**BAB III**

# METODOLOGI PENELITIAN

* 1. **Populasi dan Sampel**

Populasi dalam penelitian ini adalah citra hasil rontgen dada (*Chest X-Ray*)*.* Sedangkan sampel yang digunakan dalam penelitian ini adalah dataset citra rontgen dada yaitu norma,l *tuberculosis*(TBC), *pneumonia*, dan COVID-19. Jumlah citra yang digunakan untuk pengujian adalah 1.000 gambar *Chest X-Ray*, dimana dataset tersebut dibagi dalam beberapa rasio untuk digunakan sebagai data train dan validation. Rasio pembagian tersebut adalah 20% : 80%, 50% : 50%, dan 80% : 20%. Hasil dari tiap jumlah yang telah di uji coba akan di analisis bagaimana tingkat akurasi menggunakan tiga model *convolutional Neural Network* (CNN), yaitu VGG19, ResNet50, dan InceptionV3.

* 1. **Variabel dan Definisi Operasional**

Variabel yang digunakan dalam penelitian ini ditampilkan dalam Tabel 3.1 tentang penjelasan dan definisi operasional penelitian :

Tabel 3. 1. Definisi Operasional Variabel

|  |  |  |
| --- | --- | --- |
| **No** | **Variabel** | **Definisi** |
| **1** | Normal | Dataset citra rontgen dada paru-paru normal yang tidak terkena infeksi pneumonia, Tuberculosis(TBC) ataupun COVID-19. |
| **2** | Tuberculosis (TBC) | Dataset citra rontgen dada terinfeksi Tuberculosis (TBC). |
| **3** | Pneumonia | Dataset citra rontgen dada infeksi paru-paru. |
| **4** | COVID-19 | Dataset citra rontgen dada terinfeksi positif COVID-19. |

* 1. **Jenis dan Sumber Data**

Jenis data yang digunakan dalam penelitian ini adalah data sekunder. Data tersebut diperoleh dari berbagai sumber yang dijabarkan pada table 3.2 :

Tabel 3. 2. Sumber Dataset

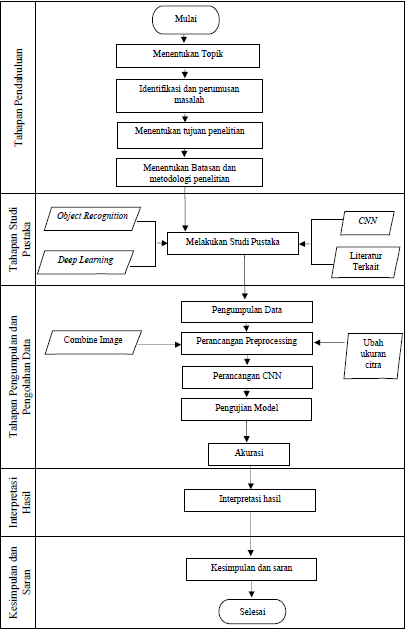
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Variabel | Judul | Sumber | Jumlah |
| 1. | Normal | Chest X-Ray Images (pneumonia) | <https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia> | 250 |
| 2. | Tuberculosis (TBC) | Pulmonary Chest X-Ray Abnormalities | <https://www.kaggle.com/kmader/pulmonary-chest-xray-abnormalities?select=Montgomery> | 250 |
| 3. | Pneumonia | Chest X-Ray Images | <https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia> | 250 |
| 4. | COVID-19 | COVID-19 Radiography Database | <https://www.kaggle.com/tawsifurrahman/covid19-radiography-database> | 219 |
| COVID-19 chest xray | <https://www.kaggle.com/bachrr/covid-chest-xray> | 31 |
| **Jumlah** | | | | 1.000 |

* 1. **Metode Analisis Data**

Metode analisis data yang digunakan dalam penelitian ini adalah metode C*onvolutional Neural Network* (CNN) dengan menggunakan model arsitektur VGG19, ResNet50, dan InceptionV3yang bertujuan untuk klasifikasi citra rontgen dada ke kelas atau label normal, *tuberculosis*(TBC), *pneumonia*, dan COVID-19.

* 1. **Tahapan penelitian**

Langkah atau tahapan yang dilakukan pada penelitian ini digambarkan melalui Gambar 3.1 berikut ini :



Gambar 3. 1.  *flowchart* Tahapan Penelitian

* 1. **Rancangan Dataset**

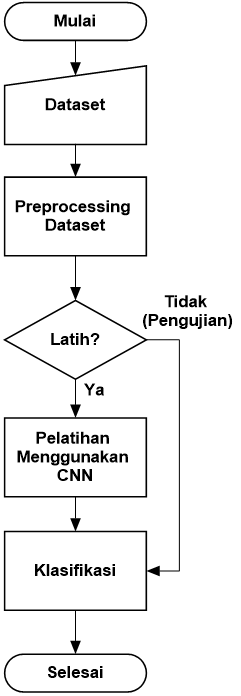
Penggunaan dataset pada metode *convolutional Neural Network* (CNN) yaitu berupa data gambar *Chest X-ray*. Model CNN akan berjalan dengan baik ketika menggunakan data gambar yang banyak. Sehingga sebuah model dapat belajar mengenali gambar tersebut. Dataset yang digunakan dalam penelitian ini berupa gambar yang dikumpulkan melalui melalui berbagai sumber (Tabel 3.2). Datasetdibagi berdasarkan 4 label yang dapat dilihat pada Gambar3.1 yaitu gambar hasil rontgen dada Normal, *Tuberculosis* (TBC), *Pneumonia*, dan positif COVID-19.

|  |  |  |  |
| --- | --- | --- | --- |
|  | CHNCXR_0502_1 |  |  |
| Normal | TBC | *Pneumonia* | COVID-19 |

Gambar 3. 2 Contoh *subclass* dataset

* 1. **Rancangan *Deep Learning***

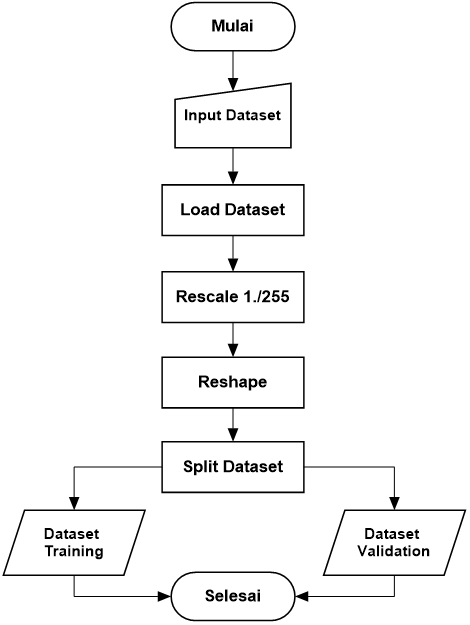
Metode yang diajukan untuk klasifikasi *Chest X-Ray* yaitu terdiri dari beberapa tahapan. Tahapan-tahapan tersebut dimulai dari pengumpulan data citra Normal, *Tuberculosis* (TBC), *Pneumonia* dan COVID-19yang akan digunakan untuk citra latih (*train*) dan dan citra uji (*test*). Kemudian masuk ke tahap *preprocessing,* kemudian pelatihan menggunakan model-model *Convolutional Neural Nerwork* (CNN)*,* lalu terakhir data citra yang digunakan sebagai pengujian (*testing*) di klasifikasikan ke dalam label dataset yang telah dilatih*.*

****

Gambar 3. 3 *Design flowchart Deep Learning*

* 1. **Rancangan *Image preprocessing***

Perancangan *preprocessing* merupakan langkah yang wajib dilakukan sebelum masuk perancangan *Convolutional Neural Nerwork* (CNN). *Preprocessing* berfungsi untuk menyiapkan citra agar dapat digunakan pada proses selanjutnya. Hal-hal yang dilakukan dalam perancangan *preprocessing* yang pertama yaitu dilakukan proses *thresholding* yang merupakan proses untuk menjadikan nilai citra menjadi 0-1 dari 0-255 yang gunanya mempermudah dalam proses input *training*. Kemudian proses *resize* menjadi ukuran 256x256 yang gunanya adalah menyesuaikan semua ukuran citra agar sama. Setelah itu dilakukan proses *split* untuk membagi data dalam beberapa rasio (20% : 80% ; 50% : 50% ; 80% : 80%) yang telah dikumpulkan menjadi dua, yaitu data *train* dan data *validation* yang digunakan untuk proses *training.*



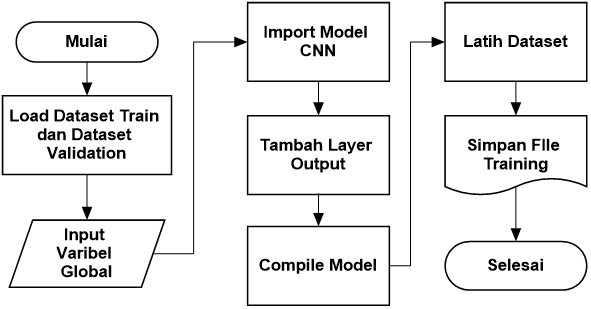
Gambar 3. 4. *Design* *flowchart preprocessing dataset*

* 1. **Rancangan Pelatihan (*Training*)**

Algoritma *Convolutional Neural Network* (CNN)membutuhkan proses *training* dan *testing*. Proses *training* ini bertujuan untuk melatih algoritma CNN dalam mengenali pola setiap gambar pada dataset yang telah melewati *preprocessing* dan membentuk sebuah model berdasarkan pelatihan tersebut. Tahap pertama dalam proses *training* adalah memanggil dataset *train* dan *validation* yang telah melewati proses *preprocessing*. Kemudian input varibel global yang akan digunakan pada proses *training* atau pelatihan. Varibel yang digunakan yaitu ukuran 256x256, batch dengan ukuran 8, dan epoch. *epoch* merupakan jumlah perulangan saat *training* dilakukan, dalam penelitian ini variasi jumlah *epoch* yang digunakan adalah 5, 10, dan 20.

Kemudian setelah melewati proses memanggil dataset dan variabel global, dilakukan proses *import* model CNN yang digunakan. Model CNN yang digunakan yaitu VGG19, ResNet50, dan InceptionV3, masing-masing model tersebut akan di *import* saat melakukan proses *training*. Model tersebut yang digunakan hanya *feature layer* saja, dan untuk *classification layer* menggunakan *layer* yang dirancang menyesuaikan jumlah kelas dataset artinya pengujian tidak menggunakan *classification layer* bawaan dari model-model tersebut. Karena jumlah kelas dataset hanya terdapat 4, maka digunakan *classfication layer* yang dirancang sebagai *layer output* atau *layer* klasifikasi.

Setelah membuat model selanjutnya adalah melakukan proses *training*, dimana proses *training* ini merupakan proses dimana machine learning bekerja sehingga algoritma yang sudah kita definisikan dapat mengingat pola dari masing masing kelas pada data yang kita train. Tahap *training* atau pelatihan adalah tahap dimana model CNN diuji dengan data latih yang sudah disediakan. Data *training* yang telahdi bagi kembali menjadi dua pada tahap *preprocessing* yaitu *training* dan validasi akan di latih pada tahap ini. Waktu yang dibutuhkan untuk proses *training* tergantung dari banyak data dan *epoch* (perulangan )yang digunakan, semakin banyak data yang dilatih maka semakin lama juga proses pelatihan data tersebut. Setelah proses *training* atau pelatihan menggunakan model yang di definisikan selesai, maka hasil *training* tersebut dalam bentuk file. Sehingga dapat digunakan untuk pengujian atau *Testing* tanpa proses *training* yang berulang-ulang.



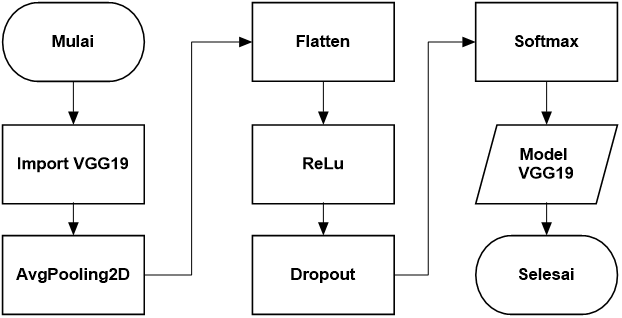
Gambar 3. 5. *Flowchart* Proses Pelatihan (*training*)

Terdapat 3 model arsitektur CNN yang digunakan pada penelitian ini yaitu VGG19, ResNet50, dan InceptionV3. Berikut penjelasan setiap model tersebut :

1. **VGG19**

VGG19 adalah varian model VGG yang singkatnya terdiri dari 19 lapisan (16 convolution *layer*s, 3 *Fully Connected* *layer*, 5 MaxPool *layer*s dan 1 SoftMax *layer*). Karakteristik utama dari arsitektur ini adalah memiliki sejumlah besar *hiperparameter*, fokus utamanya pada kernel ukuran 3×3 sederhana dalam *Convolusional* *layer* dan ukuran 2×2 di *layer* *max pooling*. Pada *layer* terakhirnya, VGG19 memiliki 2 FC (lapisan *Fully Connected*) diakhiri oleh softmax untuk output. Perbedaan antara VGG-16 dan VGG-19 adalah VGG-19 memiliki tiga blok konvolusional tambahan (Zhang et al., 2019.)

Berikut merupakan *flowchart* proses input model VGG19 yang digunakan pada proses *training* :

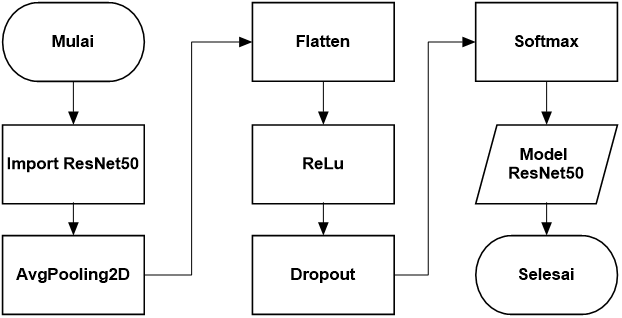


Gambar 3. 6. *Flowchart* Proses Input Model VGG19

1. **ResNet50**

Inovasi utama model ResNet50 adalah memperkenalkan jaringan arsitektur baru dalam jaringan menggunakan lapisan residu. ResNet50 terdiri dari lima langkah masing-masing dengan blok konvolusi dan identifikasi, setiap blok konvolusi memiliki 3 lapisan konvolusi dan setiap blok identifikasi juga memiliki 3 lapisan konvolusi. ResNet50 memiliki 50 jaringan residu dan menerima gambar 224×224 piksel.

Berikut merupakan *flowchart* proses input model ResNet50 yang digunakan pada proses *training* :

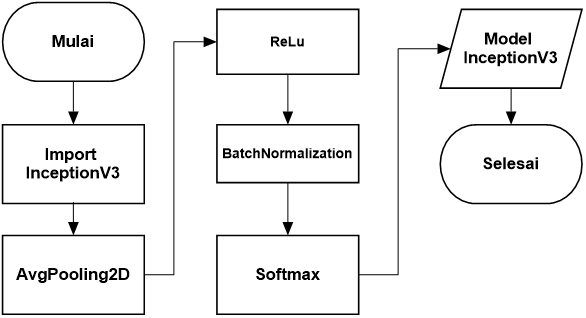


Gambar 3. 7. *flowchart* Proses Input Model ResNet50

1. **InceptionV3**

Inception berbeda dari model CNN lainnya di mana dalam strukturnya Inception adalah inception blocks, itu berarti memprosestensor input yang sama dengan beberapa filter dan menggabungkan hasilnya. InceptionV3 adalah versi baru dari model Inception yang diperkenalkan untuk pertama kalinya pada tahun 2015. InceptionV3 adalah versi yang ditingkatkan dari inceptionV1 dan inceptionV2 yang mempunyai lebih banyak parameter, dan memiliki 42 *layer*. InceptionV3 memiliki blok-blok *convolutional* *layer* paralel dengan 3 ukuran filter yang berbeda (1x1, 3x3, 5x5). Selain itu, terdapat juga 3×3 *max pooling* juga. Output digabungkan dan dikirim ke modul awal berikutnya.

Berikut merupakan *flowchart* proses input model InceptionV3 yang digunakan pada proses *training* :

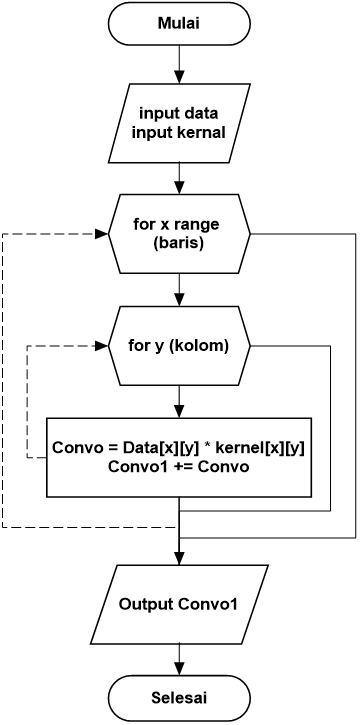


Gambar 3. 8. *Flowchart* Proses Input Model InceptionV3

Berikut penjelasan *layer*-*layer* yang digunakan pada perancangan model-model tersebut :

1. ***Convolutional* *Layer***

*Convolution Layer* merupakan bagian dari tahap pada arsitektur CNN. Tahap ini melakukan operasi konvolusi pada *output* dari *layer* sebelumnya. *layer* tersebut adalah proses utama yang mendasari jaringan arsitektur CNN. Konvolusi adalah istilah matematis dimana pengaplikasian sebuah fungsi pada *output* fungsi lain secara berulang. Operasi konvolusi merupakan operasi pada dua fungsi argumen bernilai nyata.



Gambar 3. 9. *Flowchart Convolutional Layer*

1. **Pooling**

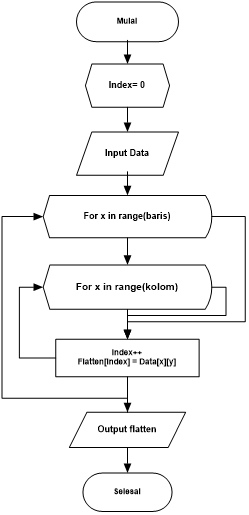
*Pooling* merupakan pengurangan ukuran matriks dengan menggunakan operasi *pooling. Pooling layer* biasanya berada setelah *convolutional*. Pada dasarnya *pooling layer* terdiri dari sebuah filter dengan ukuran dan stride tertentu yang akan secara bergantian bergeser pada seluruh area *feature map.* Dalam pooling *layer* terdapat dua macam *pooling* yang biasa digunakan yaitu *average pooling* dan *max-pooling*. Nilai yang diambil pada *average pooling* adalah nilai rata-rata, sedangkan pada *max-pooling* adalah nilai maksimal. Lapisan *Pooling* yang dimasukkan diantara lapisan konvolusi secara berturut-turut dalam arsitektur model CNN dapat secara progresif mengurangi ukuran volume output pada *feature map*, sehingga mengurangi jumlah parameter dan perhitungan di jaringan, untuk mengendalikan *Overfitting*. Lapisan pooling bekerja di setiap tumpukan *feature map* dan melakukan pengurangan pada ukurannya. Bentuk lapisan *pooling* umumnya dengan menggunakan filter dengan ukuran 2x2 yang diaplikasikan dengan langkah sebanyak dua dan beroperasi pada setiap irisan dari inputnya.

|  |  |
| --- | --- |
| MaxPooling | Avg |
| (a) | (b) |

Gambar Gambar 3. 10. (a) max pooling ; (b) average pooling

1. **Flatten**

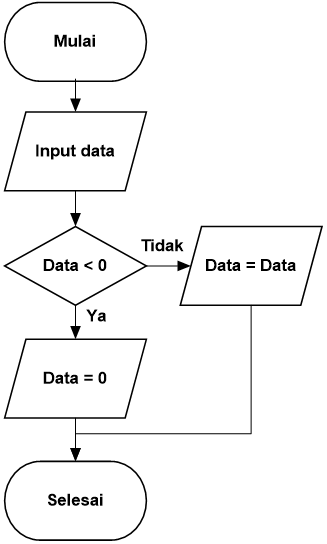
*Flowchart* Proses *Flatten* menggambarkan proses *flatten* yang lebih detail, proses *flatten* adalah proses untuk mengubah suatu matriks ke dalam bentuk *vector* atau kedalam bentuk 1 dimensi. *Flowchart* proses *flatten* dapat dilihat pada gambar 3.11 dibawah ini.



Gambar 3. 11. *Flowchart Flatten Layer*

1. **ReLu**

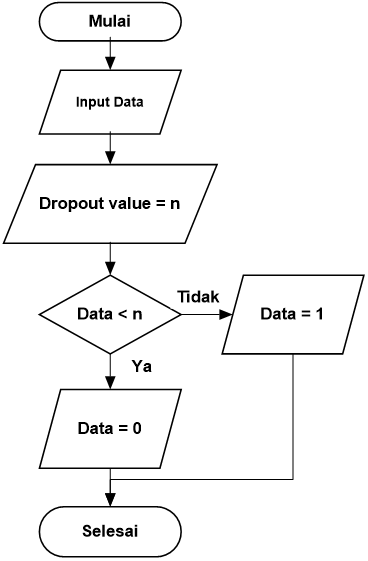
Pada dasarnya fungsi ReLU (*Rectified Linear Unit*) melakukan “*threshold*” dari 0 hingga *infinity.* Pada fungsi ini masukan dari neuron-neuron berupa bilangan negatif, maka fungsi ini akan menerjemahkan nilai tersebut kedalam nilai 0, dan jika masukan bernilai positif maka *output* dari neuron adalah nilai aktivasi itu sendiri.



Gambar 3. 12. *Flowchart* ReLu *Layer*

1. ***Dropout***

*Dropout* merupakan proses mencegah terjadinya *overfitting* dan juga mempercepat proses *learning*. Overfitting adalah kondisi dimana hampir semua data yang telah melalui proses training mencapai persentase yang baik, tetapi terjadi ketidaksesuaian prediksi. *Dropout* mengacu kepada menghilangkan sementara neuron yang berupa *hidden* mapun *visible* *layer* di dalam jaringan. Neuron yang akan dihilangkan akan dipilih secara acak. Setiap neuron akan diberikan probabilitas yang bernilai antara 0 dan 1.



Gambar 3. 13. *Flowchart* proses *dropout layer*

1. ***Batch Normalization***

*Batch Normalization* adalah metode yang menormalkan aktivasi dalam jaringan di seluruh *mini-batch* dengan ukuran tertentu. Untuk setiap fitur, Batch Normalization menghitung mean dan varian dari fitur tersebut dalam mini-batch. *Batch Normalization* menerapkan transformasi yang mempertahankan output rata-rata mendekati 0 dan standar deviasi output mendekati 1.

1. ***Softmax***

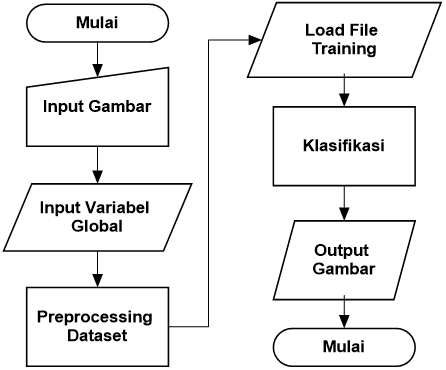
*Softmax Classiefer* adalah generalisasi dari fungsi logistik. Output dari *softmax* ini dapat digunakan untuk mewakili distribusi sebuah katrgori. *Softmax function* digunakan dalam berbagai macam metode klasifikasi contohnya *multinomial logistic regression, multiclass linear discriminant analisys, naive Bayes classiefer,* dan *neural network.* Secara sepsifiknya fungsi ini biasa digunakan pada metode klasifikasi *multinomial logistic regression* dan *multiclass linear discriminant analisys.*

*Softmax* juga memberikan hasil yang lebih intuitif dan juga memiliki interpretasi probabilistik yang lebih baik dibanding algoritma klasifikasi lainya. *Softmax* memungkinkan kita untuk menghitung probabilitas untuk semua label. Dari label yang ada akan diambil sebuah vektor nilai bernilai riil dan merubahnya menjadi vektor dengan nilai antara nol dan satu yang bila semua dijumlah akan bernilai satu

* 1. **Rancangan Pengujian (*Testing*)**

Pada tahap ini data yang telah melalui proses *training* akan digunakan untuk klasifikasi data baru yang belum diketahui kelasnya. Proses ini mencocokkan antara data masukkan dengan data yang telah melalui proses training. Tahap pertama adalah input data berupa gambar *Chest X-Ray* yang ingin diklasifikasi dan menginput variabel global yang akan digunakan pada proses preprocessing. Lalu data input tersebut di processing terlebih dahulu seperti yang dilakukan pada proses training. Proses *thresholding* yang merupakan proses untuk menjadikan nilai citra menjadi 0-1 dari 0-255, kemudian gambar *Chest X-Ray* diubah ukurannya menjadi 256x256.

Setelah gambar *Chest X-Ray* telah melewati preprocessing, tahap selanjutnya adalah memilih file training yang telah disimpan untuk digunakan sebagai klasifikasi. Setelah proses klasifikasi dijalankan, maka akan ditampilkan gambar *Chest X-Ray* beserta dengan hasil klasifikasi berupa label dan persentase akurasi. Dari penjelasan diatas bentuk alur proses pengujian/*testing* dapat dilihat pada gambar 3.13.



Gambar 3. 14. *Flowchart* Proses Pengujian (*Testing*)

* 1. **Perangkat Pengembangan dan Pengujian**

1. Perangkat Keras (*Hardware*) yang digunakan adalah Laptop Asus A407U dengan speksifikasi sebagai berikut:
2. Intel® Core i3 6006U CPU @2000 GHz
3. Intel® HD Graphics 520
4. Memory/RAM 10GB
5. Harddisk 1 TB
6. Perangkat Lunak (*Software*) yang digunakan adalah sebagai berikut:
7. Microsoft Windows 10
8. Edge
9. Python 3.6