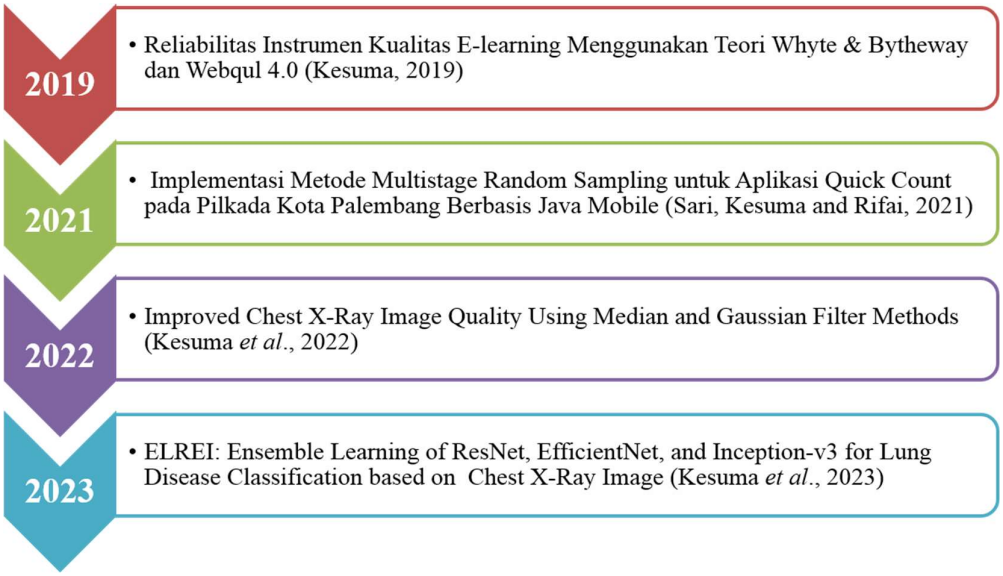
BAB II
TINJAUAN PUSTAKA

2.1. *Roadmap* Penelitian

Roadmap penelitian ini disajikan pada Gambar 2.1. Pada tahun 2019, dilakukan pengujian reliabilitas variabel evaluasi menggunakan kerangka kerja *Whyte & Bytheway* dan *WebQual* 4.0 untuk mengevaluasi kualitas *e-learning*. Pada tahun 2021, dilakukan pembuatan sistem kontrol melalui teknologi biometrik sidik jari dan pengembangan sistem *quick count* menggunakan Metode *Multistage Random Sampling*. Pada tahun 2022, dilakukan penelitian mengenai peningkatan kualitas Citra *Chest X-ray* dengan menggunakan Metode *Median* dan *Gaussian Filter*. Sementara untuk tahun 2023, berfokus pada penelitian ELREI: *Ensemble Learning* dari *ResNet*, *EfficientNet*, dan *Inception-v3* untuk klasifikasi penyakit paru-paru berdasarkan citra *Chest X-Ray*.



Gambar 2.1. *Roadmap* Penelitian Tahun 2019-2023

2.2. Penyakit Menular

Penyakit menular merupakan penyakit yang dapat menular kepada manusia baik secara langsung maupun melalui media yang disebabkan oleh mikroorganisme virus, bakteri, jamur, dan parasit [44]. Saat ini, salah satu penyakit menular yang

menjadi sebuah pandemi di berbagai negara adalah *Coronavirus Disease-2019* (*COVID-19*). *COVID-19* merupakan suatu penyakit menular akut yang menyerang sistem pernapasan secara berkala. Penyakit ini pertama kali dilaporkan di kota Wuhan, Provinsi Hubei, China pada bulan Desember 2019 yang disebabkan oleh virus corona varian baru yaitu *Severe Acute Respiratory Syndrome Coronavirus* 2 (*SARS-CoV-2*), kemudian menyebar secara gobal dan dinyatakan sebagai pandemi *COVID-19*oleh *World Health Organization* (WHO) [45]. Pada umumnya, gejala yang dirasakan penderita *COVID-19* yaitu demam, sakit tenggorokan, batuk kering, sakit kepala, kelelahan, serta kehilangan indra penciuman dan perasa. Adapun gejala berat yang dirasakan seperti sesak napas, hilangnya kemampuan bergerak dan berbicara, hingga *pneumonia* di paru-paru [46], [47]. Gejala ini akan muncul sekitar 5 hingga 6 hari setelah seseorang terpapar virus *SARS-CoV-2* [48].

Untuk mendeteksi penyakit *COVID-19* dapat dilakukan melalui tes *Reverse Transcription Polymerase Chain Reaction* (RT-PCR) dan tes antibodi [2]. Namun, kedua tes tersebut membutuhkan biaya yang cukup mahal dan tenaga medis yang terbatas [3]. Alternatif lainnya yang lebih baik ditemukan para peneliti yaitu melalui *Chest X-Ray* (CXR) paru-paru. Gambar hasil pemeriksaan CXR paru-paru dinilai cukup efektif untuk mendiagnosa penyakit *COVID-19*, karena tenaga medis dapat melihat kelainan pada paru-paru penderita [49].

2.3. Citra Digital

Citra digital merupakan representasi diskrit dari citra yang berisi data dalam bentuk *array* (larik) dengan elemen-elemen yang mewakili intensitas atau tingkat keabuan di setiap koordinat spasial (x, y) dalam citra. Saat sebuah citra direpresentasikan secara digital, intensitas piksel di setiap titik dapat diwakili oleh nilai-nilai bilangan real atau kompleks [50]. Data citra digital dapat diproses dan dianalisis menggunakan komputer dengan metode-metode pengolahan citra digital.

Pengolahan citra digital adalah pemrosesan citra dengan bantuan komputer, dimana *input* dan *output* nya berupa citra atau karakteristik dari suatu gambar yang terkait. Secara umum, pengolahan citra digital terdiri dari beberapa tahapan yaitu peningkatan kualitas citra, segmentasi citra, dan klasifikasi citra [51]. Dalam era

teknologi informasi dan komunikasi saat ini, pengolahan citra digital menjadi bidang penting yang terus berkembang dan memiliki dampak yang besar pada kehidupan saat ini.

Citra yang umum digunakan dalam pengolahan citra digital adalah citra RGB (*Red*, *Green*, *Blue*). Citra RGB ini direpresentasikan dalam bentuk matriks dengan elemen-elemennya yaitu x_1 sebagai intensitas warna merah, x_2 sebagai intensitas warna hijau, dan x_3 sebagai intensitas warna biru. Masing-masing dari intensitas warna tersebut akan mewakili 8 bit di setiap pikselnya. Dalam kombinasi ketiganya, citra ini membentuk tampilan warna lengkap yang dapat ditampilkan dan diproses oleh bantuan komputer dalam proses pengolahan citra [52].

2.4. Citra *Chest X-ray*

Citra *Chest X-ray* (CXR) adalah gambar radiografi dari dada seseorang yang diambil dengan menggunakan sinar-X. Prosedur ini merupakan salah satu metode diagnostik medis paling umum untuk melihat struktur dalam dada, seperti jantung, paru-paru, tulang rusuk, dan organ lainnya. Citra CXR dapat membantu dokter radiologi dalam menganalisis dan mendiagnosa berbagai masalah kesehatan, seperti infeksi paru-paru, pneumonia, dan COVID-19 [4].

Citra CXR merupakan alat diagnostik yang cepat dan relatif murah, yang memungkinkan tenaga medis untuk mendapatkan gambaran awal tentang kondisi dada pasien. Namun, interpretasi citra CXR harus dilakukan oleh dokter spesialis atau ahli radiologi yang terlatih dalam membaca dan menganalisis hasil citra tersebut [5]. Namun citra CXR memiliki keterbatasan dalam diagnosis yang lebih mendalam, sehingga diperlukan pemeriksaan lebih lanjut secara otomatis dengan bantuan komputer.

2.5. Perbaikan Kualitas Citra

Perbaikan kualitas citra merupakan tahapan awal yang dilakukan untuk memperoleh kualitas citra yang lebih baik sehingga mempermudah proses pengklasifikasian citra. Beberapa citra medis sering memiliki kontras yang rendah dan *noise* karena berbagai sumber interferensi, seperti proses pencitraan dan

akuisisi data, sehingga citra sulit dianalisis secara visual [53]. Untuk itu, tahapan perbaikan kualitas citra ini perlu dilakukan seperti menghilangkan bagian yang tidak penting atau *noise*, meningkatkan kontras yang gelap, dan mempertahankan semua detail dari citra [54]. Beberapa teknik yang dilakukan pada tahapan ini sebagai berikut:

2.5.1. Morfologi *Opening*

Morfologi *Opening* merupakan teknik peningkatan kualitas citra dalam memperbaiki kontur citra dan menghilangkan objek tipis yang berada di sekitar area citra sehingga citra terlihat lebih jelas [7]. Metode ini bekerja dengan menggunakan operasi erosi yang dikombinasikan dengan operasi dilatasi. Morfologi pembukaan dapat dihitung dengan menggunakan Persamaan (2.1).

$$A \circ B = (A \ominus B) \oplus B \tag{2.1}$$

dengan

$$A \oplus B = \{x|(B)_x \cap A \neq \emptyset\}$$

$$A \ominus B = \{x|(B)_x \cap A^c \neq \emptyset\}$$

dimana, A merupakan citra awal, B merupakan structuring elemen berbentuk matriks operator seperti *line shape*, *disk*, *diamond*, dan lainnya. $\oplus$ merupakan operasi dilasi, $\ominus$ adalah operasi erosi, dan $\circ$ adalah operasi Morfologi *Opening*.

2.5.2. *Median Filter*

Median Filter adalah fungsi nonlinier yang digunakan untuk pengurangan *noise* dan perataan gambar. *Median Filter* bekerja dengan mengurutkan nilai intensitas dari sekumpulan piksel secara menaik, kemudian mengubah nilai piksel yang diproses ke nilai tertentu [55]. Perhitungan *Median Filter* dapat menggunakan Persamaan (2.2).

$$MF\ (x,y) = Median\ (piksel\ sekitar(x,y)) \tag{2.2}$$

dimana, $MF\ (x, y)$ merupakan nilai piksel hasil proses *Median Filter* pada titik (x, y) dalam citra, sedangkan *Median* merupakan fungsi yang menghitung nilai *Median* dari sekumpulan piksel yang berada di sekitar titik (x, y).

2.5.3. *Mean Square Error*

Mean Square Error (MSE) adalah salah satu metode yang umum digunakan untuk mengukur tingkat kesalahan atau perbedaan antara citra asli dengan citra hasil yang telah diproses atau ditingkatkan kualitasnya. Metode ini banyak digunakan dalam bidang pengolahan citra dan evaluasi kualitas gambar. MSE bekerja dengan cara menghitung selisih kuadrat antara setiap piksel pada citra asli (sebelum ditingkatkan) dan citra hasil pemrosesan (setelah ditingkatkan), kemudian mengambil nilai rata-rata dari seluruh selisih kuadrat ini. Semakin kecil nilai MSE, semakin baik kualitas citra hasil pemrosesan, karena artinya selisih piksel antara citra asli dan hasil pemrosesan juga semakin kecil [56]. Perhitungan nilai MSE dapat dilihat pada Persamaan (2.3).

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(I(i) - K(i))^2 \tag{2.3}$$

dimana, N adalah jumlah piksel dalam citra, $I(i)$ adalah nilai piksel citra asli pada titik ke-i, $K(i)$ adalah nilai piksel hasil peningkatan kualitas citra pada titik ke-i.

2.5.4. *Peak Signal-to-Noise Ratio*

Peak Signal-to-Noise Ratio (PSNR) adalah metode yang digunakan untuk mengukur kualitas citra dengan membandingkan citra asli dengan citra hasil pemrosesan. PSNR biasanya digunakan untuk mengevaluasi sejauh mana kualitas citra yang dihasilkan mendekati citra asli. PSNR mengukur seberapa besar perbedaan antara citra asli dengan citra hasil pemrosesan dengan menggunakan rasio antara kekuatan sinyal puncak (*peak signal*) dan tingkat *noise* (*noise ratio*). PSNR biasanya diukur dalam satuan desibel (dB). Semakin tinggi nilai PSNR, maka semakin mirip citra hasil pemrosesan dengan citra aslinya sehingga kualitas

citra tersebut dianggap semakin baik. PSNR sekitar 30 dB dianggap sebagai nilai ambang batas yang baik untuk citra digital [10]. Perhitungan nilai PSNR dapat dilihat pada Persamaan (2.4).

$$PSNR = 10\, log_{10}\left(\frac{Max^2}{MSE}\right) \tag{2.4}$$

dimana Max adalah nilai maksimum piksel dalam citra (misalnya, 255 untuk citra 8-bit dengan rentang 0-255), MSE adalah *mean square error*.

2.5.5. *Structural Similarity Index Metrics*

Structural Similarity Index Metrics (SSIM) adalah metrik yang digunakan dalam evaluasi peningkatan kualitas citra atau perbandingan antara dua citra. SSIM biasanya digunakan untuk mengukur sejauh mana citra yang telah ditingkatkan kualitasnya memiliki kesamaan struktural dengan citra asli. SSIM bekerja dengan membandingkan tiga aspek utama dari dua citra yang ingin dievaluasi yaitu kecerahan (*brightness*), kontras (*contrast*), dan struktur (*structure*). Dengan mempertimbangkan ketiga elemen ini, SSIM dapat memberikan nilai yang mengindikasikan sejauh mana citra hasil pemrosesan mendekati citra asli. SSIM menghasilkan nilai antara -1 dan 1. Nilai 1 berarti citra hasil pemrosesan identik dengan citra asli, sementara nilai mendekati -1 menunjukkan perbedaan yang sangat signifikan antara kedua citra. Semakin tinggi nilai SSIM, semakin mirip citra hasil pemrosesan dengan citra asli, dan ini menunjukkan peningkatan kualitas citra [57], [58]. Perhitungan nilai SSIM dapat dilihat pada Persamaan (2.5).

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + C_1) \times (2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1) \times (\sigma_x^2 + \sigma_y^2 + C_2)} \tag{2.5}$$

dimana μ_x, μ_y adalah rata-rata piksel citra asli dan hasil peningkatan kualitas citra, σ_x, σ_y adalah simpangan baku piksel citra asli dan hasil peningkatan kualitas citra,

adalah σ_{xy} kovariansi antara piksel citra asli dan hasil peningkatan kualitas citra, C_1, C_2 adalah konstanta untuk menghindari pembagian dengan nol.

2.6. Augmentasi Data

Pada proses *training* algoritma berbasis *Deep Learning* (DL) membutuhkan jumlah data yang besar untuk menghindari masalah *overvitting* [59]. Suatu kondisi dimana model terlalu bergantung pada data latih sehingga model tidak dapat melakukan generalisasi pada data baru dan dapat mengurangi kinerja [59], [60]. Augmentasi data merupakan salah satu teknik untuk memperbanyak data latih dengan mentransformasi citra asli pada orientasi yang berbeda, sehingga tercipta keragaman data serta meningkatkan ketahanan model [61]. Transformasi yang paling banyak digunakan dalam augmentasi data yaitu rotasi, refleksi, *flipping*, dan lainnya. Augmentasi rotasi dilakukan dengan memutar citra ke kanan atau ke kiri pada sumbu antara 1° sampai dengan 359°, sedangkan *flipping* merupakan proses augmetasi dengan membalikkan citra ke arah vertikal atau horizontal [12].

2.7. Segmentasi Citra

Segmentasi citra merupakan proses pengambilan objek dari suatu citra dengan membagi piksel citra tersebut ke dalam beberapa bagian berdasarkan tekstur, intensitas, tingkat abu-abu, dan lain-lain [62], [63]. Pada kasus penyakit menular yang menyerang paru-paru, segmentasi paru-paru pada citra CXR dapat menunjukkan kelainan pada paru-paru sehingga mempermudah proses deteksi penyakit, menghemat waktu pengerjaan, serta meningkatkan akurasi [64]. Dalam segmentasi paru-paru, citra CXR akan dibagi menjadi dua bagian yaitu paru-paru sebagai *foreground* dan bagian lainnya selain paru-paru sebagai *background*. Hasil dari segmentasi paru-paru ini dapat dilanjukan ke tahapan berikutnya yaitu proses klasifikasi penyakit *COVID-19.*

2.8. Klasifikasi Citra

Klasifikasi citra merupakan suatu proses untuk mengkategorikan citra ke dalam salah satu label yang telah ditentukan menggunakan model tertentu [65].

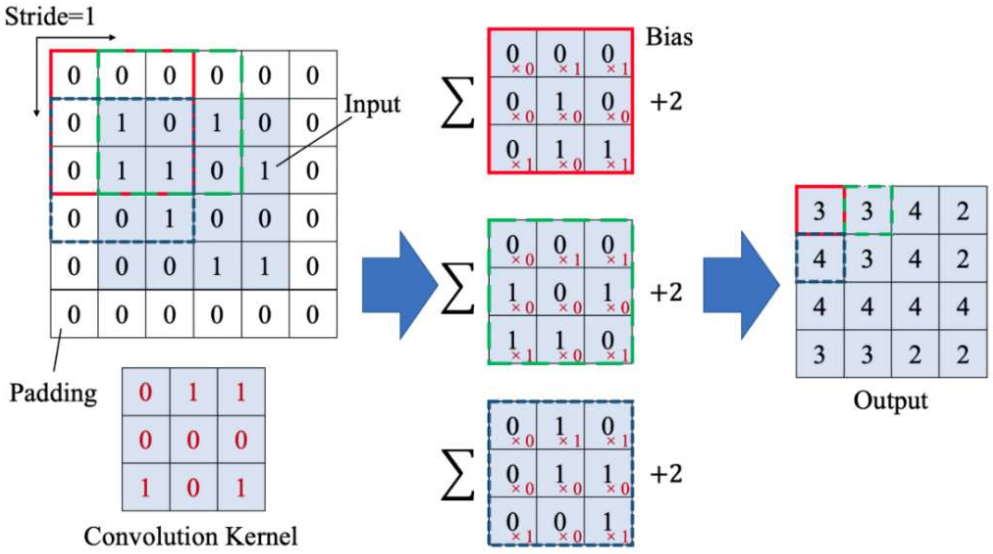
Klasifikasi citra terdiri dari dua proses antara lain tahap *training* dan *testing*. Tahap *training* merupakan suatu proses membangun model klasifikasi dengan melakukan pelatihan model menggunakan data latih yang telah ditentukan, sedangkan tahapan *testing* merupakan suatu proses memberi label pada data citra menggunakan model klasifikasi yang diperoleh dari *training*.

2.9. *Convolutional Neural Network* (CNN)

Convolutional Neural Network (CNN) merupakan salah satu metode *Deep Learning* yang menggabungkan tahapan ekstraksi fitur dan klasifikasi yang telah terbukti keberhasilannya dalam berbagai aplikasi *computer vision* seperti deteksi objek, *natural language processing*, segmentasi, serta klasifikasi citra [66], [67]. Dalam sistem klasifikasi citra, CNN lebih unggul dibandingkan metode konvensional lainnya karena tidak memerlukan tahapan *preprocessing* citra [68]. Komponen CNN terdiri dari tiga lapisan diantaranya *input layer*, *hidden layer*, dan *output layer*. *Hidden layer* pada CNN berfungsi sebagai ekstraksi fitur yang terdiri dari beberapa *convolution layer, pooling layer*, dan fungsi aktivasi, sedangkan pada *output layer* dilakukan proses klasifikasi dengan menerapkan *fully connected layer* dengan *output* berupa probabilitas [69].

2.9.1. *Convolution Layer*

Convolution layer merupakan lapisan utama pada CNN yang melakukan operasi konvolusi untuk mempelajari representasi fitur dari *input* citra [69]. *Convolution layer* menerapkan beberapa kernel untuk menghasilkan sejumlah *feature map* yang mewakili berbagai karakteristik dari *input* citra [70]. Kernel konvolusi (*filter*) merupakan matriks bobot berukuran $n \times n$ dengan entri-entri matriks berupa angka yangdibangkitkan secara acak. Pada proses *convolution layer* terdapat beberapa parameter yang digunakan diantaranya stride dan *padding*. *Stride* merupakan jumlah langkah pergeseran kernel pada saat operasi konvolusi dilakukan. *Padding* merupakan penambahan jumlah piksel yang berisi nilai 0 di sekeliling *input* citra untuk menjaga ukuran dimensi citra [69]. Adapun ilustrasi operasi konvolusi dapat dilihat pada Gambar 2.2 [71].



Gambar 2.2. Proses operasi konvolusi

Pada Gambar 2.2 terlihat proses operasi konvolusi antara *input citra* dengan kernel konvolusi. *Input* citra direpresentasikan dalam bentuk matriks berukuran 4×4serta adanya penambahan *padding same* sehingga matriks *input* berukuran 6×6. Matriks *input* tersebut dikonvolusi dengan kernel berukuran 3×3 dengan *stride* 1. Operasi konvolusi pertama diawali dengan operasi konvolusi antara area lokal matriks *input* yang ditandai dengan kotak berwarna merah dengan kernel konvolusi. Selanjutnya, kernel konvolusi bergeser satu langkah ke kanan yang ditunjukkan pada kotak berwarna hijau dan dilakukan seperti pada operasi konvolusi pertama. Lakukan proses tersebut hingga operasi konvolusi terakhir sehingga menghasilkan *feature map* berukuran 4×4 .Untuk menghitung hasil dari *convolution layer* dapat menggunakan Persamaan (2.6) sebagai berikut [72]:

$$C_q^p = \left(A_p * K_q\right) + b_q \tag{2.6}$$

dengan $*$ merupakan operasi konvolusi dan untuk menghitung entri matriks dari hasil konvolusi dapat menggunakan Persamaan (2.7) sebagai berikut:

$$c_{i,j} = \left(\sum_{u=0}^{n-1} \sum_{v=0}^{n-1} \left(a_{u+i,v+j} \times k_{u+1,v+1} \right) \right) + b_q \tag{2.7}$$

dimana, i merupakan baris, j adalah kolom, n adalah ukuran tinggi kernel, A_p merupakan matriks *input* ke-p, K_q adalah matriks kernel ke-q, $b_q$adalah bias untuk kernel ke-q, C_q^p merupakan matriks hasil konvolusi (*feature maps)* kernel ke-q pada *input* ke-p, $a_{i,j}$ adalah entri baris ke-i dan kolom ke-j pada matriks A_p, $k_{i,j}$ adalah entri baris ke-i dan kolom ke-j pada matriks kernel K_q, dan $c_{i,j}$ adalah entri baris ke-i dan kolom ke-j pada matriks C_q^p.

2.9.2. *Batch Normalization*

Batch normalization merupakan lapisan tambahan yang berfungsi untuk menormalisasikan nilai *input* pada lapisan berikutnya, mengurangi risiko *overfitting*, dan mempercepat proses *training* [70], [73]. *Batch normalization* mentransformasikan distribusi nilai *input* menjadi distribusi normal standar dengan rata-rata 0 dan varians 1 [74]. *Batch normalizaton* menormalisasikan *input* citra dengan mengurangi setiap entri *input* citra dengan rata-rata *mini-batch*, kemudian membaginya dengan varians *mini-batch* ditunjukkan pada Persamaan (2.8) berikut ini [75].

$$\hat{c}_{i,j} = \frac{c_{i,j} - \mu_j}{\sqrt{\sigma_j^2 + \varepsilon}} \tag{2.8}$$

Rata-rata (μ_j) dan variansi $\left(\sigma_j^2\right)$ untuk masing-masing *mini-batch* dari matriks *input* $m \times n$ dapat dihitung menggunakan Persamaan (2.9) dan (2.10) sebagai berikut [75]:

$$\mu_j = \frac{1}{m} \sum_{i=1}^{m} c_{i,j} \tag{2.9}$$

$$\sigma_j^2 = \frac{1}{m}\sum_{i=1}^{m}\left(c_{i,j} - \mu_j\right)^2 \tag{2.10}$$

dimana, n adalah jumlah *mini-batch* (kolom), m adalah banyaknya data dalam satu *mini-batch* (baris), c_{ij} merupakan entri matriks *input* C pada baris ke-i dan kolom ke-j, $\hat{c}_{ij}$ merupakan entri matriks *input* C yang telah dinormalisasi pada baris ke-i dan kolom ke-j, dan ε merupakan konstanta positif terkecil.

2.9.3. Fungsi Aktivasi

Dalam CNN, fungsi aktivasi diterapkan pada *output* dalam setiap *convolution layer* dan *fully connected layer* untuk menambahkan faktor nonlinier sehingga mampu mengatasi berbagai masalah yang kompleks seperti meningkatkan kecepatan saat *training* [75]. Fungsi *Rectified Linear Unit* (*ReLU*), *sigmoid*, *swish*, dan *softmax* merupakan beberapa contoh fungsi aktivasi yang sering digunakan. *ReLU* merupakan fungsi aktivasi dengan biaya komputasi yang rendah dan konvergensi gradien yang lebih baik [70]. *ReLU* membuat seluruh nilai piksel pada citra yang bernilai negatif menjadi 0, sedangkan akan bernilai sama apabila nilainya positif. Fungsi aktivasi *sigmoid* merupakan fungsi aktivasi dengan *input* berupa bilangan riil dan *output*-nya berada dalam interval $(0, 1)$ [76]. Fungsi aktivasi *swish* merupakan fungsi aktivasi yang lebih efisien untuk memproses data besar dan jaringan kompleks yang lebih dalam. Kemudian, untuk fungsi *softmax* biasanya aktivasi digunakan pada *output layer* untuk menghasilkan *output* dalam distribusi probabilitas sehingga nilai *output* berada dalam interval$(0, 1)$ dan total jumlahnya adalah 1 [76]. Secara matematis, fungsi aktivasi *ReLU*, *sigmoid*, *swish*, dan *softmax* didefinisikan pada Persamaan (2.11), (2.12), (2.13), dan (2.14) [75], [76].

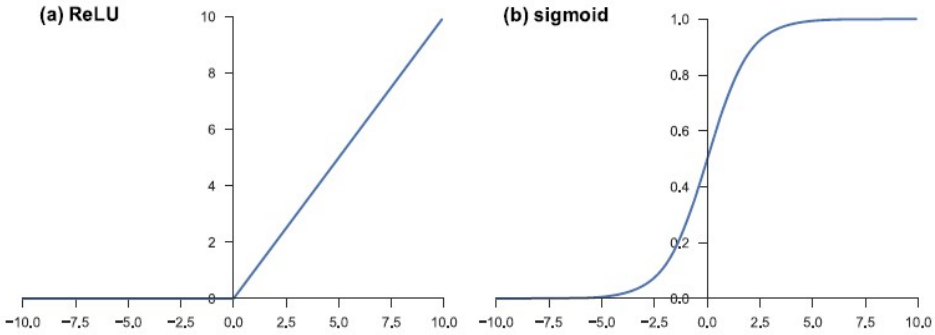
$$r\left(\hat{c}_{i,j}\right) = max\left(\hat{c}_{i,j}, 0\right) = \begin{cases} \hat{c}_{i,j}\ jika\ \hat{c}_{i,j} \geq 0 \\ 0 \quad jika\ \hat{c}_{i,j} < 0 \end{cases} \tag{2.11}$$

$$\sigma\left(\hat{c}_{i,j}\right) = \frac{1}{1 + e^{-\hat{c}_{i,j}}} \tag{2.12}$$

$$s(\hat{c}_{i,j}) = \hat{c}_{i,j} \left(\frac{1}{1 + e^{-\beta(\hat{c}_{i,j})}} \right) \tag{2.13}$$

$$p(\hat{c}_{i,j}) = \frac{e^{\hat{c}_{i,j}}}{\sum_{k=1}^{n} e^{\hat{c}_{i,j}}} \quad ; i = i, 2, \ldots, n \tag{2.14}$$

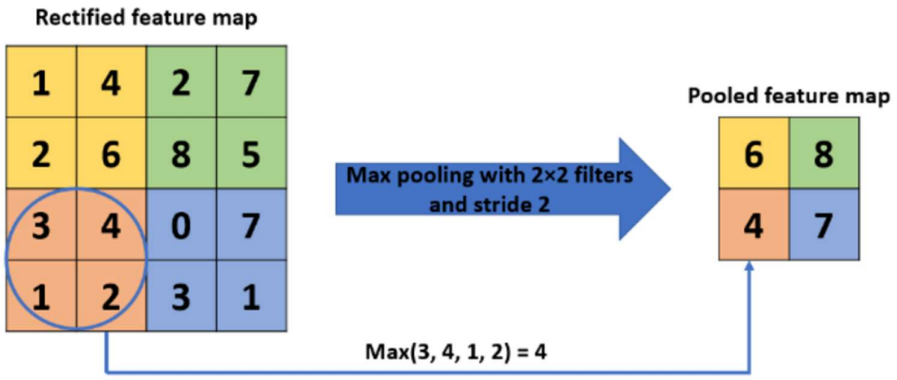
untuk $\hat{c}_{i,j} \in (-\infty, +\infty)$, dimana $r(\hat{c}_{i,j})$ adalah fungsi aktivasi *ReLU*, $\sigma(\hat{c}_{i,j})$ adalah fungsi aktivasi sigmoid dengan $\sigma(\hat{c}_{i,j}) \in (0,1)$, $s(\hat{c}_{i,j})$ adalah fungsi aktivasi *swish* dengan β merupakan parameter yang dipelajari, dan $p(\hat{c}_{i,j})$merupakan nilai probabilitas *input* ke-i dari fungsi *softmax*. Gambar grafik fungsi *ReLU* dan *sigmoid* dapat dilihat pada Gambar 2.3. sebagai berikut [69]:



Gambar 2.3. Grafik fungsi aktivasi *ReLU* dan *sigmoid*

2.9.4. *Pooling Layer*

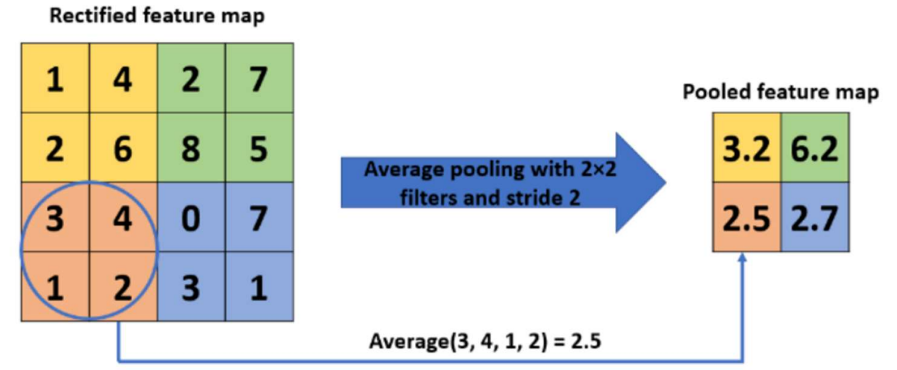
Pooling layer merupakan jenis *hidden layer* yang sangat umum digunakan pada CNN. Lapisan ini membantu memperkecil dimensi *feature map* dengan mempertahankan sebagian besar fitur dominan di setiap langkah tahap *pooling* (Alzubaidi *et al.*, 2021). *Pooling layer* melibatkan dua parameter utama untuk melakukan operasi penyatuan, yaitu ukuran kernel *pooling* dan *stride*. Adapun beberapa jenis *pooling layer* yang digunakan antara lain *max pooling* dan *average pooling*. *Max pooling* bekerja dengan mengambil nilai maksimum pada setiap submatriks yang dihasilkan melalui pergeseran kernel *pooling* dan s*tride*. Ilustrasi *max pooling* dapat dilihat pada Gambar 2.4 [77]. Ilustrasi *max pooling* dapat dilihat pada Gambar 2.4. [77].



Gambar 2.4. Ilustrasi proses *max pooling* 2 × 2

Pada Gambar 2.4. dapat dilihat proses *max pooling* dengan matriks *input* berukuran 4 × 4. Matriks *input* akan dipartisi menjadi 4 submatriks karena kernel *pooling* yang digunakan berukuran 2 × 2 dan stride 2 yang ditunjukkan pada kotak berwarna kuning, hijau, oranye, dan biru. Selanjutnya, rata-rata dari masing-masing submatriks yang telah terbentuk dihitung sehingga menghasilkan *output* berukuran 2 × 2.

Average pooling bekerrja dengan mengambil nilai rata-rata pada setiap submatriks yang dihasilkan melalui pergeseran kernel *pooling* dan s*tride*, sedangkan *global average pooling* dengan mengambil nilai rata-rata dari *input* citra. Ilustrasi *average pooling* dapat dilihat pada Gambar 2.4 [77].

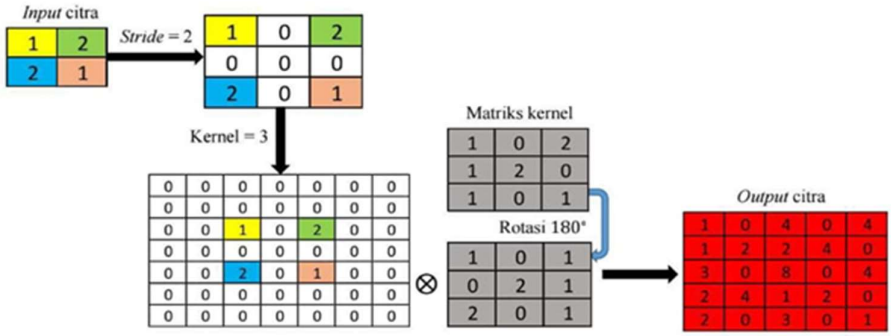


Gambar 2.5. Ilustrasi proses *average pooling* 2 × 2 dan *stride* 2

Pada Gambar 2.5. dapat dilihat proses *average pooling* dengan matriks *input* berukuran 4×4 mengunakan kernel *pooling* 2×2 dan *stride* 2. Pertama-tama, matriks *input* akan dipartisi menjadi 4 submatriks yang ditunjukkan pada kotak berwarna kuning, hijau, oranye, dan biru. Masing-masing submatriks yang telah terbentuk diambil nilai rata-ratanya sehingga menghasilkan matriks *output* yang berukuran 2×2.

2.9.5. *Transposed Convolution*

Transposed Convolution merupakan kebalikan dari *pooling layer* yang mana lapisan ini berfungsi untuk meningkatkan dimensi *feature map* untuk menghasilkan dimensi feature maps yang sama dengan *input* citra awal [78]. Ilustrasi proses *transposed convolution* dapat dilihat pada Gambar 2.6.



Gambar 2.6. Ilustrasi *transposed convolution* 2×2 dan *stride* 2

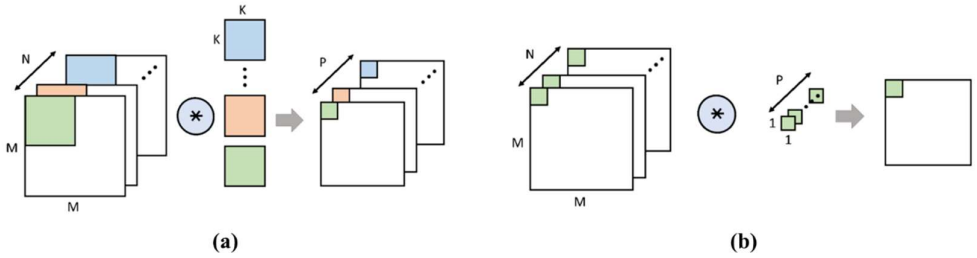
Pada Gambar 2.6 menujukkan proses *transposed convolution* mengunakan kernel 2×2 dan *stride* 2 terhadap matriks *input* berukuran 2×2. Proses *transposed convolution* diawali dengan penambahan angka 0 diantara setiap entri dari matriks *input*, dimana jumlahnya yaitu nilai *stride* dikurangi 1. Selanjutnya keseluruhan entri dari matriks *input* ditambahkan dengan 0 mengikuti aturan *padding*. Banyaknya penambahan angka 0 adalah ukuran matriks kernel dikurangi 1. Lalu matriks kernel dirotasi 180° dan dilakukan operasi konvolusi dengan matriks *input* yang berukuran 7×7. Setelah semua entri di dalam matriks tersebut di konvolusi, maka akan menghasilkan *output* berukuran 6×6 yang ditandai dengan kotak warna merah.

2.9.6. *Concatenate Layer*

Concatenate layer merupakan lapisan yang berfungsi untuk menggabungkan dua *feature map* yang berdimensi sama [79]. *Concatenate* dapat memekstraksi citra lebih baik dengan menghasilkan fitur yang lebih halus dan memberikan detail yang tajam [80]. Penggunaan *concatenate* juga mampu menangani beberapa detail fitur citra yang hilang sehingga memungkinkan informasi yang relevan dapat digunakan sebagai fitur citra [81].

2.9.7. *Depthwise Separable Convolution*

Depthwise separable convolution telah terbukti berhasil dalam klasifikasi citra, dimana lapisan ini mampu menghindari ekstraksi fitur yang berlebihan dan mengurangi jumlah paremeter serta komputasi yang digunakan dalam operasi konvolusi [82], [83]. Lapisan ini biasanya disebut dengan "*separable convolution*" yang terdiri dari dua lapisan konvolusi yaitu *depthwise convolution* dan *pointwise convolution*. *Depthwise convolution* merupakan konvolusi spatial yang dilakukan secara independen pada setiap *input channel* sehingga setiap *channel*. Lapisan selanjutnya yaitu *pointwise convolution*, proses konvolusi yang menggunakan kernel berukuran 1×1 untuk menggabungkan semua hasil *feature map* dari *depthwise convolution* sehingga menghasilkan *feature map* baru [83], [84]. Ilustrasi proses dalam *dethwise separabel convolution* dapat dilihat pada Gambar 2.7.



Gambar 2.7. Proses *depthwise separabel convolution* (a) *depthwise convolution* (b) *pointwise convolution*

Pada Gambar 2.7. menunjukkan bagaimana *depthwise convolution* dan *pointwise convolution* bekerja.Sebuah matriks *input* berukuran $N \times N \times L$ dan kernel konvolusi berukuran $N \times N \times L$, dimana L merupakan banyaknya *channel*.

Pada proses *depthwise convolution*, setiap *channel* pada matriks *input* dikonvolusi secara independen dengan masing kernelnya sehingga menghasilkan *feature map* berukuran $N \times N \times L$. Selanjutnya, diproses ke dalam *pointwise convoluion*. *Feature map* hasil *depthwise convolution* dikonvolusi dengan kernel berukuran $1 \times 1 \times L$ sehingga menghasilkan *feature map* berukuran $N \times N \times 1$. Adapun untuk menghitung hasil *depthwise convolution* dapat menggunakan Persamaan (2.15) sebagai berikut [83]:

$$D_{l_{(i,j)}} = \sum_{u=0}^{n-1} \sum_{v=0}^{n-1} \left(W_{l_{(u+i,v+j)}} \times K_{l_{(u+1,v+1)}} \right) \qquad ; i,j = i,2,\dots,n \tag{2.15}$$

dimana D_l merupakan matriks *depthwise convolution channel* ke-l,W_l merupakan matriks *input channe l*ke-l, K_l merupakan matriks kernel *channel* ke-l, dan $D_{l_{(i,j)}}$ merupakan entri matriks D_l baris ke-i dan kolom ke-j. Kemudian, dilanjutkan dengan proses *pointwise convolution* menggunakan kernel berukuran 1×1 yang dapat dihitung dengan menerapkan Persamaan (2.16) sebagai berikut [83]:

$$P_{l_{(i,j)}} = \sum_{u=0}^{n-1} \left(H_{l_{(i,j,u+1)}} \times K_{l_{(i,j,u+1)}} \right) \qquad ; i,j = i,2,\dots,n \tag{2.16}$$

dimana P_l merupakan matriks *pointwise convolution* ke-l,H_l merupakan matriks *input* ke-l,dan $P_{l_{(i,j)}}$ merupakan entri matriks P_l baris ke-i dan kolom ke-j.

2.9.8. *Dense Layer*

Dense layer merupakan lapisan yang berfungsi untuk mengklasifikasikan citra yang telah melewati proses ekstraksi fitur ke dalam label yang telah ditentukan. Lapisan ini bekerja dengan mengumpankan seluruh *ouput* pada lapisan sebelumnya ke semua neuronnya, dimana setiap neuron tersebut akan menghasilkan satu *output* ke lapisan berikutnya dan kemudian diproses ke dalam fungsi aktivasi [85].

2.9.9. *Dropout*

Pada proses *training*, masalah yang sering terjadi adalah ketidaksesuaian antara model data latih dan jumlah dataset yang tidak seimbang (*overfitting*). Untuk mencegah terjadinya *overfitting* dapat dilakukan dengan melakukan penambahan *dropout*. Cara kerja *dropout* dilakukan dengan menghilangkan neuron secara acak pada setiap lapisan yang ada pada arsitektur sesuai dengan *dropout rate* yang digunakan [86].

2.9.10. *Loss Function*

Pada *output layer* CNN juga terdapat *loss function* yang digunakan untuk menghitung kesalahan selama proses *training* [76]. Nilai ini merupakan perbedaan antara nilai aktual dengan hasil klasifikasi menggunakan metode yang diusulkan. *Loss function* terdiri dari beberapa macam diantaranya *Binary Cross Entropy* (BCE) dan *Categorical Cross Entropy* (CCE). BCE merupakan *loss function* yang digunakan dalam tugas klasifikasi biner, sedangkan CCE digunakan untuk klasifikasi multi-label. *Loss function* BCE dapat dihitung menggunakan Persamaan (2.17) sebagai berikut [87]:

$$L_{BCE} = -\frac{1}{m \times n}\left[\sum_{i=1}^{m}\sum_{j=1}^{n}\left(y_{i,j}\log\left(p_{ij}\right) + \left(1 - y_{ij}\right)\log\left(1 - p_{ij}\right)\right)\right] \tag{2.17}$$

dimana $m \times n$ menunjukkan ukuran matriks hasil segmentasi, y_{ij}merupakan entri matriks label *ground truth* pada baris ke-i kolom ke- j(0 untuk piksel *background* dan 1 untuk piksel paru-paru), dan p_{ij} merupakan entri matriks hasil segmentasi pada baris ke-i kolom ke-j, dan L adalah nilai *loss fuctionbinary cross entropy*.

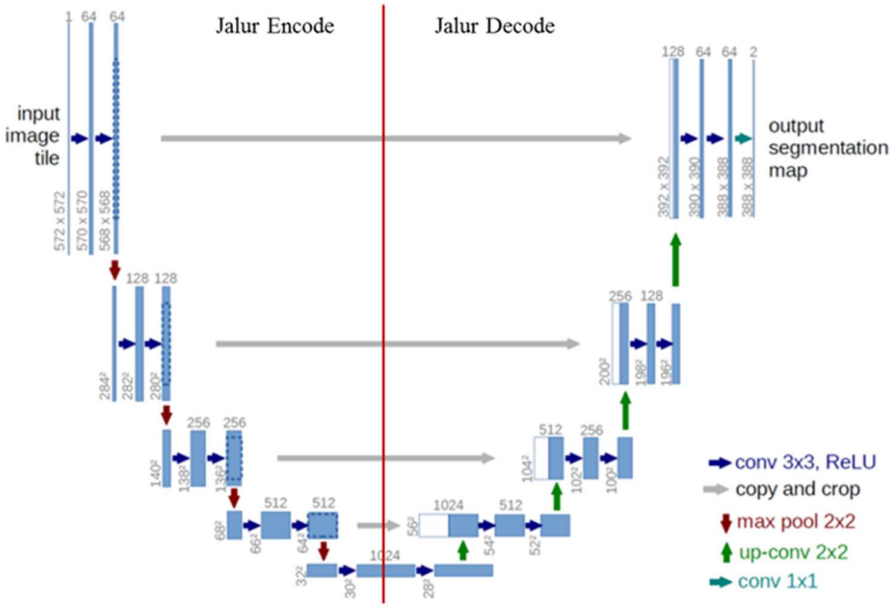
Untuk menghitung hasil *loss function* CCE dapat mengunakan Persamaan (2.18) dengan menerapkan fungsi softmax sebagai berikut [88]:

$$L_{CCE}(p, q) = -\sum_{k}^{K} p(x_k)\log q(x_k) \qquad ;\ x_k \in K \tag{2.18}$$

dimana $p(x_k)$ merupakan probabilitas dari label klasifikasi pada label ke-k, $q(x_k)$ probabilitas dari label aktual pada label ke-k, K adalah himpunan label klasifikasi, dan $L(p, q)$ adalah hasil *loss function categorical cross entropy*.

2.10. *U-Net*

U-Net merupakan salah satu arsitektur CNN yang pertama kali diperkenalkan oleh [17] dan telah terbukti keberhasilannya dalam segmentasi citra terutama pada citra medis. Arsitektur ini berbentuk seperti huruf "U" yang terdiri dari dua jalur yaitu jalur *encode* dan jalur *decode*. Pada jalur *encode* digunakan untuk mengambil informasi dari *input* citra, sedangkan jalur *decode* untuk menentukan objek yang diinginkan secara efisien [89]. Arsitektur *U-Net* pada citra CXR dapat dilihat pada Gambar 2.8. [17].



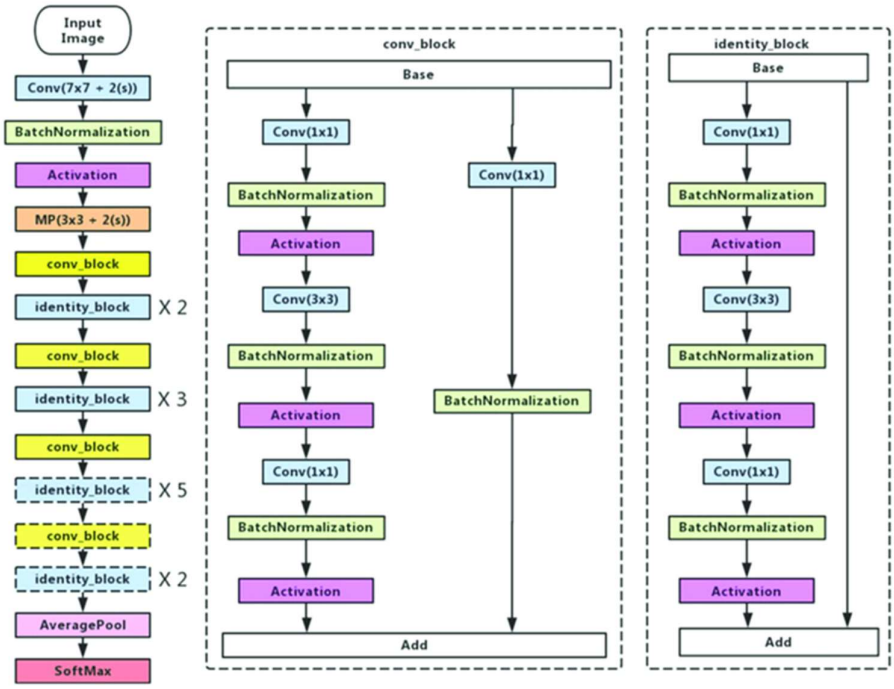
Gambar 2.8. Ilustasi arsitektur *U-Net*

Pada Gambar 2.8 menunjukkan lapisan-lapisan yang digunakan pada arsitektur *U-Net* dengan *input* citra CXR. Pada setiap blok jalur *encode* terdiri dari dua lapisan konvolusi dengan kernel berukuran 3×3 masing-masing diikuti dengan fungsi aktivasi *ReLU*, kemudian dilakukan *max pooling* 2×2 dengan

stride 2 untuk pengurangan dimensi citra. Pada setiap langkah *max pooling*, jumlah filter dimulai dari 64, kemudian untuk blok selanjutnya menjadi 128, 256, 512, dan 1024. Pada setiap blok di jalur *decode* terdiri dari *upsampling* 2×2 dan diikuti dengan penggabungan *feature maps* dari jalur *encoder* dan *feature maps* hasil *upsampling* menggunakan *concatenate*. Kemudian dilanjutkan dengan konvolusi sebanyak dua kali menggunakan kernel berukuran 3×3, dan masing-masing diikuti oleh fungsi aktivasi *ReLU*. Pada lapisan terakhir, digunakan konvolusi 1×1 untuk memetakan setiap nilai piksel *feature maps* ke kelas yang diinginkan. *Output* citra CXR setelah dilakukan konvolusi 1×1 yaitu citras hasil segmentasi dengan paru-paru yang ditandai dengan piksel berwarna putih, dan *background* yang ditandai dengan piksel berwarna hitam.

2.11. *Residual Network*

Residual Network (*ResNet*) memanfaatkan "*skip connection*" dalam lapisannya untuk menangani masalah penurunan gradien, sehingga mampu mempercepat konvergensi jaringan dalam [76]. Arsitektur ini memiliki kompleksitas komputasi yang lebih rendah dibandingkan arsitektur lainnya, meskipun kedalamannya diperbesar [25]. Jaringan yang dalam pada arsitektur *ResNet* terbukti lebih baik dalam melakukan tugas klasifikasi karena dapat mengekstraksi fitur lebih representatif [13]. Arsitektur *ResNet* yang paling terkenal adalah *ResNet-50*, yang memiliki 50 lapisan. Namun, ada juga variasi lain seperti *ResNet-18*, *ResNet-101*, dan sebagainya, yang berbeda dalam jumlah lapisan dan kompleksitasnya. Arsitektur *ResNet* telah digunakan dalam banyak tantangan pengenalan citra dan telah menunjukkan performa yang sangat baik dalam banyak tugas, termasuk klasifikasi citra .Adapun ilustrasi arsitektur *ResNet* dapat dilihat pada Gambar 2.9 [90].



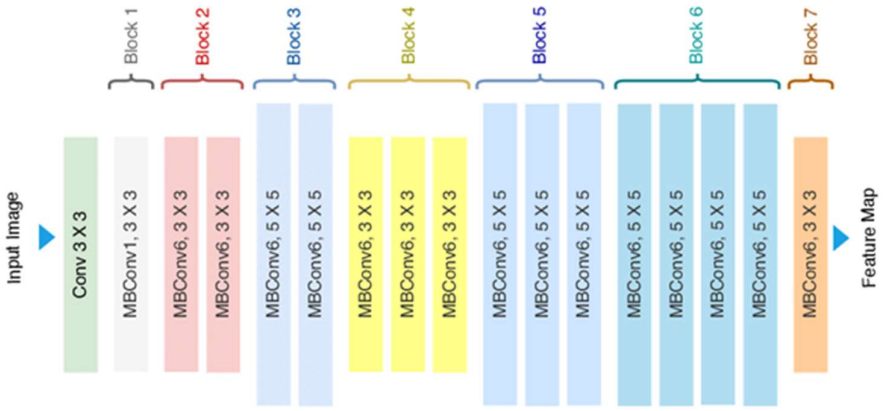
Gambar 2.9. Ilustrasi arsitektur *ResNet*

Pada Gambar 2.9. dapat dilihat bahwa arsitektur *ResNet* terdapat dua jenis *block* yang digunakan yaitu *identity_block* dan *conv_block* yang mana pada *block* tersebut terdapat *skip connection* untuk mengatasi masalah penurunan gradien akibat jaringan yang terlalu dalam. Pada bagian *conv_block*, *input* diproses ke dalam *convolution layer* dan *batch normalization* yang diikuti oleh fungsi aktivasi *ReLU* sebanyak tiga kali. Lalu, *input* juga melewati *residual connection* yang diproses ke dalam *convolution layer* dan *batch normalization*. Kemudian, dilakukan penambahan hasil dari kedua proses tersebut. Pada bagian *identity_block*, *input* diproses ke dalam *convolution layer* dan *batch normalization* yang diikuti oleh fungsi aktivasi *ReLU* sebanyak tiga kali. Kemudian, hasil tersebut ditambahkan dengan *input*.

2.12. *EfficientNet*

EfficientNet merupakan salah satu arsitektur CNN yang memiliki parameter yang sedikit dan menghasilkan akurasi klasifikasi yang lebih baik [91]. Arsitektur ini diperkenalkan oleh Tan and Le [29], dalam penelitiannya arsitektur *EfficientNet*

mengungguli jumlah parameter dan akurasi dari semua arsitektur sebelumnya pada dataset *ImageNet*. Keunggulan arsitektur *EfficientNet* dibandingkan arsitektur lainnya yaitu dapat menskalakan lebar lapisan, resolusi citra, kedalaman lapisan, dan lainnya secara efisien, serta mengurangi set parameter sehingga proses *training* lebih efisien [30]. Blok utama yang menyusun arsitektur ini adalah *Mobile inverted bottleneck convolution* (*MBConv*) yang dapat dilihat pada Gambar 2.10. [92].

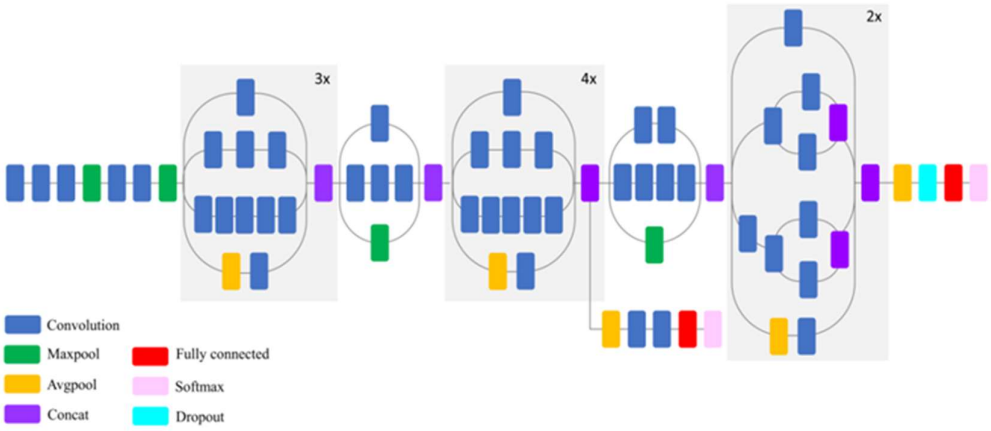


Gambar 2.10. Ilustrasi arsitektur *EfficientNet*

Pada Gambar 2.10. terlihat bahwa arsitektur *EfficientNet* terdiri dari beberapa lapisan diantaranya *convolution layer* dengan kernel berukuran 3×3 dan *MBConv.* Terdapat dua tipe *MBConv* yang digunakan yaitu *MBConv1* dan *MBConv6. MBConv1* hanya digunakan satu kali dengan kernel berukuran 3×3, sedangkan *MBConv6* digunakan setelahnya dengan ukuran kernel yang berbeda (3×3 dan 5×5). *MBConv* terdiri dari *Depthwise Convolution*, *Batch Normalization*, fungsi aktivasi *Swish*, *Squeeze and Excitation* (SE), *Convolution*, dan *Add*.

2.13. *Inception-v3*

Aristektur *Inception-v3* diperkenalkan pertama kali oleh Szegedy et al [93] yang menggunakan blok dasar yaitu *inception block.* Blok ini berfungsi untuk meminimalkan biaya komputasi tanpa mempengaruhi jaringan yang lebih dalam dan *overvitting* melalui pengurangan dimensi dengan konvolusi 1×1 yang bertumpuk [94]. Arsitektur ini memiliki 48 lapisan yang terdiri dari 11 *inception block* yang dapat dilihat pada Gambar 2.11 [95].



Gambar 2.11. Ilustrasi arsitektur *Inception-v3*

Pada Gambar 2.10 menunjukkan bahwa lapisan-lapisan yang menyusun arsitektur *Inception-v3* antara lain *convolution layer*, *max pooling*, *inception block*, *global average pooling*, *concatenate*, *dense layer*, dan *softmax*. Arsitektur ini terdiri dari 11 blok awal (3 blok A, 1 blok B, 4 blok C, 1 blok D, dan 2 blok E), masing-masing dengan jumlah lapisan konvolusi dan ukuran kernel yang bervariasi. *Output* dari setiap jalur di *inception block* digabungkan menggunakan *concatenate*.

2.14. *Ensemble Learning*

Pendekatan *ensemble learning* dilakukan dengan melatih beberapa arsitektur CNN dan menggabungkan hasil klasifikasinya untuk menghasilkan model baru yang lebih akurat daripada hasil klasifikasi tunggal [96]. Keuntungan yang dimiliki metode ini yaitu dapat memanfaatkan informasi dari beberapa pengklasifikasian, meminimalkan kesalahan klasifikasi, serta kesalahan pembelajaran fitur masih dapat diklasifikasikan dengan benar menggunakan pola yang dipelajari oleh pengklasifikasian lainnya [97]. Teknik yang sering digunakan untuk menggabungkan hasil klasifikasi tunggal dari masing-masing model pada metode *ensemble learning* adalah *weighted voting*. Teknik ini memanfaatkan bobot dari masing-masing arsitektur yang digunakan.

Untuk menentukan hasil klasifikasi menggunakan teknik *weighted voting* dapat menerapkan Persamaan (2.19) sebagai berikut:

$$f_i = \sum_{i=1}^{n} (w_i g_i) \tag{2.19}$$

dimana f_i merupakan hasil klasifikasi label untuk $f_i \in [0,\ 1]$, w_i merupakan bobot hasil *training* untuk arsitektur ke-i, dan g_i merupakan hasil fungsi aktivasi *softmax* pada arsitektur ke-i.

2.15. *Confusion Matrix*

Confusion matrix merupakan konsep dari *deep learning* yang digunakan untuk mengukur kinerja model, yang berisi informasi tentang label aktual dengan hasil yang diperoleh [65], [94]. Secara umum, *confusion matrix* direpresentasikan sebagai matriks bujur sangkar ($n \times n$) dimana baris menyatakan label aktual dan kolom menyatakan hasil yang diperoleh, dimana entri-entrinya terdiri dari *True Positive* (TP), *False Positive* (FP), *False Negative* (FN), dan *True Negative* (TN). Adapun *confusion matrix* untuk segmentasi biner dan klasifikasi multikelas pada citra CXR dapat dilihat pada Tabel 2.1. dan 2.2 [62], [98].

Tabel 2.1. *Confusion matrix* untuk segmentasi paru-paru pada citra CXR

Ground Truth	Hasil Segmentasi	
	Paru-paru	*Background*
Paru-paru	TP	FN
Background	FP	TN

Tabel 2.2. *Confusion matrix* multikelas untuk klasifikasi penyakit *COVID-19*

Aktual	Hasil Klasifikasi			
	Normal	*COVID-19*	*Pneumonia*	*Lung Opacity*
Normal	TN	FP	TN	TN
COVID-19	FN	TP	FN	FN
Pneumonia	TN	FP	TN	TN
Lung Opacity	TN	FP	TN	TN

Berdasarkan Tabel 2.1 dan 2.2 nilai TP menunjukkan jumlah citra/pikselyang termasuk dalam kelas positif yang diprediksi dengan benar, nilai TN menunjukkan

jumlah citra/piksel yang termasuk kelas negatif yang diprediksi dengan benar, nilai FN menunjukkan jumlah citra/piksel yang termasuk kelas positif namun diprediksi sebagai kelas negatif, dan nilai FP menunjukkan jumlah citra/pikselyang termasuk dalam kelas negatif namun diprediksi sebagai kelas positif [94], [99].

Nilai TP, TN, FP, dan FN pada *confusion matrix* dapat digunakan untuk menghitung kinerja metode segmentasi dan klasifikasi seperti akurasi, presisi, *recall*, dan *F1-Score*. Akurasi merupakan ukuran untuk melihat keakuratan arsitektur berdasarkan rasio jumlah prediksi yang benar terhadap jumlah total prediksi. Presisi merupakan ukuran untuk melihat kemampuan arsitektur dalam memprediksi kelas negatif dengan benar (TN). *Recall* merupakan ukuran untuk melihat kemampuan arsitektur dalam memprediksi kelas positif dengan benar (TP). *F1-Score* merupakan ukuran untuk melihat kecocokan antara label aktual dengan hasil yang diperoleh dengan menghitung rata-rata harmonik antara presisi dan *recall* [62], [76], [94]. Secara matematis, untuk menghitung nilai akurasi, presisi, *recall* dan *F-1 Score* dapat mengggunakan Persamaan (2.20), (2.21), (2.22), dan (2.23) [62], [100].

$$Akurasi = \frac{TP + TN}{TP + TN + FP + FN} \times 100\% \tag{2.20}$$

$$Presisi = \frac{TP}{TP + FP} \times 100\% \tag{2.21}$$

$$Recall = \frac{TP}{TP + FN} \times 100\% \tag{2.22}$$

$$F1\text{–}Score = \frac{2TP}{2TP + FP + FN} \times 100\% \tag{2.23}$$

Ukuran kinerja suatu algorima dapat dikategorikan ke dalam 5 kriteria yang dapat dilihat pada Tabel 2.3. sebagai acuan kesuksesan algoritma dalam melakukan segmentasi dan klasifikasi [101].

Tabel 2.3. Kategori nilai kinerja arsitektur

Nilai Kinerja (NK) (%)	Kategori
NK > 90	Sangat Baik
80 < NK ≤ 90	Baik
70 < NK ≤ 80	Cukup
60 < NK ≤ 70	Kurang baik
NK ≤ 60	Gagal

Pada Tabel 2.3. Nilai Kinerja (NK) dapat digunakan untuk nilai akurasi, presisi, *recall* dan *F1-score*. Jika nilai kinerja yang diperoleh lebih dari 90% maka nilai tersebut dikategorikan sangat baik. Jika nilai kinerja lebih dari 80% dan kurang dari sama dengan 90% maka nilai tersebut dikategorikan baik. Jika nilai kinerja lebih dari 70% dan kurang dari sama dengan 80% maka nilai tersebut dikategorikan cukup baik. Jika nilai kinerja lebih dari 60% dan kurang dari sama dengan 70% maka nilai tersebut dikategorikan kurang baik. Jika nilai kinerja kurang dari sama dengan 60% maka maka nilai tersebut dikategorikan gagal.

2.16. *State of the Art*

Penyakit *COVID-19* adalah penyakit menular pada saluran pernapasan yang disebabkan oleh virus SARS-CoV-2. Citra *Chest X-ray* (CXR) dapat digunakan sebagai alternatif untuk mendiagnosa penyakit ini, namun memerlukan tingkat ketelitian yang tinggi. Peningkatan kualitas citra CXR telah banyak dilakukan oleh beberapa penelitian terdahulu yang dapat dilihat pada Tabel 2.4.

Berdasarkan Tabel 2.4, metode peningkatan kualitas citra melalui Morfologi *Opening*, *Median Filter*, dan *Gaussian Filter* bertujuan untuk menghasilkan citra CXR yang lebih jelas dan tajam sehingga memudahkan tenaga medis dalam melihat kelainan pada paru-paru pasien. Selanjutnya dilakukan segmentasi citra CXR untuk memfokuskan pada bagian paru-parunya saja. Metode yang berkembang saat ini untuk segmentasi citra adalah *Convolutional Neural Network* (CNN), khususnya arsitektur *U-Net*. Arsitektur *U-Net* efektif digunakan untuk segmentasi citra berbasis piksel. Namun, selama proses *training*, terkadang terjadi perubahan

distribusi fitur input yang dapat menghambat kecepatan pelatihan dan kinerja. Untuk mengatasi masalah ini, diperlukan metode *Batch Normalization* (BN) yang akan menormalkan peta fitur disemua lapisan arsitektur.

Selain segmentasi citra, metode CNN juga banyak digunakan untuk klasifikasi citra. Beberapa arsitektur CNN yang populer adalah *ResNet*, *EfficientNet*, dan *Inception-v3*. *ResNet* menggunakan *skip connection* untuk menangani masalah *vanishing gradient*, sedangkan *EfficientNet* memiliki keunggulan dalam penskalaan dimensi secara seragam. *Inception-v3* menggunakan konvolusi kecil yang menumpuk untuk menghindari *overfitting*. Setiap arsitektur CNN memiliki kelebihan dan kekurangan dalam klasifikasi citra.

Untuk meningkatkan kinerja klasifikasi, metode *ensemble learning* dapat digunakan. *Ensemble learning* menggabungkan hasil klasifikasi dari berbagai arsitektur atau metode klasifikasi tunggal menjadi model baru yang lebih akurat. Metode *weighted voting* pada *ensemble learning* memberikan bobot yang berbeda untuk pengklasifikasi berdasarkan kriteria tertentu, sehingga meminimalkan tingkat kesalahan klasifikasi tunggal. Metode *ensemble learning* baru yang diusulkan adalah ELREI (*Ensemble Learning* dari *ResNet*, *EfficientNet*, dan *Inception-v3*) dengan *weighted voting* untuk klasifikasi penyakit paru-paru berdasarkan citra CXR. Metode ELREI berjalan pada setiap *epoch* pada tahap *training*, sehingga bobot hasil ELREI pada setiap *epoch* pada data *training* dan data validasi dapat dicek pada hasil performa untuk menangani *overfitting*. Beberapa penelitian terdahulu telah melakukan segmentasi dan klasifikasi citra yang dapat dilihat seperti pada Tabel 2.5.

Tabel 2.4. Beberapa penelitian terdahulu yang telah melakukan peningkatan kualitas citra

No	Tahun	Penulis	Metode	Hasil dan Kelemahan
1	2018	Susanto [102]	Menggunakan HE, CLAHE, *Contrast Stretching* (CS), dan *Sharpened Image* (SI) dikombinasikan dengan *Median Filter*	Penggabungan metode *Median Filter* menghasilkan nilai rata-rata MSE sebesar 81,78 serta nilai rata-rata PSNR sebesar 29,04.
2	2020	Umamaheswari & Geetha [103]	Menggunakan metode *Gamma Corrected Gaussian Filtering* (GCGF)	Menghasilkan nilai rata-rata PSNR sebesar 24,61 dan nilai MSE rata-rata sebesar 229,92. Hasil penelitian ini tergolong rendah karena nilai PSNR masih dibawah 30 dan nilai MSE yang terlalu tinggi serta tidak menghitung nilai SSIM.
3	2020	Roy & Maity [104]	Menggunakan kombinasi metode *Wiener Filter* dan *Poisson Noise*	Menghasilkan nilai rata-rata PSNR sebesar 33, nilai rata-rata MSE sebesar 29, dan nilai rata-rata SSIM sebesar 0,7876. Hasil penelitian ini tergolong cukup rendah karena nilai rata-rata PSNR masih dibawah 35 dan nilai SSIM masih dibawah 80.
4	2022	Basset et al [105]	Menggunakan metode *Type-2 Neutrosophic Set*	Menghasilkan nilai rata-rata PSNR sebesar 28,58, nilai rata-rata MSE sebesar 23,6, dan nilai rata-rata SSIM sebesar 0,9. Nilai rata-rata PSNR pada penelitian ini cukup rendah karena dibawah 30.
5	2022	Kesuma et al [106] [1]	Membandingkan antara Metode *Morfologi Opening* dan *Median Filter* serta *Morfologi Opening* dan *Gaussian Filter*	Hasil penelitian pada Morfologi *Opening* dan *Median Filter* yaitu PSNR sebesar 39,187, MSE 22,252, dan SSIM 0,952. Sedangkan pada *Morfologi Opening* dan *Gaussian Filter* yaitu PSNR sebesar 38,717, MSE 23,917, dan SSIM 0,956.

[1] Kesuma, L. I., Ermatita, Erwin, Sari, P., & Purabaya, R. H. (2022). Improved Chest X - Ray Image Quality Using Median and Gaussian Filter Methods. *2022 International Conference on Informatics, Multimedia, Cyber and Information System (ICIMCIS)*. (Lampiran 1)

Tabel 2.5. Beberapa penelitian terdahulu yang telah melakukan segmentasi dan klasifikasi citra

No	Tahun	Penulis	Metode	Hasil dan Kelemahan
Segmentasi menggunakan Metode *U-Net Batch Normalization*				
1	2018	Tong et al. [107]	Menerapkan modifikasi arsitektur *U-Net* dengan *batch normalization* dan *residual connection* untuk segmentasi nodul paru-paru pada citra *CT Scan* paru-paru	Menghasilkan nilai *IoU* tertinggi dibandingkan dengan arsitektur *U-Net* dasar yaitu sebesar 73,6%. Tetapi tidak menjelaskan hasil akurasi, sensitivitas, spesifisitas, dan *F1-Score* sebagai evaluasi kinerja segmentasi citra lainnya
2	2020	Smith et al. [108]	Menggunakan modifikasi arsitektur *U-Net* menggunakan *Group normalization* untuk segmentasi akar tanaman pada citra tanah	Hasil akurasi, sensitivitas, dan F1-Score masing-masing sebesar 99,7%; 74,8%; dan 70,1%. Namun, tidak menjelaskan hasil spesifisitas, dan *IoU* sebagai evaluasi kinerja segmentasi citra lainnya
Klasifikasi menggunakan Metode Konvensional				
3	2021	Mohammed et al. [67]	Menggunakan *machine learning* (*Random Forest* dan *CN2 Rule Inducer*)	➢ Rata-rata nilai ukuran kinerja pada keempat metode yang diusulkan masih dibawah 90% kecuali hasil sensitivitas pada metode Xception. ➢ Tidak menjelaskan hasil F1-Score sebagai evaluasi kinerja klasifikasi citra lainnya.
4	2021	Akinnuwesi et al. [109]	Menggunakan metode *Logistic Regression* (LR), *Support Vector Machine* (SVM), *Naive Bayes* (NB), *Decision Tree* (DT), *Multilayer Perceptron* (MLP), *Fuzzy Cognitive Map* (FCM) dan *Deep Neural Network* (DNN)	➢ Rata-rata nilai ukuran kinerja pada ketujuh metode yang diusulkan masih dibawah 85%.
5	2019	Kadry et al. [110]	Menggunakan metode *Naive Bayes* (NB), *K-Nearest Neighbors* (KNN), *Decision Tree* (DT), *Random Forest* (RF) and *Support Vector Machine with linear kernel* (SVM)	➢ Rata-rata nilai kinerja evaluasi yang diperoleh masih dibawah 84%, selain itu nilai performa kinerja metode lainnya seperti *F1-score* tidak dihitung.
6	2020	Vinod & Prabaharan [111]	Menggunakan metode *Decision Tree*	➢ Nilai kinerja evaluasi yang dihasilkan pada dataset *CT-Scan* untuk Presisi dan *F-1 Score* masih cukup rendah yaitu masih dibawah 85%, selain itu nilai kinerja lain seperti Akurasi tidak dihitung.

No	Tahun	Penulis	Metode	Hasil dan Kelemahan
7	2021	Sharma et al. [112]	Menggunakan metode *Improved Random Forest* (IRF), *Support Vector Machine* (SVM), dan *Random Forest* (RF)	➢ Nilai evaluasi kinerja masih kurang baik dalam mengklasifikasikan penyakit *COVID-19* karena nilai yang diperoleh rata-rata masih dibawah 77%. ➢ Tidak menghitung nilai akurasi.
Klasifikasi menggunakan Metode *ResNet*				
8	2021	Bharati et al. [70]	Mengusulkan algorima baru yaitu *Optimized Residual Network for COVID-19* (*CO-ResNet*)	➢ Jumlah data citra untuk setiap kelas tidak seimbang ➢ Tidak menjelaskan hasil sensitivitas, spesifisitas, dan *F1-Score* sebagai evaluasi kinerja klasifikasi citra.
9	2020	Wu et al. [113]	Menggunakan metode *ResNet* berbasis *multi-view fusion*	➢ Nilai evaluasi kinerja masih kurang baik dalam mengklasifikasikan penyakit *COVID-19* karena nilai yang diperoleh rata-rata masih dibawah 82%. ➢ Tidak menghitung nilai evaluasi kinerja lainnya seperti *F1-Score*
10	2021	Shah et al. [114]	Menggunakan metode *Ctnet-10*, *Dense-169*, *VGG-16*, *ResNet-50*, *Inception-v3*, dan *VGG-19*	Nilai akurasi untuk metode *Inception-v3* dan *ResNet* masih kurang baik mengklasifikasikan penyakit COVID-19 karena nilai akurasinya dibawah 60%. Selain itu nilai evaluasi kinerja lainnya seperti sensitivitas, spesifisitas, *F1-Score* tidak dihitung.
Klasifikasi menggunakan Metode *Inception-v3*				
11	2021	Albahli et al. [115]	Menggunakan metode *Densenet*, *Inception-v3*, dan *Inception-ResNetv4*	Hanya menghitung nilai akurasi dengan menghasilkan nilai masih dibawah 90%. Selain itu tidak menghitung nilai evaluasi kinerja lainnya seperti sensitivitas, spesifisitas, dan *F1-Score*.
12	2021	Dimas et al. [116]	Membandingkan hasil evaluasi kinerja antara metode CNN dengan *Inception-v3* dalam mengklasifikasikan citra CXR ke dalam dua kelas yaitu positif *COVID-19* dan normal	Hasil penelitian menunjukkan bahwa model *Inception-v3* mencapai nilai akurasi, sensitivitas lebih baik dari pada model CNN masing-masing sebesar 82,41%; 76,24%, sedangkan pada model CNN masing-masing sebesar 80,90%; 72,28%. Selain itu tidak menjelaskan hasil spesifisitas, dan *F1-Score* sebagai evaluasi.

No	Tahun	Penulis	Metode	Hasil dan Kelemahan
Klasifikasi menggunakan Metode *EfficientNet*				
13	2021	Monshi et al. [60]	Menerapkan model *CovidXrayNet* yang didasarkan dari arsitektur *EfficientNet-B0* dalam mengklasifikasikan citra CXR	Hasil kinerja metode yang diusulkan untuk dataset *COVIDcxr* nilai akurasi dan *F1-Score* masing-masing sebesar 88.54% dan 88.062%. Selain itu tidak menjelaskan hasil spesififsitas dalam nilai evaluasi.
14	2020	Anwar & Zakir [117]	Menggunakan metode *Efficient-Net* dengan 3 *learning rate* yaitu *Plateau*, *Cyclic*, dan *Constant*	Hasil kinerja evaluasi yang diperoleh adalah nilai akurasi, *F-1 Score* masih dibawah 90%. Selain itu tidak menghitung nilai evaluasi lainnya seperti sensitivitas, dan spesifisitas.
Klasifikasi menggunakan Metode *Ensemble Learning*				
15	2021	Paladini et al. [25]	Membandingkan dua pendekatan metode *ensemble* yaitu *Mean-Ensemble-CNNs* dan *NN-Esemble-CNN* untuk mengklasifikasikan tipe jaringan *Colorectal Cancer* (CRC). Model CNN yang digunakan yaitu *ResNet101*, *ResNeXt50*, *Inception-v3*, dan *DenseNet161*	Hasil nilai akurasi dan *F1-Score* menggunakan *Mean-Ensemble-CNNs* yaitu 86,97% dan 86,99%; sedangkan menggunakan *NN-Esemble-CNN* sebesar 87,26% dan 87,27%. Tidak menjelaskan hasil sensitivitas, dan spesifisitas sebagai evaluasi kinerja. Selain itu, *ensemble* yang dilakukan hanya menggabungkan hasil dari klasifikasi tunggal tanpa dilakukan *training*.
16	2021	Pratiwi et al. [118]	Menerapkan metode *ensemble* menggunakan tiga arsitektur *DCNNs* diantaranya *Inception-v3*, Inception *ResNetV2*, dan *DenseNet201* untuk klasifikasi lesi kulit	Hasil nilai akurasi, sensitivitas, spesifisitas, dan *F1-Score* berturut-turut sebesar 97,23%; 90,12%; 97,73%; dan 85,01. *Ensemble* yang dilakukan hanya menggabungkan hasil dari klasifikasi tunggal tanpa dilakukan *training*.
17	2021	Tang et al. [119]	Menggunakan metode *Ensemble Deep Learning*	Menghasilkan nilai akurasi sebesar 97,8%, presisi sebesar 97,83%, sensitivitas sebesar 97,83, dan *F1- score* sebesar 97,77%. *Ensemble* yang dilakukan hanya menggabungkan hasil dari klasifikasi tunggal tanpa dilakukan *training*.
18	2021	Shrivastava et al. [120]	Menggunakan metode kombinasi *DCNN Ensemble*	Menghasilkan nilai akurasi sebesar 97,47% dan sensitivitas sebesar 98,18%. Tidak menghitung nilai spesifisitas dan *F1-Score*. Selain itu, *ensemble* yang dilakukan hanya menggabungkan hasil dari klasifikasi tunggal tanpa dilakukan *training*.

No	Tahun	Penulis	Metode	Hasil dan Kelemahan
19	2022	Ho & Gwak [121]	Menggunakan metode kombinasi antara *Ensemble*, *ResNet18* dan *DenseNet121*	Menghasilkan nilai akurasi sebesar 94,1%, presisi sebesar 94,5%, sensitivitas sebesar 94,1%, dan *F1- score* sebesar 94%. *Ensemble* yang dilakukan hanya menggabungkan hasil dari klasifikasi tunggal tanpa dilakukan *training*.
20	2022	Visuna et al. [122]	Menggunakan kombinasi metode CNN dan *ensemble learning*	Menghasilkan nilai akurasi sebesar 98%, presisi sebesar 98%, sensitivitas sebesar 98,25%, dan *F1-score* sebesar 97,75%. *Ensemble* yang dilakukan hanya menggabungkan hasil dari klasifikasi tunggal tanpa dilakukan *training*.
21	2023	Kesuma et al. [123][2]	Menggunakan metode *Ensemble Learning* dari *ResNet*, *EfficientNet*, dan *Inception-v3* (ELREI)	Menghasilkan nilai akurasi 99%, presisi 98,75%, *recall* 98,75%, dan *F1-Score* 99%. Kelebihan dari penelitian ini yaitu Metode ELREI bekerja pada setiap *epoch* pada tahap *training*, bukan pada hasil akhir *training*.

[2] Kesuma, L. I., Ermatita, & Erwin. (2023). ELREI : Ensemble Learning of ResNet , EfficientNet , and Inception-v3 for Lung Disease Classification based on Chest X-Ray Image. *International Journal of Intelligent Engineering and Systems*, *16*(5), 149–161. (Lampiran 2)