BAB 3

METODOLOGI PENELITIAN DAN PERANCANGAN APLIKASI

3.1 Metodologi Penelitian

Metodologi penelitian yang digunakan dalam implementasi CNN guna mengetahui pengaruh jumlah *hidden layer* dalam klasifikasi penyakit pneumonia yaitu studi literatur, pembagian gambar, perancangan dan pembuatan aplikasi, training-testing, dan evaluasi.

a. Studi Literatur

Dalam studi literatur, pembelajaran terhadap teori-teori yang dapat mendukung proses perancangan machine learning dalam mengoptimasi proses pengklasifikasian penyakit pneumonia. Teori-teori yang bersangkutan adalah pneumonia akibat bakteri/virus, *Convolutional Neural Network* (CNN) - VGG-16, *Hidden Layer, Data Splitter*, Metode Topsis, dan *Fine Tuning*.

b. Pembagian Gambar (Image Splitter)

Pada pembagian gambar terdiri dari dua jenis pembagian, yaitu pembagian perbandingan gambar sebagai data train - data test dan pembagian gambar paru paru normal, pneumonia akibat bakteri, dan pneumonia akibat virus. Pembagian gambar data *train* – data *test* dibagi dengan perbandingan 8:2 untuk masing masing gambar paru paru normal, pneumonia akibat bakteri, serta pneumonia akibat virus. Selain itu diambil juga 8 sampel untuk paru paru normal sertau 8 sampel untuk paru paru yang terkena pneumonia untuk dijadikan gambar validasi.

c. Perancangan dan Pembuatan Aplikasi

Pada tahap ini, *data sample* yang berupa gambar berformat .jpeg hitam putih akan melewati proses *image recognition* terlebih dahulu dengan penyamaan ukuran dan warna dari hasil *x-ray* yang didapatkan (hal ini guna menghindari perbedaan kualitas gambar dari *data sample* yang ada). Tahapan selanjutnya yang dilakukan adalah mengimplementasikan model VGG16 dan algoritma *fine tuning.* Model VGG-16 dibuat dengan menggunakan bahasa pemrograman Python.

d. Training – Testing

Training dilakukan untuk membangun pengetahuan *machine learning* mengenai klasifikasi penyakit pneumonia, sedangkan *testing* digunakan untuk mengukur kepintaran dari *machine learning* dalam mengklasifikasikan penyakit pneumonia. Proses ini dilakukan dengan 20 *Epoch* untuk setiap jumlah layernya. Jumlah *layer* yang akan diuji akan digandakan untuk setiap pengujiannya (Contoh: 128, 128*2, 256, 256*2, 512, 512*2). Dalam setiap pengujian, faktor uji yang akan diamati adalah kecepatan, akurasi, dan *loss* (nilai *loss function*).

e. Evaluasi

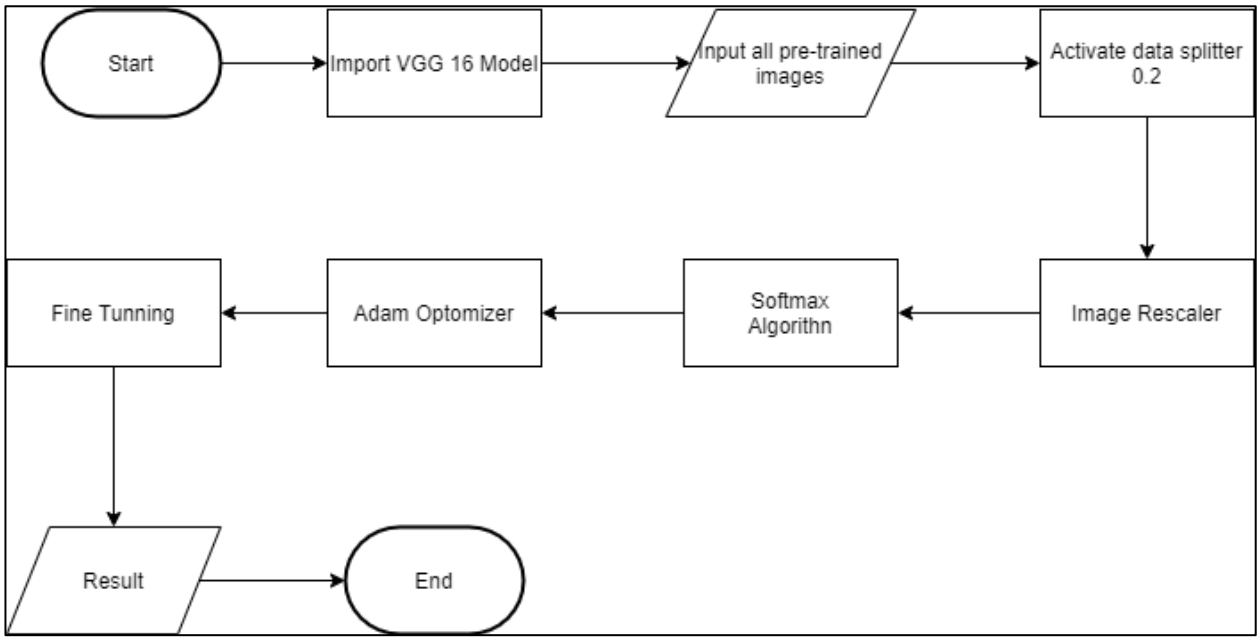
Evaluasi dengan menggunakan metode TOPSIS (Technique for Order of Preference by Similarity to Ideal Solution) pada waktu, *loss*, dan akurasi untuk setiap kedalaman *layer* dan memberikan peringkat / *ranking* untuk setiap *hidden layer* yang digunakan. Digunakannya metode TOPSIS karena metode ini memiliki komputasi yang efisien serta cocok sebagai pengukur kinerja alternatif dan juga alternatif keputusan keputusan dalam sebuah bentuk output komputasi yang sederhana.

3.2 Perancangan Aplikasi

Aplikasi dirancang dengan menggunakan flowchart untuk menunjukkan alur program. Flowchart dari aplikasi ini adalah sebagai berikut.

3.2.1 Flowchart Umum

Gambar 3.1 menunjukkan flowchart proses keseluruhan aplikasi. Proses dimulai dengan *import* model VGG 16, kemudian memasukkan semua gambar / *dataset* yang telah dibagi kedalam tiga *class* yaitu paru paru normal, pneumonia akibat bakteri, dan pneumonia akibat virus yang berjumlah 5216 gambar. Setelah itu gambar / *dataset* tersebut dibagi lagi kedalam data train dan data test yang memiliki perbandingan 8:2 untuk masing masing *class* yang terjadi dalam proses data splitter 0.2. Setelah itu, dilakukan proses-proses seperti *image rescaler*, *softmax algorithm*, *Adam optimizer*, dan *fine tuning*.



Gambar 3. 1 Flowchart Proses Sistem

3.2.2 Proses Image Rescaler

Proses selanjutnya adalah proses *Image rescaler* yang digunakan untuk menyamakan ukuran setiap gambar / data X-ray yang digunakan. Pada setiap gambar X-ray paru paru terdapat tempat / titik yang memiliki kemiripan dengan gambar lainnya, bagian ini yang nantinya akan dijadikan titik poros untuk disejajarkan dengan gambar lainnya dan disesuaikan ukurannya.

3.2.3 Proses Softmax Algorithm

Setelah proses *image rescaler* dilakukan, selanjutnya proses yang akan dijalankan adalah *softmax* yang digunakan untuk menghitung probabilitas. *Softmax algorithm* menghitung probabilitas dari semua kelas / label yang diperoleh setelah menjalankan proses *image rescaler*, kemudian diambil sebuah vektor dengan nilai riil dan mengubahnya menjadi vektor dengan nilai antara nol dan satu yang bila dijumlah akan bernilai satu.

3.2.4 Proses Adam Optimizer

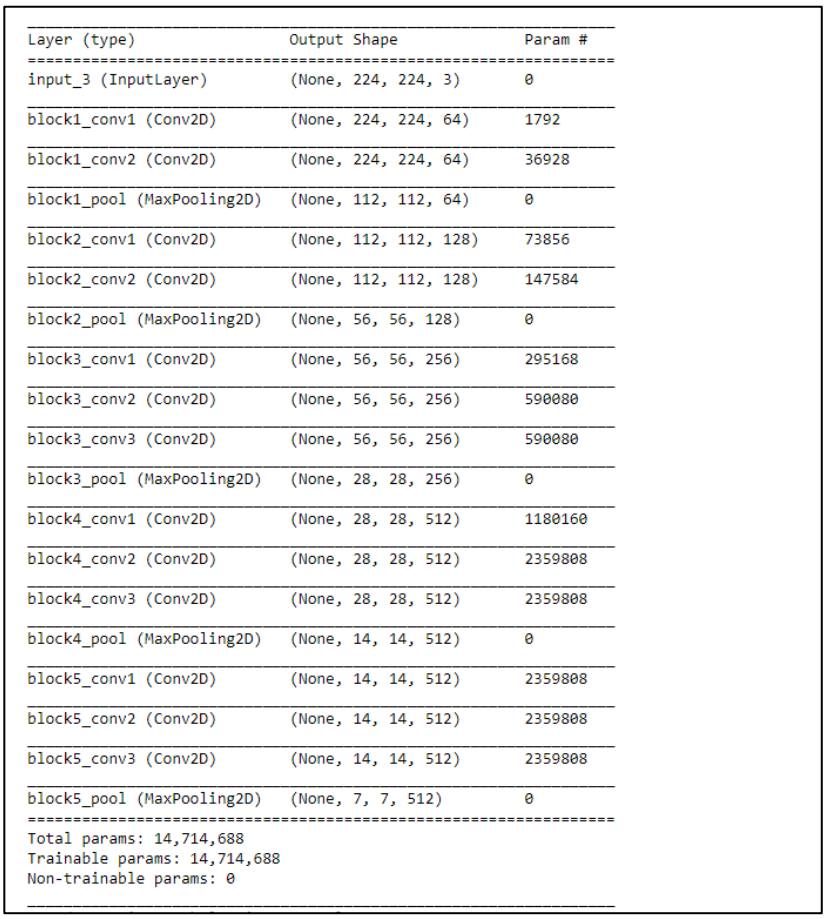
Pada proses ini terdapat algoritma yang berfungsi untuk melakukan optimisasi stokastik berdasarkan perkiraan adaptif dari *lower-order moments*. Algoritma yang ditemukan oleh Kingman dan Ba ini digunakan untuk menangani gradien yang menyebar dan memiliki *noise* pada gambar x-ray paru paru penyakit pneumonia ini.

3.2.5 Proses Fine Tuning

Proses selanjutnya adalah proses *fine tuning* yang digunakan untuk melakukan klasifikasi penyakit pneumonia. Pada proses ini terdapat jaringan *pre-trained* agar sistem tidak perlu melatih seluruh jaringan. Hal ini digunakan agar komputasi yang dilakukan tidak terlalu membebani komputer.

3.2.6 Output Program

Output dari program yaitu berupa tabel yang berisi *layer, output shape* dan jumlah parameter (Gambar 3.2). Tabel ini juga menunjukkan total jumlah parameter yang digunakan baik yang digunakan sebagai acuan *training* maupun yang tidak.



Layer (type)	Output Shape	Param #
input_3 (InputLayer)	(None, 224, 224, 3)	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2359808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0

Total params: 14,714,688
Trainable params: 14,714,688
Non-trainable params: 0

Gambar 3. 2 Tabel Total Parameter

Selain itu terdapat pula jumlah gambar yang digunakan dan hasil untuk setiap *epoch* yang terdiri dari durasi 1 *epoch*, waktu setiap *step*, nilai *loss*, acc, nilai val_loss, dan val_acc (Gambar 3.3).

```
_________________________________________________________________
Found 5216 images belonging to 2 classes.
Found 624 images belonging to 2 classes.
Epoch 1/20
260/260 [==============================] - 45s 171ms/step - loss: 0.1860 - acc: 0.9238 - val_loss: 1.1910 - val_acc: 0.7726
Epoch 2/20
260/260 [==============================] - 43s 166ms/step - loss: 0.0739 - acc: 0.9773 - val_loss: 1.2013 - val_acc: 0.7169
Epoch 3/20
260/260 [==============================] - 43s 167ms/step - loss: 0.0609 - acc: 0.9821 - val_loss: 1.4601 - val_acc: 0.7053
Epoch 4/20
260/260 [==============================] - 43s 164ms/step - loss: 0.0520 - acc: 0.9883 - val_loss: 2.9120 - val_acc: 0.7450
Epoch 5/20
260/260 [==============================] - 43s 166ms/step - loss: 0.0572 - acc: 0.9883 - val_loss: 2.2506 - val_acc: 0.7351
Epoch 6/20
260/260 [==============================] - 43s 166ms/step - loss: 0.0448 - acc: 0.9917 - val_loss: 1.5693 - val_acc: 0.7897
Epoch 7/20
260/260 [==============================] - 43s 165ms/step - loss: 0.0248 - acc: 0.9937 - val_loss: 3.5319 - val_acc: 0.7285
Epoch 8/20
260/260 [==============================] - 43s 164ms/step - loss: 0.0326 - acc: 0.9937 - val_loss: 3.4942 - val_acc: 0.7483
Epoch 9/20
260/260 [==============================] - 43s 166ms/step - loss: 0.0483 - acc: 0.9933 - val_loss: 2.0176 - val_acc: 0.8129
Epoch 10/20
260/260 [==============================] - 43s 164ms/step - loss: 0.0245 - acc: 0.9958 - val_loss: 3.3015 - val_acc: 0.7632
Epoch 11/20
260/260 [==============================] - 43s 165ms/step - loss: 0.0497 - acc: 0.9937 - val_loss: 2.9107 - val_acc: 0.7632
Epoch 12/20
260/260 [==============================] - 43s 164ms/step - loss: 0.0209 - acc: 0.9963 - val_loss: 3.5246 - val_acc: 0.7450
Epoch 13/20
260/260 [==============================] - 43s 165ms/step - loss: 0.0361 - acc: 0.9963 - val_loss: 3.7384 - val_acc: 0.7583
Epoch 14/20
260/260 [==============================] - 43s 164ms/step - loss: 0.0573 - acc: 0.9958 - val_loss: 2.9542 - val_acc: 0.7616
Epoch 15/20
260/260 [==============================] - 42s 163ms/step - loss: 0.0417 - acc: 0.9963 - val_loss: 3.0397 - val_acc: 0.8030
Epoch 16/20
260/260 [==============================] - 43s 164ms/step - loss: 0.0601 - acc: 0.9946 - val_loss: 3.3468 - val_acc: 0.7732
Epoch 17/20
260/260 [==============================] - 43s 164ms/step - loss: 0.0539 - acc: 0.9944 - val_loss: 4.8217 - val_acc: 0.6954
Epoch 18/20
260/260 [==============================] - 43s 164ms/step - loss: 0.0448 - acc: 0.9967 - val_loss: 3.6556 - val_acc: 0.7566
Epoch 19/20
260/260 [==============================] - 42s 163ms/step - loss: 0.0634 - acc: 0.9950 - val_loss: 3.3879 - val_acc: 0.7732
Epoch 20/20
260/260 [==============================] - 43s 164ms/step - loss: 0.0176 - acc: 0.9987 - val_loss: 4.4026 - val_acc: 0.7219
```

Gambar 3. 3 Tabel Akurasi

Pada tabel di atas terdapat 20 *epoch* dan setiap *epoch* memiliki waktu, *loss*, dan akurasinya masing masing. Hasil yang baik akan menunjukkan waktu yang cepat, nilai *loss* yang kecil, dan nilai akurasi yang tinggi. *Range* yang digunakan pada akurasi adalah 0 yang berarti akurasi 0% sampai dengan 1 yang berarti akurasi 100%. Selain itu *range* yang digunakan untuk *loss* juga sama seperti akurasi, namun jika hasil yang diberikan lebih dari 1 maka *range* yang digunakan adalah 0 sampai dengan 250. Hal tersebut terjadi karena fungsi yang digunakan adalah ReLU (*Rectified Linear Unit*) yang memiliki nilai maksimal 250 dan sering terjadi pada *dense layer*.