BAB 3 ANALISIS DAN PERANCANGAN SISTEM

3.1 Analisis Masalah

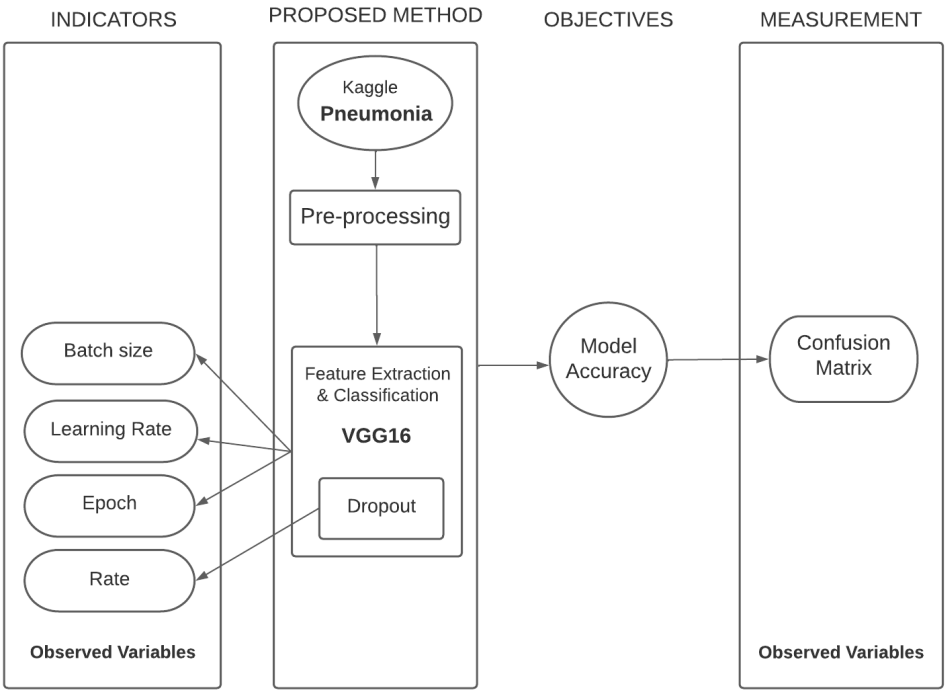
Seperti yang sudah dijelaskan pada bab 1, beberapa penelitian untuk klasifikasi pneumonia sebelumnya telah menggunakan beberapa metode *machine learning* , tetapi metode dengan akurasi terbaik adalah CNN dengan arsitektur VGG16 yang mencapai 94%. Metode VGG16 ini memiliki banyak kelebihan dan mendapatkan akurasi tertinggi daripada metode lainnya. Tetapi metode ini rentan terhadap *overfit*, karena itu diperlukan *dropout* dan *image enhancement* untuk mengurangi kondisi *overfit* dan meningkatkan akurasi.

Penelitian ini akan membangun dan menguji model VGG16 serta melihat seberapa besar pengaruh dari penggunaan *image enhancement* dan *dropout layer* terhadap akurasi dalam kasus klasifikasi pneumonia. Metode *image enhancement* yang akan digunakan pada penelitian ini adalah CLAHE.

Citra masukan dan arsitektur VGG16 akan mengikuti dari penelitian [4] yang merupakan penelitian dengan tingkat akurasi terbaik, yakni 94% dengan ukuran masukan berdimensi 256 x 256 yang didukung dengan penelitian [19] dan [21] yang membandingkan beberapa ukuran citra dan mendapatkan kesimpulan bahwa ukuran citra 256x256 merupakan ukuran citra terbaik dengan akurasi tertinggi. Oleh karena itu, penelitian ini akan menggunakan ukuran citra masukan berdimensi 256 x 256 piksel.

3.2 Kerangka Pemikiran

Berikut ini adalah kerangka pemikiran dari metode yang diusulkan untuk membangun sistem klasifikasi pneumonia.



Gambar 3.1 Kerangka Pemikiran

Berdasarkan Gambar 3.1, penelitian ini akan dimulai dengan mengambil dataset *X-ray* pneumonia yang didapatkan dari situs Kaggle. Data ini kemudian akan masuk ke tahap *preprocessing*. Penelitian ini bertujuan untuk melihat hasil akurasi dalam klasifikasi pneumonia dengan VGG16 serta melakukan perbandingkan akurasi apabila menggunakan *image enhancement* dan *dropout*. Berikut ini akan dijelaskan setiap bagian pada kerangka pemikiran.

1. *Indikators* adalah variabel yang memengaruhi akurasi dari sebuah model atau metode. Penelitian ini akan melakukan observasi terhadap beberapa indikator, antara lain:
   (a) *Batch Size* adalah *hyparameter* pada CNN yang berfungsi untuk menentukan jumlah sampel yang akan digunakan dalam satu iterasi proses *training*. Jumlah *batch size* ini tidak boleh melebihi jumlah sampel yang ada di data *training*. Dengan adanya *batch size*, proses *training* dapat lebih efisien karena hanya membutuhkan lebih sedikit memori untuk melatih sampel.
   (b) *Learning Rate* adalah salah satu *hyparameter* pada CNN yang berfungsi untuk menentukan ukuran langkah pada setiap iterasi untuk mendapatkan *loss function* yang minimum. Semakin besar *learning rate*, maka ketelitian

model akan berkurang. Sebaliknya, semakin kecil *learning rate*, maka model akan semakin teliti, tetapi proses *training* akan semakin lama.

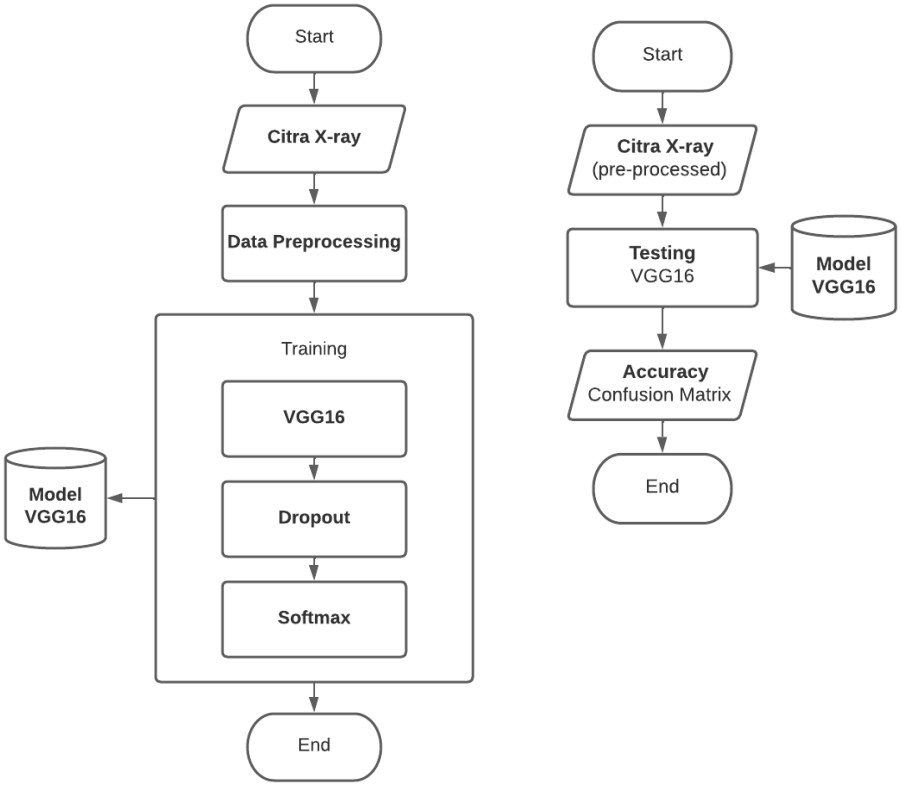
(c) *Epoch* adalah iterasi yang perlu dilakukan pada proses *training*. Satu *epoch* menandakan satu kali iterasi semua data *training* dimasukkan kedalam model. Semakin besar jumlah *epoch* maka akan semakin baik model dalam memahami pola dalam data.

(d) *Rate* adalah salah satu parameter dari *dropout layer* yang mengatur probabilitas sebuah node akan dihilangkan atau dipertahankan pada proses *training* sehingga akan mengurangi kompleksitas dari sebuah jaringan. Jika *rate* yang ditetapkan adalah 1 artinya tidak akan ada node yang dihilangkan, sebaliknya jika *rate* yang ditetapkan adalah 0 maka semua node akan dihilangkan.

2. *Proposed Method* menggambarkan proses serta metode-metode yang digunakan pada penelitian ini. Pada penelitian ini terdapat 3 tahap utama. Pertama, data citra *X-ray* akan diambil sebagai data masukan. Selanjutnya, data citra akan dilakukan *preprocessing* agar lebih banyak informasi yang dapat ditangkap oleh model pada tahap selanjutnya. Tahap *Preprocessing* ini akan melakukan *image resize* dan *image enhancement* menggunakan CLAHE. Setelah dilakukan *preprocessing*, data citra akan dimasukkan ke tahap ekstraksi fitur dan klasifikasi menggunakan VGG16. Untuk mengurangi *overfit*, penelitian ini akan menguji *rate* pada *dropout layer*.
3. *Objectives* menggambarkan target yang akan diukur dari penelitian ini. Penelitian ini akan mengukur akurasi dari model VGG16 yang dibuat dengan atau tanpa *dropout* dan *image enhancement* dalam klasifikasi pneumonia.
4. *Measurement* mendefinisikan metode pengukuran yang akan dilakukan pada penelitian ini sebagai landasan keberhasilan atau tidaknya suatu model. *Measurement* pada penelitian ini akan dilakukan dengan metode *confusion matrix*.

3.3 Urutan Proses Global

Penelitian ini akan dibagi menjadi 2 proses utama, yaitu proses *training* dan proses *testing*. Proses *training* adalah proses dimana model akan dilatih dengan data yang ada untuk menemukan pola, sedangkan *testing* adalah proses untuk pengukuran keberhasilan suatu model. Gambar 3.2 menunjukan urutan proses global pada penelitian ini dalam bentuk *flowchart*.



(a) *Flowchart Training* **(b)** *Flowchart Testing*

Gambar 3.2 *Flowchart* Global

3.3.1 Proses *Training*

Berikut adalah uraian proses *training* yang dilakukan pada penelitian ini sesuai Gambar 3.2(a):

1. Citra masukan merupakan citra *X-ray* dengan 1 *channel* (*grayscale*)
2. Citra akan diubah ukurannya menjadi 256x256 piksel.
3. Data akan dibagi menjadi 90% *train data* dan 10% *test data* melalui metode pada *sklearn* yaitu *train_test_split*. Parameter yang digunakan pada *method train_test_split* yang digunakan adalah *arrays* yang menampung data citra dan label, *test_size* yang bernilai 0.1, *random_state* yang bernilai 42 dan *shuffle* yang bernilai *True*.
4. Kualitas citra akan ditingkatkan menggunakan algoritma CLAHE melalui metode dari pustaka OpenCV yaitu *createCLAHE*. Pada *method createCLAHE* parameter yang digunakan adalah *clip limit* bernilai 1 dan *tile grid size* bernilai 8x8. Penelitian ini menggunakan CLAHE untuk meningkatkan pemahaman model terhadap citra dan meningkatkan akurasi.
5. *Training* menggunakan VGG16

6. Terapkan *dropout layer* dengan menguji *hyparameter rate*. *Dropout layer* digunakan untuk mengurangi kondisi *overfit* yang terjadi pada metode CNN. *Dropout layer* yang digunakan diimpor dari pustaka Keras yaitu *Dropout*. Pada penelitian ini, parameter *Dropout* yang digunakan adalah *rate* dengan nilai 0 (tidak menggunakan dropout) , 0.2, 0.3, 0.4 dan 0.5.
7. Terapkan teknik *Softmax* untuk klasifikasi. Teknik *Softmax* ini dapat dipanggil dari *method Dense* dengan parameter *units* berjumlah 2 dan *activation* berisi *Softmax*,
8. Hasil keluaran dari proses *training* adalah model VGG16 yang disimpan di memori.

Pada penelitian ini akan membandingkan proses *training* menggunakan *image enhancement*, *dropout layer* dengan model yang tidak menggunakannya.

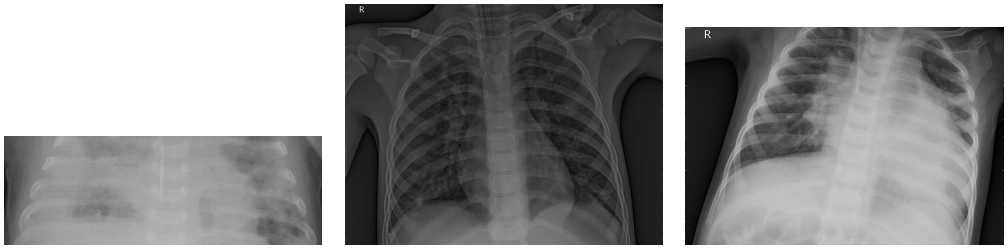
3.3.2 Proses *Testing*

Berikut adalah uraian proses *testing* yang dilakukan pada penelitian ini sesuai Gambar 3.2(b) :

1. Citra masukan merupakan citra *X-ray* yang telah dilakukan pengubahan ukuran menjadi 256x256 serta operasi CLAHE.
2. Dari citra tersebut, akan dilakukan tahap *testing* dengan model VGG16 yang didapatkan dari tahap *training*.
3. Setelah didapatkan hasil klasifikasi dari proses *testing*, model akan di evaluasi dengan menggunakan *confusion matrix* dan *classification report*. *Confusion matrix* dipanggil dari kelas *confusion_matrix* dengan parameter *y_true* yang berisi larik label yang benar, *y_pred* yang berisi larik prediksi. Lalu *Classification report* dipanggil dari kelas *classification_report* dengan parameter *y_true* yang berisi larik label, *y_test* yang berisi larik prediksi dan *digits* bernilai 4.

3.4 Analisis Data Sampling

Dataset yang digunakan pada penelitian ini adalah dataset *Chest X-ray Images* (Pneumonia) [22]. Setiap citra pada dataset ini memiliki *bit depth* bernilai 8 atau *greyscale*. Dataset ini akan dibagi menjadi 90% data *train* dan 10% data *test*. Pada penelitian ini, kelas yang digunakan hanya 2 yaitu kelas pneumonia dan kelas normal. Data citra yang digunakan memiliki pemotongan, pencahayaan dan rotasi yang berbeda seperti pada gambar 3.3.



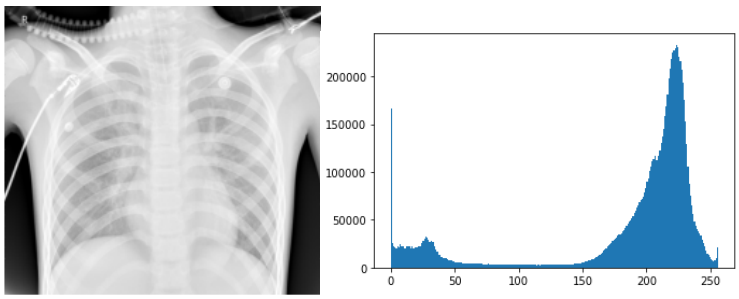
(a) **(b)** **(c)**

Gambar 3.3 (a) Pemotongan tidak konsisten, (b) Kurang pencahayaan, (c) Rotasi

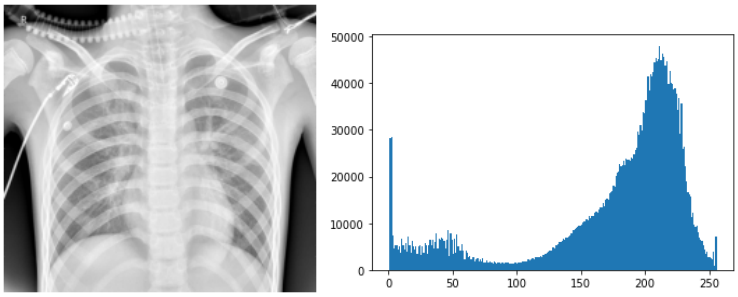
Gambar 3.3 (a) merupakan contoh citra yang mendefinisikan pemotongan citra yang tidak konsisten, dikarenakan citra hanya menunjukkan sedikit bagian dari *X-ray* paru-paru, (b) merupakan contoh citra dengan tingkat pencahayaan yang lebih rendah dari sebagian besar citra yang ada pada dataset, sedangkan (c) merupakan contoh citra *X-ray* paru-paru dengan rotasi yang berbeda dari sebagian besar citra dalam dataset. Dapat dilihat dari Gambar 3.3. Selain itu, berdasarkan gambar 3.3 (a) sampai (c) dapat dilihat setiap citra yang yang terdapat pada dataset memiliki ukuran yang berbeda-beda.

3.5 *Preprocessing*

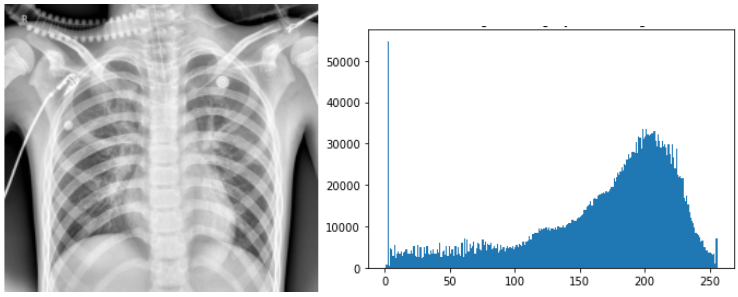
Penelitian ini menggunakan 3 *preprocessing* yang dilakukan yaitu *Image resize* ini digunakan untuk mengubah ukuran gambar menjadi ukuran 256x256 untuk mempermudah proses *training*. Kemudian akan dilakukan *image enhancement* dengan menggunakan algoritme CLAHE sehingga dapat meningkatkan kualitas citra untuk mempermudah analisis pola citra.



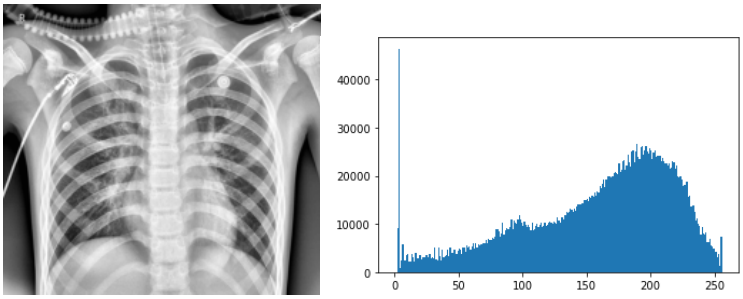
Gambar 3.4 Tanpa CLAHE



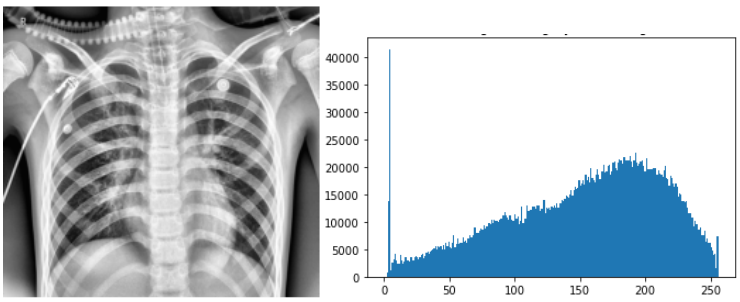
Gambar 3.5 CLAHE, dengan *clip limit = 1*



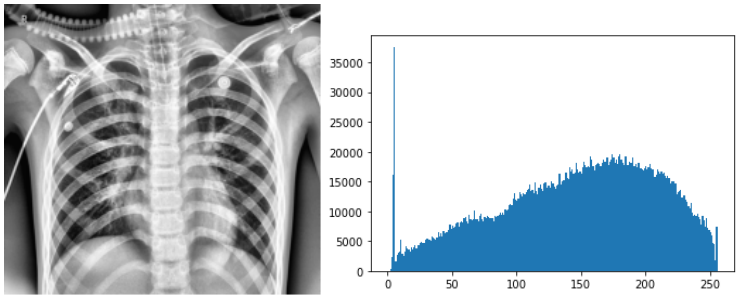
Gambar 3.6 CLAHE, dengan *clip limit = 2*



Gambar 3.7 CLAHE, dengan *clip limit = 3*



Gambar 3.8 CLAHE, dengan *clip limit = 4*



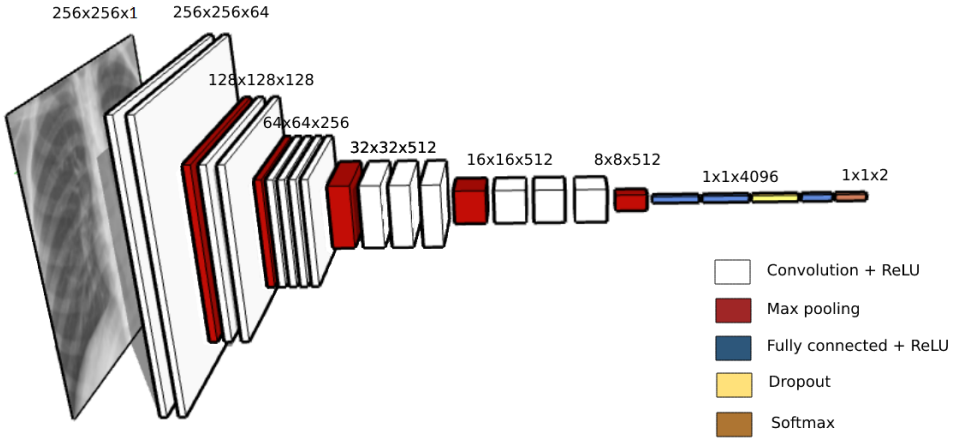
Gambar 3.9 CLAHE, dengan *clip limit = 5*

Dari gambar 3.4 sampai dengan 3.9 dapat dilihat perubahan citra serta histogram tanpa CLAHE dan dengan menggunakan CLAHE. Histogram citra asal memiliki distribusi *left skew* yang tersebar pada nilai *grayscale* 0 sampai 250 sehingga memiliki citra dengan tampilan kecerahan yang lebih tinggi. Pada CLAHE terdapat parameter *clip limit* yang mengatur tingkat pemotongan histogram. Semakin tinggi *clip limit* yang digunakan, maka semakin tinggi kontras yang dihasilkan.

Pada citra asli dapat dilihat bahwa citra paru-paru pneumonia memiliki tanda bayangan putih dengan tingkat kecerahan tinggi yang menutupi paru-paru sehingga paru-paru kurang terlihat dengan jelas sedangkan pada citra dengan *clip limit* bernilai 1 dapat dilihat bahwa bayangan putih yang menutupi paru-paru sedikit memudar karena adanya pertambahan kontras. Pada citra dengan *clip limit* bernilai 2 sampai 5, bayangan putih yang menutupi paru-paru mulai hilang seiring pertambahan *clip limit* sehingga akan mungkin citra tersebut mungkin akan di identifikasi sebagai citra paru-paru normal karena kehilangan fitur penting yaitu bayangan putih yang ada pada citra.

Pada penelitian ini, *clip limit* yang akan digunakan bernilai 1 dikarenakan nilai *clip limit* yang tinggi akan mengurangi fitur penting dalam suatu gambar sehingga dapat menyebabkan kesalahan pembelajaran model.

3.6 *Analisis VGG16*



Gambar 3.10 *Arsitektur VGG16 yang digunakan*

Penelitian ini akan menggunakan arsitektur VGG16 yang setiap lapisannya akan di impor dari pustaka Keras. Pada penelitian ini, citra masukan akan menggunakan ukuran 256 x 256 piksel sehingga keluaran yang dikeluarkan setiap lapisan akan berbeda dengan arsitektur VGG16 yang mengacu pada Bab 2. Ukuran *filter* yang digunakan akan mengacu pada jurnal [4] dengan ukuran *filter* adalah 64 sampai 512. Warna kuning pada Tabel 3.1 menggambarkan perbedaan arsitektur VGG16 yang akan digunakan serta arsitektur VGG16 biasa. Arsitektur VGG16 yang akan digunakan terdiri dari 13 *convolutional layer*, 4 *pooling layer*, 3 *fully connected layer* dan 1 *dropout layer* yang disusun sebagai berikut:

Tabel 3.1 Proses pada VGG16

Blok	***Layer***	**Masukan**	**Keluaran**
1	*Convolutional Layer*	256x256x1	256x256 x64
	Convolutional Layer	256x256 x64	256x256 x64
	Max Pooling Layer	256x256 x64	128x128 x64
2	*Convolutional Layer*	128x128 x64	128x128 x128
	Convolutional Layer	128x128 x128	128x128 x128
	Max Pooling Layer	128x128 x128	64x64 x128
3	*Convolutional Layer*	64x64 x128	64x64 x256
	Convolutional Layer	64x64 x256	64x64 x256

Blok	***Layer***	**Masukan**	**Keluaran**
	Convolutional Layer	64x64 x256	64x64 x256
	Max Pooling Layer	64x64 4x256	32x32 x256
4	*Convolutional Layer*	32x32 x256	32x32 x512
	Convolutional Layer	32x32 x512	32x32 x512
	Convolutional Layer	32x32 x512	32x32 x512
	Max Pooling Layer	32x32 x512	16x16 x512
5	*Convolutional Layer*	16x16 x512	16x16 x512
	Convolutional Layer	16x16 x512	16x16 x512
	Convolutional Layer	16x16 x512	16x16 x512
	Max Pooling Layer	16x16 x512	8x8 x512
6	*Flatten*	32768	32768
	Fully Connected Layer	4096	4096
	Fully Connected Layer	4096	4096
	Dropout Layer	4096	4096
	Fully Connected Layer	4096	2

Sesuai dengan tabel 3.1, penelitian ini akan dimulai dengan uraian proses sebagai berikut :

1. Data citra *grayscale* akan dimasukkan ke dalam *convolution layer* dengan ukuran 256x256 piksel. Citra akan disimpan berupa data numerik pada bentuk matriks.
2. Pada *convolutional layer*, matriks masukan akan diberi padding bernilai 1. Pada lapisan ini, *kernel* yang digunakan berdimensi 3x3 dengan nilai yang dihasilkan dengan persamaan 2.4. *Convolutional layer* dipanggil dari kelas *Conv2D* dengan parameter yang *filters* yang bernilai dari 64 sampai 512, *kernel_size* yang bernilai 3, *padding* yang berisi *same* dan *activation* yang berisi relu.
3. Operasi konvolusi akan dilakukan pada *convolutional layer* yang menghasilkan sebuah *feature map* dengan *stride* bernilai 1.
4. *Feature map* yang dihasilkan akan melalui fungsi aktivasi ReLU sesuai dengan persamaan 2.7.

   Pada *pooling layer*, operasi *Max Pooling* akan digunakan dengan *spatial extent* 2x2 dan *stride* bernilai 2. Dimensi keluaran dari *Max Pooling layer* dapat dihitung dengan persamaan 2.5. Lapisan *Max Pooling* dapat dipanggil dengan

menggunakan kelas *MaxPool2D* dari pustaka Keras yang menggunakan parameter *pool size* bernilai 2 dan *stride* bernilai 2.

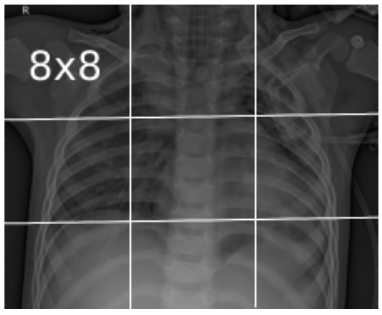
6. Proses ini akan dilakukan berulang kali sampai dengan blok ke 5 sesuai dengan Tabel 3.1.
7. Pada *fully connected layer*, keluaran *feature map* akan diubah menjadi larik 1 dimensi dan melakukan perhitungan sesuai dengan persamaan 2.6. Keluaran yang dihasilkan akan melewati fungsi aktivasi ReLU. Proses ini akan berulang sebanyak 2 kali. *Fully connected layer* akan menggunakan kelas *dense* dari pustaka Keras dengan menggunakan parameter *units* bernilai 4096 dan fungsi aktivasi Relu.
8. Pada *dropout layer*, beberapa nilai keluaran dari *fully connected layer* akan dinonaktifkan secara acak sehingga dapat mencegah *overfit*. *Dropout layer* akan menggunakan kelas *dropout* dari pustaka Keras dan menguji parameter *rate*nya.
9. Keluaran yang dihasilkan dari *dropout layer* akan dimasukkan ke *fully connected layer* untuk klasifikasi dengan menggunakan fungsi aktivasi *Softmax* sesuai dengan persamaan 2.8. Proses klasifikasi ini akan menggunakan kelas *dense* dari pustaka Keras dengan *units* bernilai 2 dan fungsi aktivasi *Softmax*.

3.7 Analisis Manual

Pada bagian ini, akan dilakukan tahapan proses beserta contoh dan ilutrasi perhitungan manual yang dilakukan dalam sistem.

3.7.1 CLAHE

CLAHE adalah salah satu teknik *image enhancement* yang memiliki *clip limit* dimana berfungsi untuk menahan suatu histogram agar citra tidak terlalu terang. Penerapan CLAHE akan menggunakan *method createCLAHE* yang diambil dari pustaka OpenCV dengan memodifikasi parameter *clip limit* dan *tile grid size*. Pada operasi CLAHE, citra masukan akan dibagi menjadi blok-blok subcitra sesuai dengan parameter *tile grid size* yaitu 8x8.



Gambar 3.11 Contoh pembagian blok-blok subcitra

Setiap blok subcitra ini nantinya akan dilakukan perhitungan PDF. Lalu dari nilai PDF tersebut akan ditetapkan parameter *clip limit* yang mengatur batas maksimum dari suatu frekuensi histogram. Jika suatu frekuensi melebihi nilai *clip limit*, maka nilai frekuensi tersebut akan dipotong dan nilai lebihnya akan ditampung. Pada penelitian ini, parameter *clip limit* yang digunakan adalah 1. Batas histogram dapat dihitung dengan persamaan 2.2.

$$\begin{aligned} N_{clip} &= max(1, \frac{N_{CL} \times N_x \times N_y}{N_{graylevel}} \times N_{maxfreq}) \\ &= max(1, \frac{1 \times 8 \times 8}{256} \times 10) \\ &= 2.5 \end{aligned}$$

Setelah didapatkan nilai lebih dari hasil pemotongan histogram, nilai tersebut akan dibagi secara merata ke seluruh nilai histogram pada blok tersebut lalu akan dihitung nilai CDFnya. Lalu dari nilai CDF, akan dilakukan *histogram equalization* sesuai dengan persamaan 2.1. Tahapan ini akan terus berulang sampai seluruh blok subcitra selesai diproses.

3.7.2 VGG16

Arsitektur yang digunakan pada penelitian ini adalah VGG16. Data citra akan dibagi menjadi 90% data *training* dan 10% data *testing* sehingga data *training* berjumlah 5270 dan data *testing* berjumlah 586. Selama proses *training*, citra masukan yang digunakan adalah 256x256 piksel dan sudah melalui *image enhancement CLAHE*. Setiap piksel pada citra masukan merepresentasikan nilai dari 0 sampai 255 yang membentuk sebuah matriks seperti pada gambar 3.12.

130	143	155	165	173	105
122	136	148	159	168	93
111	128	140	152	162	60
98	120	132	145	155	145
80	100	135	150	167	183
182	150	88	56	78	190

Gambar 3.12 Contoh matriks masukan

Matriks citra masukan tersebut akan digunakan pada masukan untuk *convolutional layer*. Pada *convolutional layer*, matriks masukan diberi *padding* berukuran satu seperti pada gambar 3.13 (a). *Padding* ini digunakan pada citra agar menghasilkan keluaran dengan dimensi sama dengan masukan. Pada lapisan ini, *kernel* yang digunakan berdimensi 3x3 dan nilainya akan diinisialisasikan dengan *glorot uniform* secara acak dari persamaan 2.4 seperti berikut:

$$
\begin{aligned}
W &= U\left[-\sqrt{\frac{6}{fan_{in}+fan_{out}}}, \sqrt{\frac{6}{fan_{in}+fan_{out}}}\right] \\
&= U\left[-\sqrt{\frac{6}{(1\times3\times3)+(64\times3\times3)}}, \sqrt{\frac{6}{(1\times3\times3)+(64\times3\times3)}}\right] \\
&= U\left[-\sqrt{\frac{6}{(9)+(576)}}, \sqrt{\frac{6}{(9)+(576)}}\right] \\
&= U\left[-\sqrt{\frac{6}{585}}, \sqrt{\frac{6}{585}}\right] \\
&= U\left[-\sqrt{0.0102}, \sqrt{0.0102}\right] \\
&= U\left[-0.101, 0.101\right]
\end{aligned}
$$

Gambar 3.13 (b) mendefinisikan contoh *kernel* dengan masukan secara acak secara *uniform* dari nilai -0.101 sampai 0.0101. Selanjutnya, *kernel* akan dioperasikan terhadap citra masukan dan menghasilkan *feature map*. Proses operasi ini akan bergerak secara horizontal ke arah kanan dengan nilai *stride* satu.

0	0	0	0	0	0	0	0
0	130	143	155	165	173	105	0
0	122	136	148	159	168	93	0
0	111	128	140	152	162	60	0
0	98	120	132	145	155	145	0
0	80	100	135	150	167	183	0
0	182	150	88	56	78	190	0
0	0	0	0	0	0	0	0

(a) Citra masukan dengan *padding* bernilai satu

0.0019	0.0816	0.0976
-0.0548	-0.0291	-0.0361
-0.0692	-0.0126	0.0186

(b) *Kernel* 3x3

Gambar 3.13 Contoh citra masukan dan *kernel*

Proses operasi konvolusi ini akan melakukan perkalian *kernel* terhadap seluruh citra masukan sesuai dengan persamaan 2.3.

$$\begin{aligned} v_k &= w_{k_{0,0}}x_{0,0} + w_{k_{0,1}}x_{0,1} + w_{k_{0,2}}x_{0,2} \\ &+ w_{k_{1,0}}x_{1,0} + w_{k_{1,1}}x_{1,1} + w_{k_{1,2}}x_{1,2} \\ &+ w_{k_{2,0}}x_{2,0} + w_{k_{2,1}}x_{2,1} + w_{k_{2,2}}x_{2,2} \\ &= 0.0019 * 0 + 0.0816 * 0 + 0.0976 * 0 \\ &+ (-0.0548) * 0 + (-0.0291) * 130 + (-0.0361) * 143 \\ &+ (-0.0692) * 0 + (-0.0126) * 122 + 0.0186 * 136 \\ &= -7.95.. \end{aligned}$$

Operasi konvolusi akan menghasilkan sebuah *feature map*. Ukuran *feature map* yang dihasilkan ini akan berubah sesuai dengan ukuran *padding* yang diaplikasikan pada citra masukan. Pada penelitian ini, keluaran dari operasi konvolusi yang dilakukan adalah *feature map* dengan dimensi yang sama seperti citra masukan yang digambarkan pada seperti pada Gambar 3.14.

-7.95	-24.28	-26.61	-28.65	-29.25	-25.33
17.09	4.37	3.74	3.26	-3.71	-14.97
16.37	5.08	4.01	3.64	-1.41	-15.26
15.22	6.41	5.03	3.31	-7.40	-21.36
14.27	-2.15	-0.11	4.08	6.07	-10.13
5.58	3.98	13.05	19.53	19.58	5.46

Gambar 3.14 Contoh keluaran operasi konvolusi

Setelah didapatkan *feature map*, *feature map* ini akan melalui fungsi aktivasi nonlinier. Pada penelitian ini fungsi aktivasi yang digunakan adalah ReLU sesuai dengan persamaan 2.7. Fungsi ini akan mengubah nilai kurang dari 0 menjadi nilai 0 dan nilainya dipertahankan jika nilainya diatas dari 0. Hasil *feature map* yang dihasilkan setelah melalui ReLU digambarkan sebagai berikut.

0.00	0.00	0.00	0.00	0.00	0.00
17.09	4.37	3.74	3.26	0.00	0.00
16.37	5.08	4.01	3.64	0.00	0.00
15.22	6.41	5.03	3.31	0.00	0.00
14.27	0.00	0.00	4.08	6.07	0.00
5.58	3.98	13.05	19.53	19.58	5.46

Gambar 3.15 *feature map* setelah melalui ReLU

Langkah selanjutnya adalah melakukan pengurangan dimensi data untuk mengurangi komputasi dalam melatih data. Pengurangan dimensi ini dapat dilakukan dengan *pooling layer*. Pada penelitian ini, operasi *Max Pooling* akan digunakan dalam mengurangi dimensi dimana akan mengambil nilai maksimum dari suatu area pooling. *Max Pooling* ini digunakan untuk mengurangi waktu *training* dan menyimpan informasi yang penting. Penelitian ini juga akan menggunakan *spatial extent* 2x2 dengan *stride* bernilai 2. Dimensi keluaran dari operasi ini dapat dihitung melalui persamaan 2.5.

$$h^1 = \frac{6-2+2}{2} = 3$$
$$w^1 = \frac{6-2+2}{2} = 3$$

Dari persamaan 2.5, didapatkan dimensi keluaran yang dihasilkan pada *pooling layer* adalah 3x3 piksel dengan hasil perhitungan *Max Pooling* sebagai berikut.

17.09	3.74	0.00
16.37	5.03	0.00
14.27	19.53	19.58

Gambar 3.16 Keluaran Max Pooling

Setelah blok ke 5 selesai di proses akan didapatkan sebuah keluaran *feature map*. Keluaran ini akan diproses dengan *flatten* agar menjadi larik 1 dimensi sehingga dapat memudahkan klasifikasi. Nantinya akan terdapat 1 larik dengan 32768 buah nilai. Selanjutnya, larik tersebut akan dimasukkan pada *fully connected layer* dan melakukan inisialisasi *kernel* untuk lapisan ini. Pada *fully connected layer* ini, akan dilakukan inisialisasi *kernel* sebanyak 4096 kali, sehingga nantinya akan terdapat 1 larik berisi 4096 nilai.

17.09	3.74	0.00	16.37	5.03	0.00	14.27	19.53	19.58	...

Gambar 3.17 *Flattened Feature Map*

-0.0090	-0.0086	0.0002	-0.0017	-0.0113	-0.0050	-0.0100	0.0100	0.0007	...

Gambar 3.18 Contoh *kernel* pada *fully connected layer*

Dari citra masukan dan *kernel* yang ada di *fully connected layer*, akan dilakukan perhitungan sesuai dengan persamaan 2.6 dengan asumsi bias bernilai 0.

$$\begin{aligned} y &= f(W^T x + b) \\ &= f((-0.0090) * 17.09 + (-0.0086) * 3.74 + 0.0002 * 0 + ... + 0) \\ &= f(-0,2032) \end{aligned}$$

Operasi ini akan dilakukan berkali-kali dengan nilai *kernel* yang berbeda-beda sehingga mendapatkan keluaran. Hasil keluaran dari *fully connected layer* akan melewati fungsi aktivasi ReLU sesuai dengan persamaan 2.7. Proses ini akan berlangsung sebanyak 2 kali.

0	0	3.25751	0	0	1.45028	0	2.16853	0	...

Gambar 3.19 Contoh keluaran yang dihasilkan pada *fully connected layer*

Keluaran yang dihasilkan dari *fully connected layer* akan digunakan sebagai masukan pada *dropout layer*. *Dropout layer* ini akan menonaktifkan beberapa nilai fitur secara acak sehingga dapat mencegah *overfit*. Pada proses *training*, jika nilainya tidak dinonaktifkan, nilainya akan diperbesar menggunakan persamaan 2.10 sebagai berikut:

$$\begin{aligned} Scaled &= \frac{1}{1-p} \times w \\ &= \frac{1}{1-0.5} \times 3.25751 \\ &= 6.515.. \end{aligned}$$

0	0	6.515	0	0	2.9006	0	0	0	...

Gambar 3.20 Contoh keluaran yang dihasilkan pada *dropout layer*.

Dari Gambar 3.22 dapat dilihat terdapat beberapa nilai yang dinonaktifkan dengan operasi *dropout* (berwarna merah). Dengan nilai dropout rate 0.5 artinya setiap neuron memiliki 50% kesempatan untuk dinonaktifkan secara acak. Kemudian, *feature map* tersebut akan dimasukkan ke dalam lapisan terakhir *fully connected layer* yang kemudian akan dikalikan dengan 2 buah kernel yang menghasilkan larik dengan 2 nilai.

0.1717	-0.077

Gambar 3.21 Contoh keluaran yang dihasilkan pada *fully connected layer* terakhir.

Hasil tersebut kemudian akan menjadi masukan untuk fungsi aktivasi *Softmax* dan akan dihitung dengan persamaan 2.8. Perhitungan ini dilakukan untuk mencari probabilitas setiap kelas yang telah didefinisikan sebelumnya. Pada penelitian ini, terdapat 2 buah kelas yaitu kelas normal dan kelas pneumonia. Berikut adalah contoh perhitungan fungsi aktivasi *Softmax*.

$$\begin{aligned}\Phi i &= \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \\ &= \frac{e^{0.1717}}{(e^{0.1717} + e^{-0.077})} \\ &= 0.5619\end{aligned}$$

Hasil akhir perhitungan adalah probabilitas setiap kelas yang terdeteksi pada dataset terkait. Nilai probabilitas tertinggi akan diterima sebagai hasil estimasi. Berikut ini adalah contoh hasil keluaran *Softmax* dengan mengambil kelas ke-1 (nilai probabilitas 0.5619) sebagai hasilnya.

0.5619	0.4381

Gambar 3.22 Contoh keluaran *Softmax* yang dihasilkan

Setelah proses *training* dan *testing* selesai, akan didapatkan nilai akurasi serta nilai *loss* untuk data *train* dan *test*. Untuk melihat pengaruh *dropout* terhadap *overfit*, *generalization gap* akan dihitung dengan persamaan 2.12. Berikut adalah contoh perhitungan *generalization gap* dengan asumsi *training loss* bernilai 4.80% dan *testing loss* bernilai 11.29%:

$$\begin{aligned} gen(f) &= R(f) - R_s(f) \\ &= 11.29\% - 4.80\% \\ &= 6.49\% \end{aligned}$$