dalam artian, precision menggambarkan perbandingan antara prediksi positif yang benar dan seluruh data yang diklasifikasikan sebagai kelas positif [52]. Precision dihitung menggunakan Persamaan 2.3.

$$Precision = \frac{TP}{TP + FP} \tag{2.3}$$

Pada Persamaan 2.3, precision diperoleh dari perbandingan antara *True Positive* (TP) dengan seluruh data yang diprediksi sebagai positif, yaitu *True Positive* (TP) dan *False Positive* (FP). Nilai precision yang tinggi menunjukkan bahwa model memiliki tingkat kesalahan prediksi positif yang rendah.

Metrik berikutnya adalah *recall*, yang digunakan untuk menilai sejauh mana model mampu menemukan data yang benar-benar termasuk kelas positif. Recall menggambarkan perbandingan antara data positif yang berhasil dikenali dengan benar dan seluruh data positif aktual [52]. Recall dihitung menggunakan Persamaan 2.4.

$$Recall = \frac{TP}{TP + FN} \tag{2.4}$$

Pada Persamaan 2.4, recall diperoleh dari perbandingan antara *True Positive* (TP) dengan seluruh data positif aktual, yaitu *True Positive* (TP) dan *False Negative* (FN). Nilai recall yang tinggi menunjukkan bahwa model memiliki kemampuan

yang baik dalam mendeteksi kelas positif dan menghasilkan jumlah *False Negative* yang rendah.

Untuk memperoleh keseimbangan antara precision dan recall, digunakan metrik *F1-score*. F1-score merupakan rata-rata harmonik dari precision dan recall, sehingga metrik ini berguna ketika diperlukan ukuran performa yang mempertimbangkan ketepatan prediksi positif sekaligus kemampuan model dalam mengenali data positif [52]. F1-score dihitung menggunakan Persamaan 2.5.

$$F1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \tag{2.5}$$

Pada Persamaan 2.5, F1-score akan bernilai tinggi apabila precision dan recall sama-sama memiliki nilai yang tinggi. Dengan demikian, F1-score dapat memberikan gambaran yang lebih seimbang terhadap performa model, khususnya ketika analisis tidak hanya berfokus pada jumlah prediksi benar secara keseluruhan, tetapi juga pada keseimbangan antara kesalahan prediksi positif dan kesalahan dalam mengenali kelas positif.

Penggunaan beberapa metrik evaluasi secara bersamaan memberikan gambaran yang lebih komprehensif terhadap performa model karena ketepatan prediksi positif dan kemampuan mendeteksi kelas positif dapat dianalisis secara bersamaan [52]. Dengan demikian, evaluasi berbasis *confusion matrix* memungkinkan analisis yang lebih mendalam terhadap kemampuan model dalam melakukan klasifikasi citra.