BAB 2
LANDASAN TEORI

Bab ini membahas teori-teori yang menjadi dasar dalam penelitian mengenai perbandingan model CNN *baseline*, MobileNetV2, dan MobileNetV3 untuk klasifikadi pneumonia dari citra *Chest X-Ray*. Pembahasan meliputi konsep pneumonia, citra *Chest X-Ray*, *deep learning*, CNN, arsitektur MobileNet, arsitektur MobileNetV2, arsitektur MobileNetV3, serta metode *Explainable Artificial Intelligence* (XAI) yang digunakan untuk menjelaskan hasil prediksi model. Selain itu, bab ini juga membahas metrik evaluasi model klasifikasi, dan optimasi model *deep learning*. Pada bagian akhir, disajikan penelitian sebelumnya yang relevan sebagai dasar pembanding dan pendukung penelitian ini.

2.1 Pneumonia

Pneumonia merupakan penyakit infeksi pada sistem pernapasan yang menyerang jaringan paru-paru dan menyebabkan terjadinya inflamasi pada alveoli sebagai tempat utama pertukaran gas. Proses peradangan tersebut dapat mengakibatkan alveoli terisi oleh cairan atau eksudat sehingga menghambat pertukaran oksigen dan karbon dioksida secara optimal. Akibatnya, penderita dapat mengalami berbagai gejala, seperti batuk, demam, sesak napas, nyeri dada, serta penurunan fungsi pernapasan. Dalam praktik klinis, diagnosis pneumonia umumnya ditegakkan melalui kombinasi evaluasi gejala,pemeriksaan fisik, dan pemeriksaan penunjang, termasuk penggunaan citra *Chest X-Ray* atau rontgen dada.

Pemeriksaan *Chest X-Ray* banyak dimanfaatkan sebagai sarana penunjang diagnosis karena mampu memberikan representasi visual kondisi paru-paru secara cepat dan informatif. Pada citra rontgen dada, keberadaan pneumonia umumnya ditunjukkan oleh munculnya area opasitas atau infiltrasi pada jaringan paru. Meskipun demikian, manifestasi radiologis pneumonia dapat menunjukkan variasi pola dan dalam beberapa kasus memiliki karakteristik yang menyerupai penyakit paru lainnya. Oleh sebab itu, proses interpretasi citra *Chest X-Ray* memerlukan tingkat ketelitian yang tinggi serta didukung oleh kompetensi dan pengalaman tenaga medis agar diagnosis dapat dilakukan secara akurat [8, 9].

Perkembangan kecerdasan buatan, khususnya *deep learning*, memberikan peluang untuk membantu proses klasifikasi pneumonia dari citra *Chest X-Ray*.

Beberapa penelitian menunjukkan bahwa pendekatan berbasis *deep learning* dapat digunakan untuk mengklasifikasikan citra paru normal dan pneumonia dengan performa yang baik [10, 23, 24]. Dengan demikian, pneumonia menjadi salah satu objek penelitian yang relevan dalam pengembangan sistem klasifikasi citra medis berbasis kecerdasan buatan.

2.2 Citra Chest X-Ray

Citra *Chest X-Ray* merupakan citra medis yang dihasilkan melalui pemeriksaan rontgen dada. Citra ini digunakan untuk melihat struktur anatomi dada, seperti paru-paru, tulang rusuk, jantung, dan diafragma. Dalam deteksi pneumonia, citra *Chest X-Ray* digunakan untuk mengamati perubahan visual pada paru-paru, seperti bercak putih, area infiltrat, atau opasitas yang dapat menunjukkan adanya infeksi. Citra *Chest X-Ray* memiliki beberapa keunggulan, antara lain proses pemeriksaan yang relatif cepat, biaya yang lebih rendah dibandingkan beberapa modalitas citra medis lain, serta ketersediaannya yang luas di fasilitas kesehatan. Meskipun demikian, citra rontgen dada juga memiliki beberapa tantangan, seperti variasi kualitas citra, perbedaan posisi pasien, perbedaan intensitas pencahayaan, variasi resolusi, serta kemungkinan adanya *noise* atau artefak. Faktor-faktor tersebut dapat memengaruhi performa model dalam melakukan klasifikasi citra [11, 25].

Dalam penelitian berbasis *deep learning*, citra *Chest X-Ray* umumnya melalui beberapa tahap pengolahan sebelum digunakan sebagai masukan model. Tahapan tersebut meliputi pemeriksaan kualitas data, penghapusan data rusak atau duplikat, penyesuaian ukuran citra, normalisasi piksel, augmentasi data, dan pembagian dataset menjadi data latih, validasi, dan uji. Preprocessing yang tepat dapat membantu model mempelajari pola visual yang lebih relevan serta mengurangi risiko kesalahan klasifikasi [16, 26].

2.3 Deep Learning untuk Klasifikasi Citra

Deep learning merupakan perkembangan dari *machine learning* yang menggunakan arsitektur jaringan saraf tiruan dengan banyak lapisan untuk mempelajari pola dan representasi data secara hierarkis. Pada bidang klasifikasi citra, metode ini mampu mengenali fitur-fitur visual yang terdapat pada gambar dan mengklasifikasikannya ke dalam kelas tertentu. Berbeda dengan pendekatan

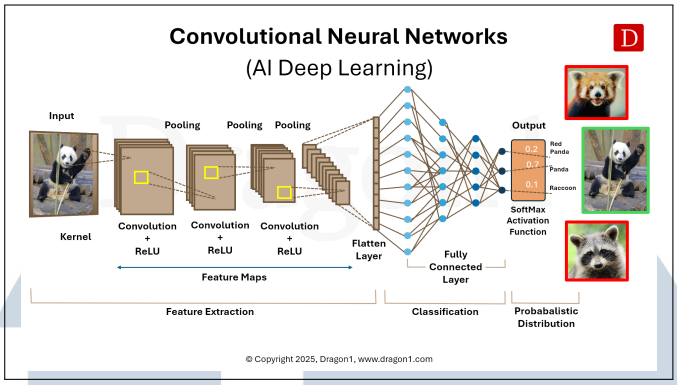
tradisional yang memerlukan ekstraksi fitur secara manual, model *deep learning* dapat secara otomatis mengekstraksi fitur yang relevan selama proses pelatihan, sehingga menghasilkan performa yang lebih efektif dalam berbagai tugas pengenalan citra.

Dalam bidang citra medis, *deep learning* banyak digunakan karena mampu menangani data citra yang kompleks. Pada kasus deteksi pneumonia, model *deep learning* dapat dilatih untuk membedakan citra paru normal dan citra paru yang menunjukkan indikasi pneumonia. Pendekatan ini telah banyak digunakan dalam penelitian sebelumnya, baik menggunakan CNN sederhana, model *transfer learning*, maupun model *ensemble* [8, 9, 10]. Secara umum, proses klasifikasi citra menggunakan *deep learning* terdiri dari beberapa tahap. Pertama, dataset citra dikumpulkan dan diberi label sesuai kelasnya. Kedua, citra diproses agar memiliki format dan ukuran yang sesuai dengan kebutuhan model. Ketiga, model dilatih menggunakan data latih untuk mempelajari pola visual dari setiap kelas. Keempat, model dievaluasi menggunakan data uji untuk mengetahui kemampuannya dalam melakukan klasifikasi pada data yang belum digunakan dalam proses pelatihan. Keunggulan *deep learning* dalam klasifikasi citra adalah kemampuannya dalam melakukan ekstraksi fitur secara otomatis. Namun, model *deep learning* juga memiliki kelemahan, seperti membutuhkan data yang cukup banyak, rentan terhadap ketidakseimbangan data, dan sulit dijelaskan karena sifatnya yang sering dianggap sebagai *black box* [18, 17]. Oleh karena itu, selain mengevaluasi performa klasifikasi, diperlukan pendekatan interpretabilitas seperti XAI.

2.4 Convolutional Neural Network

Convolutional Neural Network (CNN) merupakan salah satu metode *deep learning* yang banyak digunakan dalam pengolahan citra karena memiliki kemampuan untuk mengekstraksi fitur secara otomatis tanpa memerlukan proses *feature engineering* secara manual. Berbeda dengan metode pembelajaran mesin konvensional yang bergantung pada fitur hasil ekstraksi manual, CNN mempelajari representasi fitur secara hierarkis mulai dari pola sederhana, seperti tepi dan tekstur, hingga pola yang lebih kompleks, seperti bentuk objek. Kemampuan tersebut menjadikan CNN banyak diterapkan pada berbagai permasalahan klasifikasi citra, termasuk klasifikasi citra medis seperti *Chest X-Ray*.

Arsitektur dasar *Convolutional Neural Network* ditunjukkan pada Gambar 2.1.



Gambar 2.1. Arsitektur *Convolutional Neural Network* (CNN) [1].

Berdasarkan Gambar 2.1, citra masukan diproses melalui lapisan *convolution*, fungsi aktivasi, dan *pooling* untuk mengekstraksi fitur, kemudian diteruskan ke lapisan *fully connected* guna menghasilkan prediksi kelas.

2.4.1 Arsitektur CNN

Secara umum, arsitektur CNN terdiri atas beberapa lapisan utama, yaitu *input layer*, *convolution layer*, fungsi aktivasi *Rectified Linear Unit* (ReLU), *pooling layer*, *flatten layer*, *fully connected layer*, dan *output layer*. Setiap lapisan memiliki fungsi yang berbeda dalam proses ekstraksi fitur hingga menghasilkan keputusan klasifikasi.

Berdasarkan Gambar 2.1, citra masukan diproses menggunakan *convolution layer* untuk menghasilkan *feature map*. Selanjutnya, fungsi aktivasi ReLU diterapkan untuk memperkenalkan sifat nonlinier, kemudian *pooling layer* mengurangi dimensi spasial *feature map* agar proses komputasi lebih efisien. Kombinasi *convolution–ReLU–pooling* dilakukan secara berulang sehingga fitur yang dipelajari berkembang dari pola sederhana, seperti tepi dan tekstur, menjadi representasi objek yang lebih kompleks. Setelah proses ekstraksi fitur selesai, *feature map* diubah menjadi vektor satu dimensi melalui *flatten layer* dan diteruskan ke *fully connected layer*. Pada tahap akhir, fungsi aktivasi *Softmax* menghasilkan probabilitas setiap kelas, kemudian kelas dengan probabilitas terbesar dipilih sebagai hasil prediksi model.

2.4.2 Mekanisme Pembelajaran CNN

Berdasarkan Gambar 2.1, proses pembelajaran CNN terdiri atas dua tahapan utama, yaitu *forward propagation* dan *backpropagation*. Pada tahap *forward propagation*, citra masukan diproses secara berurutan melalui *convolution layer*, fungsi aktivasi ReLU, *pooling layer*, *flatten layer*, *fully connected layer*, dan fungsi aktivasi *Softmax* hingga menghasilkan probabilitas setiap kelas.

Selama proses tersebut, lapisan konvolusi mengekstraksi fitur visual, sedangkan kombinasi *convolution–ReLU–pooling* yang dilakukan berulang memungkinkan model mempelajari representasi fitur secara hierarkis, mulai dari tepi dan tekstur hingga pola yang berkaitan dengan objek pada citra. Selanjutnya, hasil prediksi dibandingkan dengan label sebenarnya menggunakan *loss function*. Nilai *loss* kemudian digunakan pada proses *backpropagation* untuk menghitung gradien setiap parameter. Gradien tersebut dimanfaatkan oleh algoritma optimasi untuk memperbarui bobot kernel dan *fully connected layer* sehingga nilai *loss* semakin kecil dan performa model meningkat pada setiap iterasi pelatihan.

2.4.3 Convolution Layer

Berdasarkan Gambar 2.1, proses pertama pada tahap *feature extraction* adalah *convolution layer*. Pada lapisan ini, citra masukan diproses menggunakan sejumlah *kernel* atau *filter* yang digeser (*sliding*) secara sistematis pada seluruh area citra. Setiap kernel melakukan operasi perkalian elemen demi elemen dengan nilai piksel pada wilayah pengamatan, kemudian seluruh hasil perkalian dijumlahkan untuk menghasilkan satu nilai pada *feature map*. Proses tersebut diulangi hingga seluruh citra selesai dipindai sehingga dihasilkan sekumpulan *feature map* yang merepresentasikan karakteristik visual, seperti tepi, tekstur, sudut, maupun pola objek.

Selama proses pelatihan, bobot setiap kernel diperbarui melalui *backpropagation* untuk meminimalkan nilai *loss*. Akibatnya, kernel secara bertahap mampu mempelajari fitur yang semakin relevan dengan objek yang diklasifikasikan. Dengan demikian, proses ekstraksi fitur dilakukan secara otomatis tanpa memerlukan *feature engineering* secara manual.

Secara matematis, proses konvolusi dinyatakan oleh Rumus 2.1.

$$Y(i,j) = \sum_{m=0}^{k-1} \sum_{n=0}^{k-1} X(i+m, j+n) \times K(m,n) \tag{2.1}$$

Pada Rumus 2.1, $X(i,j)$ menyatakan citra masukan (*input image*), $K(m,n)$ menyatakan bobot kernel yang digunakan pada proses konvolusi, $Y(i,j)$ menyatakan nilai keluaran pada *feature map*, sedangkan k menyatakan ukuran kernel yang digunakan dalam proses konvolusi. Rumus 2.1 menunjukkan bahwa setiap nilai pada *feature map* diperoleh dari penjumlahan hasil perkalian antara nilai piksel citra dengan bobot kernel pada suatu wilayah pengamatan. Secara algoritmik, satu kernel menghasilkan satu *feature map*, sehingga penggunaan lebih banyak kernel memungkinkan model mempelajari representasi fitur yang semakin beragam sebelum diteruskan ke lapisan berikutnya.

Ukuran *feature map* dipengaruhi oleh ukuran citra masukan, ukuran kernel, nilai *padding*, dan *stride*. Hubungan tersebut dinyatakan oleh Rumus 2.2.

$$O = \frac{N - K + 2P}{S} + 1 \tag{2.2}$$

Pada Rumus 2.2, O menyatakan ukuran *feature map* yang dihasilkan, N menyatakan ukuran citra masukan (*input image*), K menyatakan ukuran kernel yang digunakan pada proses konvolusi, P menyatakan nilai *padding*, sedangkan S menyatakan nilai *stride*. Rumus 2.2 menunjukkan bahwa nilai *stride* yang lebih besar akan menghasilkan *feature map* yang lebih kecil, sedangkan *padding* membantu mempertahankan informasi pada tepi citra. Oleh karena itu, pemilihan parameter konvolusi berpengaruh terhadap jumlah informasi yang dipertahankan selama proses ekstraksi fitur, sehingga model dapat mempelajari karakteristik radiologis pneumonia secara lebih optimal.

2.4.4 Fungsi Aktivasi ReLU

Berdasarkan Gambar 2.1, setiap keluaran dari *convolution layer* diproses menggunakan fungsi aktivasi *Rectified Linear Unit* (ReLU). Fungsi aktivasi ini berperan dalam memberikan sifat nonlinier pada jaringan sehingga CNN mampu mempelajari pola yang lebih kompleks. Tanpa fungsi aktivasi, seluruh lapisan pada jaringan hanya akan membentuk transformasi linear sehingga kemampuan model dalam membedakan karakteristik antar kelas menjadi terbatas.

Secara matematis, fungsi aktivasi ReLU dinyatakan oleh Rumus 2.3.

$$f(x) = \max(0,x) \tag{2.3}$$

Pada Rumus 2.3, x menyatakan nilai masukan yang diperoleh dari hasil operasi konvolusi, sedangkan $f(x)$ menyatakan nilai keluaran setelah fungsi aktivasi ReLU diterapkan. Rumus 2.3 menunjukkan bahwa apabila nilai masukan lebih kecil dari nol maka nilai keluaran akan menjadi nol, sedangkan nilai masukan yang lebih besar atau sama dengan nol akan dipertahankan. Dengan mekanisme tersebut, hanya fitur yang memberikan respons positif yang diteruskan ke lapisan berikutnya, sedangkan aktivasi yang kurang relevan akan dieliminasi.

Penerapan fungsi aktivasi ReLU setelah setiap proses konvolusi, seperti ditunjukkan pada Gambar 2.1, membantu model mempelajari representasi fitur yang lebih diskriminatif sebelum dilanjutkan ke *pooling layer*. Selain itu, ReLU memiliki kompleksitas komputasi yang rendah karena hanya melakukan operasi pembandingan terhadap nol sehingga proses pelatihan menjadi lebih cepat dibandingkan fungsi aktivasi seperti *sigmoid* maupun *hyperbolic tangent* (tanh). Penggunaan ReLU juga membantu mengurangi permasalahan *vanishing gradient* pada proses *backpropagation*, sehingga pembelajaran pada jaringan yang memiliki banyak lapisan menjadi lebih stabil.

2.4.5 Pooling Layer

Berdasarkan Gambar 2.1, setiap keluaran dari fungsi aktivasi ReLU selanjutnya diproses pada *pooling layer*. Lapisan ini berfungsi untuk mereduksi dimensi spasial dari *feature map* sehingga jumlah parameter yang harus diproses pada lapisan berikutnya menjadi lebih sedikit. Dengan berkurangnya ukuran *feature map*, kebutuhan komputasi dapat ditekan tanpa menghilangkan karakteristik penting yang telah dipelajari pada proses konvolusi.

Salah satu metode *pooling* yang paling banyak digunakan adalah *max pooling*. Metode ini bekerja dengan memilih nilai aktivasi tertinggi pada setiap wilayah pengamatan (*pooling window*) sebagai representasi dari wilayah tersebut. Dengan demikian, hanya fitur yang memiliki respons paling kuat yang dipertahankan untuk diteruskan ke lapisan berikutnya.

Secara matematis, proses *max pooling* dinyatakan oleh Rumus 2.4.

$$Y(i,j) = \max_{(m,n)\in R} X(i+m, j+n) \tag{2.4}$$

Pada Rumus 2.4, $X(i,j)$ menyatakan nilai pada *feature map*, R menyatakan

wilayah pengamatan (*pooling window*) yang digunakan dalam proses *max pooling*, sedangkan $Y(i, j)$ menyatakan nilai keluaran yang dihasilkan dari proses *max pooling*. Rumus 2.4 menunjukkan bahwa setiap nilai keluaran diperoleh dengan memilih nilai maksimum pada setiap wilayah pengamatan. Mekanisme tersebut memungkinkan informasi yang memiliki aktivasi tertinggi tetap dipertahankan, sedangkan informasi yang kurang relevan dihilangkan. Selain mengurangi ukuran data, proses *pooling* juga membantu meningkatkan ketahanan model terhadap pergeseran posisi objek (*translation invariance*) sehingga model tetap mampu mengenali karakteristik utama citra meskipun terjadi sedikit perubahan posisi objek. Seperti ditunjukkan pada Gambar 2.1, proses *pooling* dilakukan setelah setiap kombinasi *convolution layer* dan fungsi aktivasi ReLU. Kombinasi proses tersebut diulang beberapa kali sehingga fitur yang dihasilkan menjadi semakin ringkas, representatif, dan siap digunakan pada tahap klasifikasi.

2.4.6 Flatten Layer dan Fully Connected Layer

Setelah seluruh proses ekstraksi fitur selesai dilakukan, kumpulan *feature map* diubah menjadi sebuah vektor satu dimensi melalui proses *flatten*. Tahap ini tidak mengubah nilai fitur yang telah dipelajari, melainkan hanya mengubah bentuk representasinya dari matriks dua dimensi menjadi vektor satu dimensi agar dapat diproses oleh *fully connected layer*. Pada Gambar 2.1, *flatten layer* menjadi penghubung antara tahap *feature extraction* dan tahap *classification*.

Selanjutnya, vektor hasil proses *flatten* diteruskan menuju *fully connected layer*. Pada lapisan ini, setiap neuron terhubung dengan seluruh neuron pada lapisan sebelumnya sehingga seluruh fitur hasil ekstraksi dapat digabungkan menjadi representasi yang lebih komprehensif. Bobot pada setiap neuron dipelajari secara otomatis selama proses pelatihan menggunakan algoritma *backpropagation* sehingga model mampu membedakan karakteristik setiap kelas dengan lebih baik. Keluaran dari *fully connected layer* selanjutnya diteruskan menuju fungsi aktivasi *Softmax* untuk menghasilkan distribusi probabilitas setiap kelas sebagai dasar dalam proses klasifikasi.

2.4.7 Fungsi Aktivasi Softmax

Berdasarkan Gambar 2.1, keluaran dari *fully connected layer* selanjutnya diproses menggunakan fungsi aktivasi *Softmax*. Fungsi ini merupakan tahap

terakhir pada arsitektur CNN yang bertugas mengubah nilai keluaran (*logits*) menjadi distribusi probabilitas untuk setiap kelas. Dengan demikian, hasil prediksi model tidak hanya menunjukkan kelas yang dipilih, tetapi juga tingkat keyakinan model terhadap masing-masing kelas.

Secara matematis, fungsi aktivasi *Softmax* dinyatakan oleh Rumus 2.5.

$$P(y_i) = \frac{e^{z_i}}{\sum_{j=1}^{C} e^{z_j}} \tag{2.5}$$

Pada Rumus 2.5, z_i menyatakan nilai keluaran (*logit*) dari neuron ke-i, C menyatakan jumlah kelas yang digunakan dalam proses klasifikasi, sedangkan $P(y_i)$ menyatakan probabilitas suatu data termasuk ke dalam kelas ke-i. Rumus 2.5 menunjukkan bahwa setiap nilai keluaran neuron ditransformasikan menjadi probabilitas pada rentang 0 hingga 1, dengan jumlah seluruh probabilitas sama dengan satu. Pada penelitian ini terdapat dua kelas keluaran, yaitu *NORMAL* dan *PNEUMONIA*. Kelas dengan probabilitas terbesar dipilih sebagai hasil prediksi akhir model. Pada Gambar 2.1, keluaran tersebut ditunjukkan sebagai *Probabilistic Distribution* yang merepresentasikan probabilitas masing-masing kelas.

2.4.8 Penerapan CNN pada Klasifikasi Citra Medis

Dalam bidang pencitraan medis, CNN telah banyak dimanfaatkan untuk mendukung identifikasi penyakit berdasarkan karakteristik visual pada citra *Chest X-Ray*. Berbagai penelitian telah menerapkan CNN maupun pengembangannya untuk mendeteksi pneumonia, mulai dari VGG16, *Ensemble CNN*, hingga *Attention CNN* [12, 13, 14]. Hasil penelitian tersebut menunjukkan bahwa CNN mampu mempelajari karakteristik radiologis yang berhubungan dengan keberadaan pneumonia secara efektif sehingga menjadi salah satu pendekatan yang banyak digunakan pada klasifikasi citra medis.

Pada penelitian ini, CNN digunakan sebagai model *baseline* untuk mengevaluasi performa arsitektur konvensional sebelum dibandingkan dengan MobileNetV2 dan MobileNetV3. Citra *Chest X-Ray* berukuran 512×512 diproses melalui lapisan *convolution*, ReLU, dan *pooling* untuk mengekstraksi fitur, kemudian dilanjutkan ke *fully connected layer* dan fungsi aktivasi *Softmax* untuk menghasilkan probabilitas dua kelas, yaitu *NORMAL* dan *PNEUMONIA*. Penggunaan model *baseline* memungkinkan proses evaluasi dilakukan secara lebih objektif sehingga peningkatan performa yang dihasilkan oleh MobileNetV2 dan

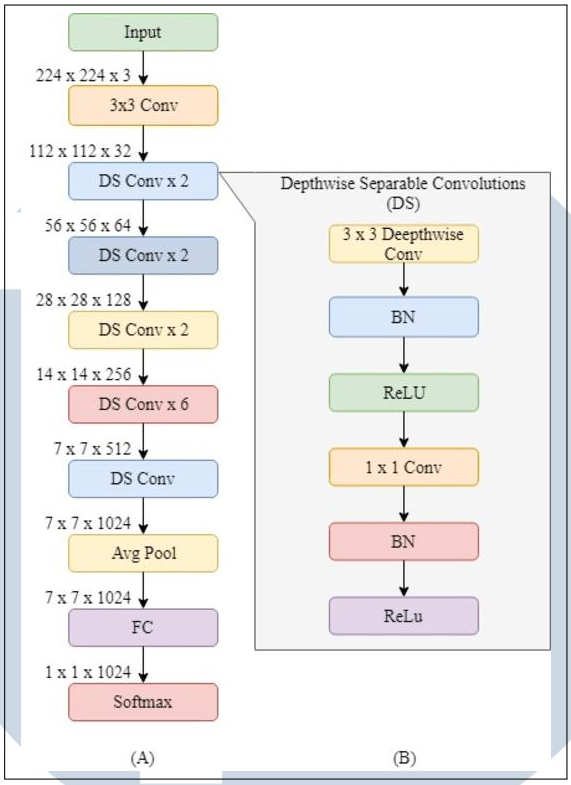
MobileNetV3 dapat dianalisis secara komprehensif.

Pada penelitian ini, konsep *depthwise separable convolution* pada MobileNet digunakan sebagai dasar arsitektur MobileNetV2 dan MobileNetV3 yang diterapkan untuk klasifikasi pneumonia pada citra *Chest X-Ray*. Pemisahan proses konvolusi menjadi *depthwise convolution* dan *pointwise convolution* memungkinkan pengurangan jumlah parameter dan kebutuhan komputasi tanpa mengurangi kemampuan model dalam mengekstraksi fitur. Dengan mekanisme tersebut, MobileNet diharapkan mampu menghasilkan proses klasifikasi yang lebih efisien dibandingkan CNN *baseline* sehingga perbandingan performa ketiga model dapat dilakukan secara objektif.

2.5 Arsitektur MobileNet

MobileNet merupakan keluarga arsitektur *Convolutional Neural Network* (CNN) yang dirancang untuk menghasilkan model klasifikasi citra yang ringan dan efisien. Arsitektur ini dikembangkan untuk mengurangi kebutuhan komputasi pada CNN konvensional sehingga dapat diterapkan pada perangkat dengan sumber daya terbatas, seperti *mobile device* dan sistem *embedded*. Efisiensi tersebut diperoleh melalui penerapan konsep *depthwise separable convolution*, yang mampu mengurangi jumlah parameter dan operasi komputasi tanpa mengurangi kemampuan model dalam mengekstraksi fitur secara signifikan [2].

Arsitektur dasar MobileNet ditunjukkan pada Gambar 2.2.

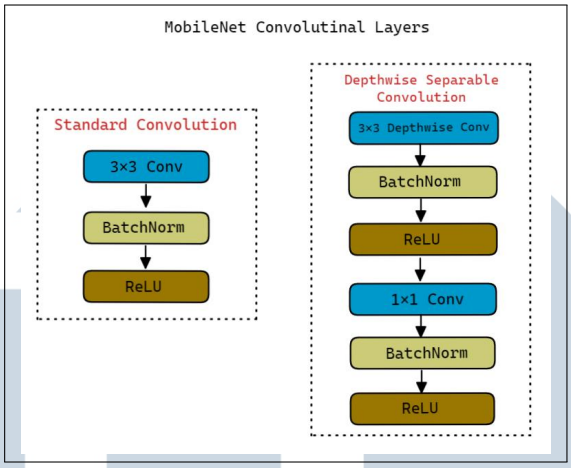


Gambar 2.2. Arsitektur MobileNet [2].

Berdasarkan Gambar 2.2, MobileNet terdiri atas serangkaian blok *depthwise separable convolution* yang disusun secara berurutan. Setiap blok melakukan ekstraksi fitur melalui proses *depthwise convolution* untuk mempelajari informasi spasial pada setiap kanal, kemudian dilanjutkan dengan *pointwise convolution* menggunakan kernel berukuran 1×1 untuk menggabungkan informasi antar kanal. Di antara kedua proses tersebut diterapkan *batch normalization* dan fungsi aktivasi ReLU untuk meningkatkan stabilitas pelatihan dan kemampuan model dalam mempelajari hubungan nonlinier.

Melalui mekanisme tersebut, MobileNet mampu menghasilkan representasi fitur yang baik dengan jumlah parameter dan kebutuhan komputasi yang jauh lebih rendah dibandingkan CNN konvensional.

Perbandingan antara *standard convolution* dan *depthwise separable convolution* pada MobileNet ditunjukkan pada Gambar 2.3.



Gambar 2.3. Perbandingan *standard convolution* dan *depthwise separable convolution* pada MobileNet [2].

Berdasarkan Gambar 2.3, *standard convolution* melakukan ekstraksi fitur spasial dan penggabungan kanal dalam satu operasi konvolusi. Sebaliknya, MobileNet memisahkan proses tersebut menjadi *depthwise convolution* dan *pointwise convolution* sehingga jumlah parameter dan operasi komputasi dapat dikurangi secara signifikan. Konsep inilah yang menjadi dasar pengembangan MobileNetV2 dan MobileNetV3. Dalam bidang pencitraan medis, MobileNet telah banyak dimanfaatkan untuk klasifikasi citra *Chest X-Ray* karena mampu memberikan keseimbangan antara efisiensi komputasi dan akurasi klasifikasi [15, 27, 16]. Oleh karena itu, MobileNet dipilih sebagai dasar pengembangan MobileNetV2 dan MobileNetV3 yang digunakan pada penelitian ini.

2.5.1 Mekanisme MobileNet

Berdasarkan Gambar 2.2, proses pembelajaran MobileNet terdiri atas dua tahapan utama, yaitu *feature extraction* dan *classification*. Pada tahap *feature extraction*, citra masukan diproses melalui serangkaian blok *depthwise separable convolution*. Setiap blok melakukan *depthwise convolution* untuk mengekstraksi informasi spasial pada setiap kanal, kemudian dilanjutkan dengan *pointwise convolution* untuk menggabungkan informasi antar kanal. Di antara kedua proses tersebut diterapkan *batch normalization* dan fungsi aktivasi ReLU agar proses pelatihan lebih stabil dan mampu mempelajari hubungan nonlinier.

Kombinasi blok tersebut dilakukan secara berulang sehingga model mampu mempelajari representasi fitur secara bertahap, mulai dari fitur sederhana hingga

pola yang lebih kompleks. Selanjutnya, *feature map* diteruskan menuju tahap klasifikasi menggunakan *Global Average Pooling* atau *flatten*, kemudian diproses oleh *fully connected layer* dan fungsi aktivasi *Softmax* untuk menghasilkan probabilitas setiap kelas. Selama proses pelatihan, hasil prediksi dibandingkan dengan label sebenarnya menggunakan *loss function*. Nilai *loss* kemudian digunakan pada proses *backpropagation* untuk menghitung gradien setiap parameter, yang selanjutnya diperbarui oleh algoritma optimasi agar kesalahan prediksi semakin kecil pada setiap iterasi pelatihan.

2.5.2 Depthwise Separable Convolution

Komponen utama yang membedakan MobileNet dengan CNN konvensional adalah penggunaan *depthwise separable convolution*. Berbeda dengan *standard convolution* yang melakukan ekstraksi fitur spasial dan penggabungan informasi antar kanal dalam satu operasi, metode ini memisahkan proses tersebut menjadi dua tahap, yaitu *depthwise convolution* dan *pointwise convolution* [2].

Berdasarkan Gambar 2.3, *depthwise convolution* melakukan konvolusi secara independen pada setiap kanal masukan untuk mengekstraksi informasi spasial, sedangkan *pointwise convolution* menggunakan kernel berukuran 1×1 untuk menggabungkan informasi antar kanal. Pemisahan kedua proses tersebut mampu mengurangi jumlah parameter dan kebutuhan komputasi secara signifikan tanpa mengurangi kemampuan ekstraksi fitur.

Kompleksitas komputasi pada *standard convolution* dinyatakan oleh Rumus 2.6

$$C_{SC} = D_K^2 M N D_F^2 \tag{2.6}$$

Pada Rumus 2.6, D_K menyatakan ukuran kernel, M menyatakan jumlah kanal masukan (*input channels*), N menyatakan jumlah kanal keluaran (*output channels*), sedangkan D_F menyatakan ukuran *feature map*. Rumus 2.6 menunjukkan bahwa proses *standard convolution* melakukan ekstraksi fitur dan penggabungan kanal secara bersamaan sehingga membutuhkan jumlah operasi komputasi yang relatif besar.

Sedangkan kompleksitas *depthwise convolution* dinyatakan oleh Rumus 2.7

$$C_{DW} = D_K^2 M D_F^2 \tag{2.7}$$

Rumus 2.7 menunjukkan bahwa *depthwise convolution* hanya melakukan proses konvolusi pada masing-masing kanal masukan secara terpisah sehingga jumlah operasi komputasi menjadi lebih sedikit dibandingkan *standard convolution*. dan kompleksitas *pointwise convolution* dinyatakan oleh Rumus 2.8

$$C_{PW} = MND_F^2 \tag{2.8}$$

Rumus 2.8 menunjukkan bahwa *pointwise convolution* menggunakan kernel berukuran 1×1 untuk menggabungkan informasi dari seluruh kanal sehingga menghasilkan jumlah kanal keluaran yang baru. Dengan demikian, total kompleksitas *depthwise separable convolution* dinyatakan oleh Rumus 2.9.

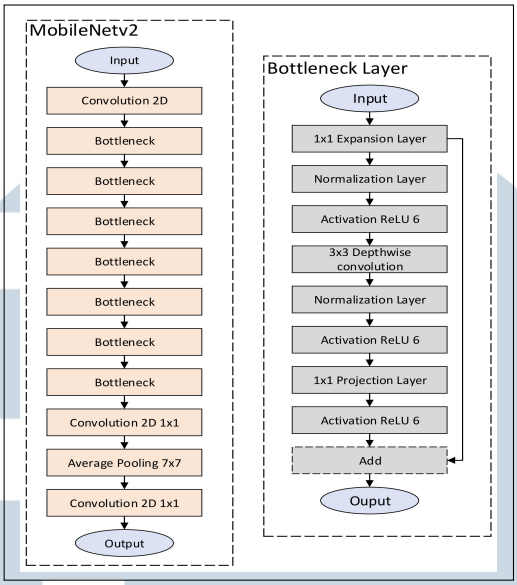
$$C_{DSC} = D_K^2 MD_F^2 + MND_F^2 \tag{2.9}$$

Berdasarkan Rumus 2.9, jumlah operasi komputasi menjadi jauh lebih kecil dibandingkan *standard convolution*. Pada kernel berukuran 3×3, pengurangan komputasi dapat mencapai sekitar 8–9 kali sehingga MobileNet mampu mempertahankan performa klasifikasi dengan model yang lebih ringan dan efisien. Konsep *depthwise separable convolution* menjadi dasar pengembangan MobileNetV2 dan MobileNetV3. Kedua arsitektur tersebut mempertahankan mekanisme ini dengan menambahkan blok khusus untuk meningkatkan representasi fitur dan efisiensi komputasi.

2.5.3 MobileNetV2

MobileNetV2 merupakan pengembangan dari MobileNet yang diperkenalkan oleh Sandler *et al.* untuk meningkatkan kemampuan representasi fitur dengan tetap mempertahankan efisiensi komputasi [3]. Arsitektur ini memperkenalkan dua komponen utama, yaitu *Inverted Residual* dan *Linear Bottleneck*, yang memungkinkan informasi penting tetap dipertahankan selama proses ekstraksi fitur.

Struktur blok *Inverted Residual Bottleneck* pada MobileNetV2 ditunjukkan pada Gambar 2.4.



Gambar 2.4. Blok *Inverted Residual Bottleneck* pada MobileNetV2 [3].

Berdasarkan Gambar 2.4, setiap blok MobileNetV2 terdiri atas empat komponen utama, yaitu *Expansion Layer*, *Depthwise Convolution*, *Linear Bottleneck*, dan *Residual Connection*.

A Mekanisme MobileNetV2

Berdasarkan Gambar 2.4, proses pembelajaran dimulai dengan *Expansion Layer* yang memperbesar jumlah kanal menggunakan konvolusi 1×1. Selanjutnya, *depthwise convolution* mengekstraksi informasi spasial pada setiap kanal secara independen. Hasil ekstraksi kemudian diproyeksikan kembali menggunakan *Linear Bottleneck* untuk mengurangi dimensi kanal tanpa menghilangkan informasi penting. Apabila dimensi masukan dan keluaran sama, diterapkan *Residual Connection* agar informasi dari lapisan sebelumnya dapat diteruskan secara langsung sehingga proses pembelajaran menjadi lebih stabil.

B Expansion Layer

Tahap pertama pada blok MobileNetV2 adalah *Expansion Layer*, yaitu konvolusi 1×1 yang memperbesar jumlah kanal pada *feature map*. Tujuan proses ini adalah meningkatkan kapasitas representasi fitur sebelum dilakukan ekstraksi fitur spasial.

Secara matematis, proses ekspansi dinyatakan oleh Rumus 2.10.

$$X_e = W_e X \tag{2.10}$$

Pada Rumus 2.10, X menyatakan *feature map* masukan, W_e menyatakan bobot konvolusi berukuran 1×1, sedangkan X_e menyatakan hasil proses ekspansi. Rumus tersebut menunjukkan bahwa jumlah kanal pada *feature map* diperluas terlebih dahulu untuk meningkatkan kapasitas representasi fitur sebelum dilakukan proses *depthwise convolution*. Rumus 2.10 menunjukkan bahwa konvolusi 1×1 meningkatkan jumlah kanal tanpa mengubah dimensi spasial *feature map*. Proses ini menghasilkan representasi fitur yang lebih kaya sehingga informasi yang dipelajari sebelum tahap *depthwise convolution* menjadi lebih optimal.

C Depthwise Convolution

Setelah proses ekspansi, keluaran dinormalisasi menggunakan *Batch Normalization* dan diproses dengan fungsi aktivasi ReLU6 sebelum dilakukan *depthwise convolution* berukuran 3×3 (Gambar 2.4). Berbeda dengan konvolusi standar, *depthwise convolution* melakukan ekstraksi fitur spasial pada setiap kanal secara independen sehingga jumlah parameter dan kebutuhan komputasi menjadi lebih kecil.

Operasi *depthwise convolution* pada MobileNetV2 dinyatakan oleh Rumus 2.11.

$$Y_d = DW(X_e) \tag{2.11}$$

Pada Rumus 2.11, DW menyatakan operasi *depthwise convolution*, X_e menyatakan hasil proses ekspansi (*expansion layer*), sedangkan Y_d menyatakan keluaran hasil ekstraksi fitur. Rumus tersebut menunjukkan bahwa setiap kanal pada hasil ekspansi diproses secara independen menggunakan kernel konvolusi sehingga informasi spasial dapat dipelajari secara efisien sebelum diteruskan menuju *Linear Bottleneck*. Rumus 2.11 menunjukkan bahwa setiap kanal diproses secara terpisah sehingga informasi spasial dapat dipelajari secara efisien sebelum diteruskan menuju *Linear Bottleneck*.

D Linear Bottleneck

Tahap berikutnya adalah *Linear Bottleneck*, yaitu konvolusi 1×1 yang memproyeksikan hasil *depthwise convolution* ke dimensi kanal yang lebih kecil. Berbeda dengan lapisan sebelumnya, proses ini tidak menggunakan fungsi aktivasi sehingga informasi penting tetap dipertahankan selama reduksi dimensi.

Secara matematis, proses proyeksi dinyatakan oleh Rumus 2.12.

$$Y = W_p Y_d \tag{2.12}$$

Pada Rumus 2.12, W_p menyatakan bobot konvolusi berukuran 1×1, Y_d menyatakan keluaran dari proses *depthwise convolution*, sedangkan Y menyatakan hasil proyeksi. Rumus 2.12 menunjukkan bahwa konvolusi 1×1 menggabungkan informasi antar kanal sekaligus mereduksi dimensi fitur sehingga representasi menjadi lebih ringkas tanpa kehilangan informasi penting sebelum diteruskan ke *Residual Connection*.

E Residual Connection

Tahap terakhir pada blok MobileNetV2 adalah *Residual Connection*. Berdasarkan Gambar 2.4, apabila dimensi masukan dan keluaran sama, *feature map* masukan dijumlahkan dengan hasil transformasi blok menggunakan operasi *Add*. Mekanisme ini mempertahankan aliran informasi sekaligus memperlancar propagasi gradien pada jaringan yang lebih dalam.

Secara matematis, *Residual Connection* dinyatakan oleh Rumus 2.13.

$$Y = X + F(X) \tag{2.13}$$

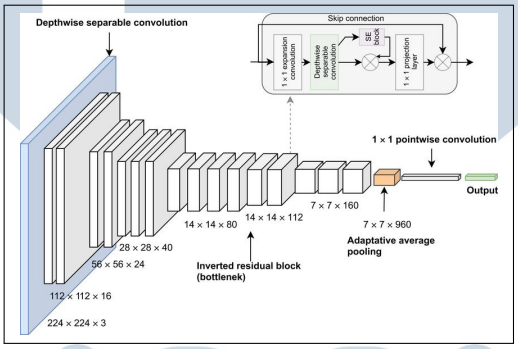
Pada Rumus 2.13, X menyatakan *feature map* masukan, $F(X)$ menyatakan hasil transformasi blok, sedangkan Y menyatakan keluaran akhir. Rumus 2.13 menunjukkan bahwa keluaran diperoleh dari penjumlahan antara masukan dan hasil transformasi sehingga informasi penting tetap dipertahankan serta risiko degradasi performa dapat dikurangi. MobileNetV2 telah banyak diterapkan pada klasifikasi citra *Chest X-Ray*. Penelitian oleh Rifai *et al.* [16] menunjukkan bahwa kombinasi MobileNetV2 dengan *White Balance* dan CLAHE mampu meningkatkan performa

diagnosis pneumonia. Pada penelitian ini, MobileNetV2 digunakan sebagai salah satu model pembanding terhadap CNN *baseline* dan MobileNetV3 untuk mengevaluasi pengaruh arsitektur *transfer learning* terhadap klasifikasi pneumonia.

2.5.4 MobileNetV3

MobileNetV3 merupakan pengembangan lanjutan dari MobileNet yang dirancang untuk meningkatkan akurasi klasifikasi dengan tetap mempertahankan efisiensi komputasi [4]. Arsitektur ini mengombinasikan *depthwise separable convolution*, *Squeeze-and-Excitation (SE) Block*, fungsi aktivasi *Hard-Swish*, serta *Residual Connection* untuk meningkatkan kemampuan representasi fitur tanpa menambah kompleksitas model secara signifikan.

Arsitektur MobileNetV3 ditunjukkan pada Gambar 2.5.



Gambar 2.5. Arsitektur MobileNetV3 [4].

Berdasarkan Gambar 2.5, MobileNetV3 mempertahankan penggunaan *depthwise separable convolution* seperti pada MobileNetV2, kemudian menambahkan *Squeeze-and-Excitation (SE) Block* untuk meningkatkan perhatian terhadap kanal yang lebih penting serta menggunakan fungsi aktivasi *Hard-Swish* guna meningkatkan efisiensi komputasi dan kemampuan representasi fitur.

A Mekanisme MobileNetV3

Berdasarkan Gambar 2.5, citra masukan diproses melalui serangkaian blok *Inverted Residual* yang terdiri atas *Expansion Layer*, *Depthwise Convolution*, *SE Block*, dan *Projection Layer*. Apabila dimensi masukan dan keluaran sama, blok tersebut dihubungkan menggunakan *Residual Connection*. Seiring bertambahnya jumlah blok, ukuran spasial *feature map* semakin kecil sementara jumlah kanal

meningkat sehingga model mampu mempelajari representasi fitur yang lebih kompleks. Setelah proses ekstraksi fitur selesai, keluaran dirangkum menggunakan *Adaptive Average Pooling*, kemudian diproses melalui konvolusi 1×1 dan lapisan klasifikasi untuk menghasilkan probabilitas setiap kelas.

B SE Block

Salah satu perbedaan utama antara MobileNetV2 dan MobileNetV3 adalah penambahan *Squeeze-and-Excitation (SE) Block*. Berdasarkan Gambar 2.5, komponen ini ditempatkan setelah *depthwise convolution* untuk memberikan bobot perhatian (*attention*) pada setiap kanal sehingga model dapat memfokuskan proses ekstraksi fitur pada informasi yang paling relevan.

SE Block terdiri atas dua tahap, yaitu *Squeeze* dan *Excitation*. Pada tahap *Squeeze*, setiap kanal diringkas menggunakan *Global Average Pooling* sehingga diperoleh representasi global setiap kanal.

Secara matematis, proses *Squeeze* dinyatakan oleh Rumus 2.14.

$$z_c = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} X_c(i, j) \tag{2.14}$$

Pada Rumus 2.14, X_c menyatakan *feature map* pada kanal ke-c, H dan W masing-masing menyatakan tinggi dan lebar *feature map*, sedangkan z_c menyatakan representasi global dari kanal tersebut. Rumus 2.14 menunjukkan bahwa setiap kanal diringkas menjadi satu nilai representatif yang menggambarkan informasi global pada kanal tersebut. Representasi ini digunakan sebagai masukan pada tahap *Excitation* untuk menentukan tingkat kepentingan setiap kanal. Selanjutnya dilakukan tahap *Excitation* menggunakan dua lapisan *fully connected* untuk menghasilkan bobot perhatian setiap kanal.

Secara matematis, proses *Excitation* dinyatakan oleh Rumus 2.15.

$$s = \sigma\left(W_2\, \delta\left(W_1 z\right)\right) \tag{2.15}$$

Pada Rumus 2.15, z menyatakan hasil proses *Squeeze*, W_1 dan W_2 menyatakan bobot pada lapisan *fully connected*, $\delta(\cdot)$ menyatakan fungsi aktivasi ReLU, $\sigma(\cdot)$ menyatakan fungsi aktivasi Sigmoid, sedangkan s menyatakan bobot perhatian untuk setiap kanal. Rumus 2.15 menunjukkan bahwa bobot perhatian

digunakan untuk memperkuat kanal yang mengandung informasi penting dan menekan kanal yang kurang relevan. Mekanisme ini membantu MobileNetV3 menghasilkan representasi fitur yang lebih diskriminatif sehingga meningkatkan kualitas proses klasifikasi.

C Hard-Swish

MobileNetV3 memperkenalkan fungsi aktivasi *Hard-Swish* sebagai pengganti ReLU6 pada beberapa lapisan. Fungsi ini merupakan aproksimasi dari *Swish* yang lebih sederhana sehingga mampu meningkatkan akurasi tanpa menambah kompleksitas komputasi secara signifikan.

Secara matematis, *Hard-Swish* dinyatakan oleh Rumus 2.16

$$\text{HardSwish}(x) = x \cdot \frac{\text{ReLU6}(x+3)}{6} \tag{2.16}$$

Pada Rumus 2.16, x menyatakan nilai masukan, $\text{ReLU6}(\cdot)$ menyatakan fungsi aktivasi ReLU6, sedangkan $\text{HardSwish}(x)$ menyatakan nilai keluaran setelah fungsi aktivasi *Hard-Swish* diterapkan. Rumus 2.16 menunjukkan bahwa fungsi ini menghasilkan aktivasi yang lebih halus dibandingkan ReLU sehingga propagasi gradien menjadi lebih stabil dan kemampuan model dalam mempelajari fitur kompleks meningkat.

D Residual Connection

Selain menggunakan *SE Block* dan *Hard-Swish*, MobileNetV3 tetap mempertahankan mekanisme *Residual Connection*. Berdasarkan Gambar 2.5, *skip connection* diterapkan apabila dimensi *feature map* masukan dan keluaran sama sehingga informasi dari lapisan sebelumnya dapat diteruskan secara langsung.

Secara matematis, *Residual Connection* dinyatakan oleh Rumus 2.17

$$Y = X + F(X) \tag{2.17}$$

Pada Rumus 2.17, X menyatakan *feature map* masukan, $F(X)$ menyatakan hasil transformasi blok, sedangkan Y menyatakan keluaran akhir. Rumus 2.17 menunjukkan bahwa mekanisme ini mempertahankan informasi penting,

memperlancar propagasi gradien, serta meningkatkan stabilitas pelatihan pada jaringan yang lebih dalam.

Dalam bidang pencitraan medis, MobileNetV3 telah digunakan untuk berbagai tugas klasifikasi karena mampu memberikan keseimbangan antara akurasi dan efisiensi komputasi [4, 28]. Pada penelitian ini, MobileNetV3 dibandingkan dengan CNN *baseline* dan MobileNetV2 untuk mengevaluasi pengaruh pengembangan arsitektur MobileNet terhadap performa klasifikasi pneumonia pada citra *Chest X-Ray*.

2.6 Explainable Artificial Intelligence

Explainable Artificial Intelligence (XAI) merupakan pendekatan yang digunakan untuk menjelaskan proses pengambilan keputusan pada model kecerdasan buatan sehingga hasil prediksi yang dihasilkan dapat dipahami oleh manusia. Dalam model *deep learning*, khususnya *Convolutional Neural Network* (CNN), proses klasifikasi sering kali bersifat kompleks karena melibatkan jutaan parameter yang dipelajari selama proses pelatihan. Kondisi tersebut menyebabkan model sering dianggap sebagai *black box*, yaitu mampu menghasilkan prediksi dengan akurasi tinggi, namun sulit menjelaskan alasan di balik keputusan yang diambil. Dalam bidang medis, interpretabilitas model menjadi aspek yang sangat penting karena hasil prediksi dapat memengaruhi proses diagnosis dan pengambilan keputusan klinis. Oleh sebab itu, model yang digunakan untuk menganalisis citra medis tidak hanya dituntut memiliki performa klasifikasi yang baik, tetapi juga mampu memberikan penjelasan mengenai bagian citra yang menjadi dasar pengambilan keputusan. Penjelasan tersebut dapat meningkatkan kepercayaan pengguna terhadap sistem kecerdasan buatan serta membantu proses validasi hasil prediksi oleh tenaga medis [17, 18].

Pada citra medis, metode XAI umumnya digunakan untuk menghasilkan visualisasi berupa *heatmap* yang menunjukkan area citra dengan kontribusi terbesar terhadap hasil klasifikasi. Pada citra *Chest X-Ray*, visualisasi tersebut membantu mengidentifikasi apakah model memusatkan perhatian pada area paru-paru yang mengalami indikasi pneumonia atau justru pada bagian lain yang tidak relevan. Berbagai penelitian telah menerapkan metode XAI pada klasifikasi citra *Chest X-Ray* untuk penyakit pneumonia, COVID-19, maupun tuberkulosis [21, 20].

2.6.1 Grad-CAM dan Grad-CAM++

Gradient-weighted Class Activation Mapping (Grad-CAM) merupakan salah satu metode XAI yang digunakan untuk memvisualisasikan bagian citra yang paling berpengaruh terhadap prediksi model *deep learning*. Metode ini memanfaatkan gradien dari kelas target terhadap *feature map* pada lapisan konvolusi terakhir untuk menentukan tingkat kepentingan setiap kanal. Hasil visualisasi ditampilkan dalam bentuk *heatmap* yang menunjukkan area citra dengan kontribusi terbesar terhadap keputusan model.

Bobot setiap *feature map* dihitung menggunakan rata-rata gradien terhadap kelas target yang dinyatakan oleh Rumus 2.18.

$$\alpha_k^c = \frac{1}{Z}\sum_i\sum_j \frac{\partial y^c}{\partial A_{ij}^k} \tag{2.18}$$

Pada Rumus 2.18, A^k menyatakan *feature map* ke-k, y^c menyatakan skor kelas target, Z menyatakan jumlah piksel pada *feature map*, sedangkan α_k^c menyatakan bobot kontribusi kanal ke-k. Rumus 2.18 menunjukkan bahwa bobot setiap kanal ditentukan berdasarkan rata-rata gradien terhadap kelas target. Semakin besar nilai gradien, semakin besar kontribusi kanal tersebut dalam menghasilkan prediksi.

Selanjutnya, *heatmap* Grad-CAM dihitung menggunakan Rumus 2.19.

$$L_{Grad-CAM}^c = ReLU\left(\sum_k \alpha_k^c A^k\right) \tag{2.19}$$

Rumus 2.19 menunjukkan bahwa setiap *feature map* dikalikan dengan bobotnya masing-masing, kemudian dijumlahkan dan diproses menggunakan fungsi aktivasi ReLU. Hasil akhirnya berupa *heatmap* yang menunjukkan area citra yang memberikan kontribusi terbesar terhadap prediksi model.

Grad-CAM++ merupakan pengembangan dari Grad-CAM yang dirancang untuk menghasilkan visualisasi yang lebih akurat, terutama ketika terdapat lebih dari satu area penting pada citra. Berbeda dengan Grad-CAM yang menggunakan rata-rata gradien sebagai bobot setiap kanal, Grad-CAM++ memanfaatkan turunan orde kedua dan ketiga sehingga pembobotan setiap piksel menjadi lebih presisi.

Bobot pada Grad-CAM++ dihitung menggunakan Rumus 2.20.

$$\alpha_{ij}^{kc} = \frac{\frac{\partial^2 y^c}{\left(\partial A_{ij}^k\right)^2}}{2\frac{\partial^2 y^c}{\left(\partial A_{ij}^k\right)^2} + \sum_{a,b} A_{ab}^k \frac{\partial^3 y^c}{\left(\partial A_{ij}^k\right)^3}} \tag{2.20}$$

Rumus 2.20 menunjukkan bahwa bobot dihitung menggunakan turunan orde kedua dan ketiga sehingga kontribusi setiap piksel dapat direpresentasikan secara lebih rinci dibandingkan Grad-CAM.

Selanjutnya, *heatmap* Grad-CAM++ dihitung menggunakan Rumus 2.21.

$$L_{Grad-CAM++}^c = ReLU\left(\sum_k w_k^c A^k\right) \tag{2.21}$$

Pada Rumus 2.21, w_k^c menyatakan bobot kanal hasil perhitungan Grad-CAM++, A^k menyatakan *feature map* ke-k, sedangkan $L_{Grad-CAM++}^c$ menyatakan *heatmap* hasil visualisasi. Rumus 2.21 menunjukkan bahwa *heatmap* dibentuk melalui penjumlahan *feature map* yang telah diberi bobot, kemudian diproses menggunakan fungsi ReLU untuk menampilkan area yang memberikan kontribusi positif terhadap prediksi model. Dibandingkan Grad-CAM, Grad-CAM++ menghasilkan lokalisasi yang lebih akurat, terutama ketika objek tersebar pada beberapa wilayah citra. Pada penelitian ini, Grad-CAM dan Grad-CAM++ digunakan untuk memvisualisasikan area paru-paru yang menjadi perhatian model ketika mengklasifikasikan citra *Chest X-Ray*. Hasil visualisasi digunakan untuk mengevaluasi apakah model memfokuskan perhatian pada wilayah yang relevan dengan indikasi pneumonia sehingga proses pengambilan keputusan model menjadi lebih mudah diinterpretasikan [22, 29, 30].

2.7 Evaluasi Model Klasifikasi

Evaluasi model klasifikasi dilakukan untuk mengukur kemampuan model dalam memprediksi kelas dengan benar. Pada penelitian deteksi pneumonia, evaluasi model sangat penting karena kesalahan prediksi dapat berdampak pada interpretasi kondisi citra medis. Model yang baik tidak hanya dinilai berdasarkan akurasi, tetapi juga berdasarkan kemampuannya dalam membedakan kelas normal dan pneumonia secara seimbang. Beberapa metrik evaluasi yang umum digunakan dalam klasifikasi citra medis antara lain *confusion matrix*, *accuracy*, *precision*, *recall*, *F1-score*, ROC Curve, dan AUC. Metrik-metrik tersebut digunakan dalam

berbagai penelitian deteksi pneumonia berbasis *deep learning* untuk menilai performa model secara menyeluruh [10, 24, 31].

2.7.1 Confusion Matrix

Evaluasi model klasifikasi tidak hanya dilakukan melalui nilai akurasi, tetapi juga dapat dianalisis menggunakan *confusion matrix*. Metode ini menyajikan ringkasan hasil prediksi model dalam bentuk matriks sehingga kesesuaian antara prediksi dan label sebenarnya dapat diamati secara sistematis. Untuk kasus klasifikasi biner, informasi yang ditampilkan mencakup empat kemungkinan hasil, yaitu *True Positive* (TP), *True Negative* (TN), *False Positive* (FP), dan *False Negative* (FN).

True Positive (TP) menunjukkan jumlah data yang termasuk dalam kelas pneumonia dan berhasil diprediksi dengan benar sebagai pneumonia oleh model. Sebaliknya, *True Negative* (TN) menunjukkan jumlah data yang termasuk dalam kelas normal dan berhasil diprediksi dengan benar sebagai normal. Adapun *False Positive* (FP) merupakan jumlah data yang sebenarnya termasuk dalam kelas normal, tetapi diprediksi sebagai pneumonia oleh model. Sementara itu, *False Negative* (FN) menunjukkan jumlah data yang sebenarnya termasuk dalam kelas pneumonia, tetapi diprediksi sebagai normal oleh model. Dalam konteks klasifikasi pneumonia, nilai FN menjadi perhatian penting karena menunjukkan kasus pneumonia yang tidak terdeteksi oleh model. Oleh karena itu, *confusion matrix* diperlukan untuk melihat jenis kesalahan yang dilakukan model, bukan hanya melihat nilai akurasi secara keseluruhan.

2.7.2 Accuracy, Precision, Recall, dan F1-Score

Accuracy merupakan metrik yang menunjukkan proporsi prediksi benar terhadap seluruh data. Perhitungan *accuracy* dinyatakan oleh Rumus 2.22.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.22}$$

Rumus 2.22 menunjukkan bahwa nilai *accuracy* diperoleh dari perbandingan jumlah prediksi yang benar terhadap seluruh data yang diuji. Semakin tinggi nilai *accuracy*, semakin baik kemampuan model dalam melakukan klasifikasi secara keseluruhan.

Precision menunjukkan seberapa banyak prediksi positif yang benar-benar merupakan kelas positif. Dalam deteksi pneumonia, *precision* menunjukkan ketepatan model ketika memprediksi suatu citra sebagai pneumonia. Perhitungan *Precision* dinyatakan oleh Rumus 2.23.

$$Precision = \frac{TP}{TP + FP} \tag{2.23}$$

Rumus 2.23 menunjukkan bahwa semakin tinggi nilai *precision*, semakin sedikit prediksi positif yang salah. Metrik ini penting untuk mengurangi kesalahan diagnosis berupa *false positive*.

Recall menunjukkan seberapa banyak data positif yang berhasil ditemukan oleh model. Dalam kasus pneumonia, metrik ini menggambarkan kemampuan model mendeteksi citra yang benar-benar pneumonia. Perhitungan *Recall* dinyatakan oleh Rumus 2.24.

$$Recall = \frac{TP}{TP + FN} \tag{2.24}$$

Rumus 2.24 menunjukkan bahwa nilai *recall* akan semakin tinggi apabila jumlah *false negative* semakin kecil. Pada deteksi pneumonia, nilai *recall* yang tinggi penting untuk meminimalkan kasus pneumonia yang tidak terdeteksi.

F1-score merupakan rata-rata harmonik dari *precision* dan *recall*. Perhitungan *F1-score* dinyatakan oleh Rumus 2.25.

$$F1\text{-}Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{2.25}$$

Rumus 2.25 menunjukkan bahwa nilai *F1-score* memberikan keseimbangan antara *precision* dan *recall* dalam mengevaluasi performa model. Metrik ini banyak digunakan ketika distribusi kelas pada *dataset* tidak seimbang.

Penggunaan metrik *accuracy*, *precision*, *recall*, dan *F1-score* banyak ditemukan dalam penelitian klasifikasi pneumonia berbasis citra *Chest X-Ray*. Metrik tersebut membantu mengevaluasi performa model dari berbagai sudut pandang, terutama ketika *dataset* memiliki distribusi kelas yang tidak seimbang [32, 10, 33].

2.7.3 ROC Curve dan AUC

ROC Curve (*Receiver Operating Characteristic Curve*) merupakan representasi grafis yang menggambarkan hubungan antara *True Positive Rate* (TPR) dan *False Positive Rate* (FPR) pada berbagai nilai ambang (*threshold*) klasifikasi. Kurva ini digunakan sebagai salah satu metode evaluasi untuk menilai kemampuan model dalam membedakan kelas positif dan kelas negatif. Semakin baik performa model, semakin dekat kurva ROC menuju sudut kiri atas grafik, yang menunjukkan tingkat deteksi positif yang tinggi dengan tingkat kesalahan positif yang rendah.

True Positive Rate (TPR) sama dengan *recall* dan perhitungan *True Positive Rate* dinyatakan oleh Rumus 2.26.

$$TPR = \frac{TP}{TP + FN} \tag{2.26}$$

Rumus 2.26 menunjukkan bahwa semakin tinggi nilai TPR maka semakin banyak data positif yang berhasil dikenali oleh model.

False Positive Rate (FPR) menunjukkan proporsi data negatif yang salah diprediksi sebagai positif. Perhitungan *False Positive Rate* dinyatakan oleh Rumus 2.27.

$$FPR = \frac{FP}{FP + TN} \tag{2.27}$$

Rumus 2.27 menunjukkan bahwa semakin kecil nilai FPR maka semakin sedikit data negatif yang salah diklasifikasikan sebagai positif.

AUC (*Area Under the Curve*) merupakan ukuran yang menyatakan luas area di bawah kurva ROC (*Receiver Operating Characteristic*). Nilai AUC digunakan untuk menilai kemampuan model dalam membedakan antara kelas yang berbeda. Semakin tinggi nilai AUC atau semakin mendekati nilai 1, semakin baik kemampuan model dalam melakukan diskriminasi antar kelas. Dalam konteks deteksi pneumonia, kurva ROC dan nilai AUC berperan sebagai metrik evaluasi untuk mengukur kemampuan model dalam membedakan citra *NORMAL* dan *PNEUMONIA* pada berbagai nilai *threshold*. Oleh karena itu, metrik ini sering digunakan sebagai pelengkap *accuracy*, *precision*, *recall*, dan *F1-score* dalam mengevaluasi performa model klasifikasi [10, 31, 24].

2.8 Optimasi Model Deep Learning

Optimasi model *deep learning* merupakan proses pembelajaran yang bertujuan meminimalkan nilai *loss* melalui pembaruan parameter model secara iteratif sehingga kemampuan prediksi dan generalisasi model terhadap data baru dapat ditingkatkan. Selama proses pelatihan, bobot (*weights*) dan bias diperbarui berdasarkan informasi yang diperoleh dari data latih sehingga model mampu mempelajari representasi fitur yang semakin baik.

Secara umum, proses optimasi melibatkan tiga tahapan utama, yaitu *forward propagation*, perhitungan nilai *loss*, dan *backpropagation*. Selanjutnya, parameter model diperbarui menggunakan algoritma optimasi hingga model mencapai kondisi konvergen. Pada penelitian ini, proses optimasi menggunakan *Cross Entropy Loss*, *Adam Optimizer*, *ReduceLROnPlateau*, serta *Early Stopping* untuk mendukung proses pelatihan model.

2.8.1 Forward Propagation dan Backpropagation

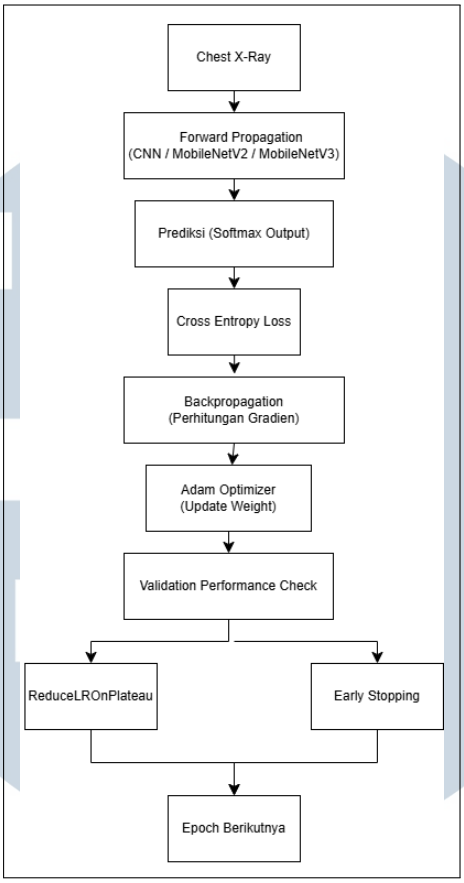
Proses pelatihan model *deep learning* terdiri atas dua tahapan utama, yaitu *forward propagation* dan *backpropagation*. Kedua proses tersebut dilakukan secara berulang pada setiap *epoch* hingga model mencapai kondisi konvergen. Pada tahap *forward propagation*, citra masukan diproses melalui seluruh lapisan jaringan sesuai arsitektur yang digunakan, yaitu CNN *baseline*, MobileNetV2, atau MobileNetV3, hingga menghasilkan probabilitas setiap kelas menggunakan fungsi aktivasi *Softmax*.

Secara matematis, proses tersebut dapat dinyatakan oleh Rumus 2.28

$$\hat{y} = f(x, \theta) \tag{2.28}$$

Pada Rumus 2.28, x menyatakan citra masukan (*input image*), θ menyatakan parameter model yang terdiri atas bobot (*weights*) dan bias, sedangkan $\hat{y}$ menyatakan hasil prediksi model. Rumus 2.28 menunjukkan bahwa fungsi $f(\cdot)$ merepresentasikan seluruh proses komputasi pada jaringan, mulai dari ekstraksi fitur hingga menghasilkan prediksi.

Proses pelatihan model *deep learning* melalui tahapan *forward propagation* dan *backpropagation* ditunjukkan pada Gambar 2.6.



Gambar 2.6. Flowchart proses pelatihan model *deep learning*.

Berdasarkan Gambar 2.6, hasil prediksi dari proses *forward propagation* dibandingkan dengan label sebenarnya untuk menghitung nilai *loss*. Selanjutnya dilakukan proses *backpropagation*, yaitu menghitung gradien dari fungsi *loss* terhadap setiap parameter model menggunakan aturan rantai (*chain rule*). Gradien tersebut menjadi dasar bagi algoritma optimasi untuk memperbarui bobot model sehingga nilai *loss* semakin kecil pada iterasi berikutnya.

Secara matematis, gradien pada proses *backpropagation* dinyatakan oleh Rumus 2.29

$$\frac{\partial L}{\partial \theta} \tag{2.29}$$

Pada Rumus 2.29, L menyatakan nilai *loss*, sedangkan θ menyatakan parameter model. Rumus 2.29 menunjukkan bahwa setiap parameter diperbarui berdasarkan besar gradien yang diperoleh sehingga model mampu menghasilkan prediksi yang semakin mendekati label sebenarnya pada setiap *epoch*. Pada

penelitian ini, pembaruan parameter dilakukan menggunakan *Adam Optimizer*, sedangkan nilai *loss* dihitung menggunakan *Cross Entropy Loss*.

2.8.2 Loss Function dan Optimizer

Pada proses pelatihan model *deep learning*, *loss function* digunakan untuk mengukur selisih antara hasil prediksi model dan label sebenarnya. Nilai *loss* menjadi dasar dalam proses *backpropagation* untuk memperbarui parameter model. Semakin kecil nilai *loss*, semakin baik kemampuan model dalam melakukan klasifikasi. Pada penelitian ini digunakan *Cross Entropy Loss* karena permasalahan yang diselesaikan merupakan klasifikasi dua kelas, yaitu *NORMAL* dan *PNEUMONIA*, dengan keluaran berupa probabilitas hasil fungsi aktivasi *Softmax*.

Cross Entropy Loss dinyatakan oleh Rumus 2.30

$$L = -\sum_{i=1}^{C} y_i \log(\hat{y}_i) \tag{2.30}$$

Pada Rumus 2.30, C menyatakan jumlah kelas, y_i menyatakan label sebenarnya (*ground truth*), sedangkan $\hat{y}_i$ menyatakan probabilitas hasil prediksi model untuk kelas ke-i. Rumus 2.30 menunjukkan bahwa nilai *loss* akan semakin kecil apabila probabilitas prediksi pada kelas yang benar semakin tinggi. Nilai tersebut kemudian digunakan untuk menghitung gradien pada proses *backpropagation*. Parameter model selanjutnya diperbarui menggunakan *Adam Optimizer*. Adam mengombinasikan konsep *Momentum* dan *RMSProp* sehingga mampu menghasilkan proses konvergensi yang lebih cepat dan stabil.

Estimasi momen pertama dihitung menggunakan Rumus 2.31.

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t \tag{2.31}$$

Selanjutnya, estimasi momen kedua dihitung menggunakan Rumus 2.32.

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \tag{2.32}$$

Parameter model kemudian diperbarui menggunakan Rumus 2.33.

$$\theta_{t+1} = \theta_t - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon} \tag{2.33}$$

Pada Rumus 2.33, g_t menyatakan gradien pada iterasi ke-t, m_t menyatakan estimasi momen pertama (*first moment*), v_t menyatakan estimasi momen kedua (*second moment*), α menyatakan *learning rate*, sedangkan θ menyatakan parameter model. Berdasarkan Rumus 2.31 hingga Rumus 2.33, Adam menghitung rata-rata gradien dan kuadrat gradien untuk menyesuaikan besar pembaruan setiap parameter secara adaptif, sehingga proses optimasi menjadi lebih stabil dan konvergen lebih cepat.

2.8.3 Learning Rate Scheduler dan Early Stopping

Selain *loss function* dan *optimizer*, proses pelatihan juga dipengaruhi oleh pengaturan *learning rate*. Pada penelitian ini digunakan *ReduceLROnPlateau* sebagai *learning rate scheduler* dan *Early Stopping* untuk menjaga stabilitas proses pelatihan. *ReduceLROnPlateau* memantau nilai *validation loss*. Apabila tidak terjadi peningkatan performa selama sejumlah *epoch*, maka nilai *learning rate* akan dikurangi sehingga proses optimasi dapat mencapai konvergensi yang lebih baik.

Perubahan nilai *learning rate* dinyatakan oleh Rumus 2.34.

$$lr_{baru} = lr_{lama} \times factor \tag{2.34}$$

Rumus 2.34 menunjukkan bahwa nilai *learning rate* dikurangi menggunakan faktor pengali (*factor*) ketika performa validasi tidak mengalami peningkatan. Pada penelitian ini digunakan nilai *factor* sebesar 0,5 dan *patience* selama 2 *epoch*.

Selain itu, *Early Stopping* memantau nilai *validation loss* pada setiap *epoch*. Apabila tidak ditemukan peningkatan performa setelah sejumlah *epoch* sesuai nilai *patience*, proses pelatihan dihentikan secara otomatis. Mekanisme ini mencegah model terus belajar dari data pelatihan ketika kemampuan generalisasinya tidak lagi meningkat sehingga risiko *overfitting* dapat dikurangi. Pada penelitian ini, *ReduceLROnPlateau* dan *Early Stopping* diterapkan secara bersamaan untuk menjaga stabilitas proses pelatihan dan memperoleh model dengan performa yang lebih optimal.

2.9 Penelitian Terkait

Penelitian mengenai klasifikasi pneumonia menggunakan citra *Chest X-Ray* telah banyak dilakukan menggunakan berbagai arsitektur *deep learning*, seperti CNN, *transfer learning*, MobileNet, serta metode *Explainable Artificial Intelligence* (XAI). Beberapa penelitian yang menjadi dasar dan acuan dalam penelitian ini dirangkum pada Tabel 2.1.

Tabel 2.1. Penelitian terkait

Peneliti	**Dataset**	**Metode**	**Hasil Penelitian**
Jiang *et al.* (2021) [12]	Chest X-Ray Images (Pneumonia)	Improved VGG16	Modifikasi arsitektur VGG16 meningkatkan akurasi klasifikasi pneumonia dibandingkan VGG16 standar.
Salehi *et al.* (2021) [23]	Chest X-Ray Images (Pneumonia)	Transfer Learning	Pendekatan *transfer learning* meningkatkan performa klasifikasi pneumonia dibandingkan pelatihan model dari awal.
Manickam *et al.* (2021) [24]	Chest X-Ray Images (Pneumonia)	Transfer Learning	Pemilihan arsitektur *transfer learning* dan optimizer berpengaruh terhadap performa klasifikasi pneumonia.
Al Reshan *et al.* (2023) [15]	Chest X-Ray Images (Pneumonia) dan NIH ChestX-ray14	MobileNet	MobileNet menghasilkan model yang ringan dengan jumlah parameter lebih sedikit tanpa mengurangi performa klasifikasi.
Shamrat *et al.* (2023) [27]	Chest X-Ray Images (Pneumonia)	Customized MobileNetV2	Customized MobileNetV2 memberikan akurasi tinggi dengan kompleksitas komputasi yang rendah pada klasifikasi penyakit paru.
Rifai *et al.* (2024) [16]	Chest X-Ray Images	MobileNetV2 + CLAHE	Penerapan *White Balance* dan CLAHE meningkatkan performa MobileNetV2 dalam diagnosis pneumonia.
Howard *et al.* (2019) [4]	ImageNet	MobileNetV3	MobileNetV3 meningkatkan efisiensi model melalui *Squeeze-and-Excitation Block* dan fungsi aktivasi *Hard-Swish*.

Lanjut pada halaman berikutnya

Tabel 2.1 Penelitian terkait (lanjutan)

Peneliti	**Dataset**	**Metode**	**Hasil Penelitian**
Bhandari *et al.* (2022) [21]	COVID-19 Radiography Database	CNN + Grad-CAM	Grad-CAM meningkatkan interpretabilitas model dengan memvisualisasikan area citra yang berkontribusi terhadap hasil prediksi.
Colin dan Surantha (2025) [22]	Chest X-Ray Images (Pneumonia)	ResNet50 + XAI	Layer-wise Relevance Propagation (LRP) memberikan interpretabilitas terbaik, sedangkan Grad-CAM membantu memvisualisasikan area yang menjadi dasar keputusan model.

Berdasarkan Tabel 2.1, sebagian besar penelitian terdahulu memanfaatkan dataset *Chest X-Ray Images (Pneumonia)* yang dipublikasikan oleh Paul Mooney melalui platform Kaggle sebagai acuan dalam pengembangan model klasifikasi pneumonia. Berbagai arsitektur, seperti CNN, *transfer learning*, MobileNet, dan MobileNetV2, telah menunjukkan performa yang baik dalam mendeteksi pneumonia pada citra *Chest X-Ray*. Di sisi lain, penelitian mengenai interpretabilitas model umumnya menerapkan metode Grad-CAM atau Grad-CAM++ pada satu arsitektur tertentu untuk menjelaskan proses pengambilan keputusan model. Meskipun demikian, penelitian yang membandingkan CNN *baseline*, MobileNetV2, dan MobileNetV3 secara langsung menggunakan dataset yang sama, konfigurasi pelatihan yang seragam, serta mengevaluasi interpretabilitas menggunakan Grad-CAM dan Grad-CAM++ masih relatif terbatas. Oleh karena itu, penelitian ini dilakukan untuk membandingkan performa ketiga arsitektur tersebut sekaligus mengevaluasi interpretabilitas hasil prediksi dalam satu kerangka eksperimen yang sama.