BAB III
METODE PENELITIAN

3.1 ALAT DAN BAHAN

Rancangan penelitian ini membutuhkan sejumlah alat dan bahan untuk merancang sistem klasifikasi tuberkulosis berdasarkan citra *chest x-ray* menggunakan CNN dengan tujuh lapisan konvolusi. Alat dan bahan yang akan dijelaskan dalam penelitian ini meliputi dataset citra *chest x-ray* seta penggunaan *hardware* dan *software.*

3.1.1 Dataset

Penelitian ini menggunakan bahan berupa kumpulan *dataset* yang diperoleh dari *National Library of Medicine* (NLM) yaitu TB NIAID (*National Institute of Allergy and Infectious Diseases*) sebanyak 4200 yang berukuran 512x512 *pixels* dengan format png [7].

Gambar 3. 1 Jumlah dataset per-kelas [7]

Berdasarkan Gambar 3.1 dataset terdiri dari dua kelas, yaitu kelas normal dan kelas tuberkulosis. Terdapat 3500 gambar pada kelas normal dan 700 gambar pada kelas tuberkulosis. Untuk meningkatkan akurasi, seluruh gambar dalam dataset akan dijalani proses konvolusi sebanyak tujuh kali. Konvolusi ini merupakan tahap penting dalam pengolahan gambar dengan menggunakan metode CNN.

3.1.2 Perangkat Keras (*Hardware*)

Perangkat keras untuk menjalankan *software* dalam penelitian ini yaitu menggunakan laptop. Peneliti memanfaatkan laptop untuk pengolahan data, pengumpulan data, dan pengembangan sistem klasifikasi. Peneliti menggunakan laptop dengan spesifikasi yang tertera pada Tabel 3.1 untuk menjalankan analisis data yang kompleks dan melakukan simulasi yang memerlukan kapasitas komputasi tinggi dengan konfigurasi yang sesuai seperti yang tercantum dalam tabel tersebut, peneliti dapat memastikan bahwa proses penelitian berjalan lancar dan memungkinkan untuk menghasilkan hasil yang akurat.

Tabel 3. 1 Spesifikasi perangkat keras

Sistem Oprasi	**Windows 10**
Processor	Intel® Core™ i3-8130U CPU@ 2.20GHz
RAM	4,00 GB

3.1.3 Perangkat Lunak (*Software*)

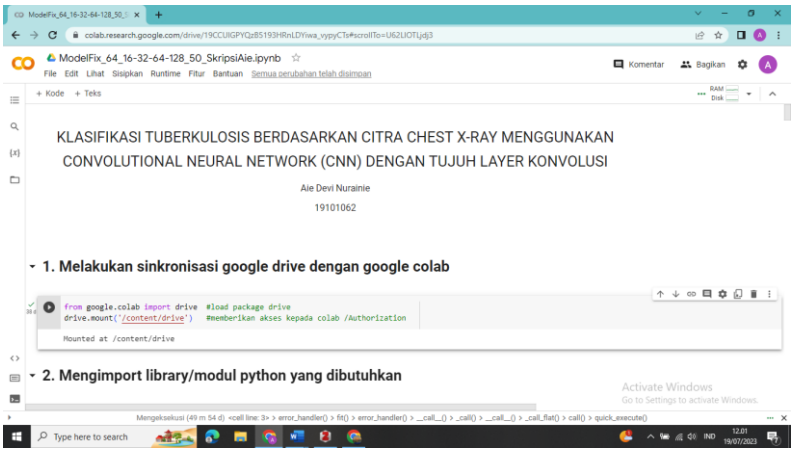
Pada penelitian ini digunakan beberapa perangkat lunak dengan sistem operasi *windows* dengan *software* yang tertera pada Tabel 3.2 untuk memberikan panduan yang jelas mengenai alat-alat yang digunakan dalam kerangka penelitian ini.

Tabel 3. 2 *Software Tools*

Software	**Fungsi**
Windows 10	Sebagai Sistem Operasi
Google Chrome	Sebagai situs web
Google Colaboratory	*Jupyter Notebook* versi *cloud*

A. *Google Colaboratory*

Google Collaboratory atau *Google Colab* merupakan aplikasi berbasis web yang digunakan untuk membuat, menyimpan serta membagian program melalui *Google Drive*.



Gambar 3. 2 Tampilan awal *google colab* [31]

Gambar 3.2 merupakan tampilan dari *Google Colab* yang dapat digunakan untuk programmer pemula melalui bahasa pemrograman *Python*, karena berbasis web sehingga tidak perlu memasang aplikasi apapun pada perangkat yang digunakan, pada dasarnya *google colab* ini sama seperti aplikasi *Jupyter Notebook*, hanya saja berbentuk *cloud* yang dioperasikan melalui *browser* sehingga tidak memberatkan kinerja perangkat yang dimiliki [32].

B. Bahasa Pemrograman *Python*

Python merupakan bahasa permrograman interpretatif multiguna dengan filosofi perancangan yang berfokus pada tingkat keterbacaan kode.

Gambar 3. 3 *Python* [34]

Gambar 3.3 merupakan logo dari *Python* dimana *python* digunakan untuk berbagai keperluan pengembangan perangkat lunak dan dapat berjalan di berbagai platform sistem operasi sitasi [33].

C. *Library*

Untuk memudahkan penggunaan bahasa pemrograman python, maka digunakan sebuah *library* untuk membuat sebuah paket yang berisi ribuan model

terkait. *Library* Python adalah sebuah kumpulan modul atau data yang berada pada suatu fungsi atau rumpun yang sama dan berisi banyak sekali kode yang dapat digunakan berulang kali pada program yang berbeda sesuai dengan kebutuhan [35]. Pada *python* terdapat beberapa contoh *library* yang populer digunakan khususnya untuk keperluan data *science* ,berikut contoh *library* yang digunakan.

a) *Tensorflow*

Tensorflow merupakan *interface* untuk mengeksekusi perintah dengan menggunakan informasi yang dimiliki mengenai objek yang dikenal serta dapat membedakan antara objek satu dengan objek lainnya, *tensorflow* ini mendukung *arsitektur Neural Network* (NN) seperti *Recurrent Neural Network* (RNN), CNN, dan *Deep Belief Network* (DBN). *Tensorflow* memiliki fitur untuk menjalakan proses pelatihan suatu model dengan menggunakan *Central Processing unit* (CPU) ataupun *Graphic Processing Unit* (GPU) [36].

b) NumPy

Numpy atau disebut *numerical Python* merupakan salah satu *library* yang banyak digunakan untuk memproses komputasi numerik. *Library* ini mampu untuk membuat objek N-dimensi *array*. Maksud *array* disini adalah sekumpulan variable yang tipe datanya sama [37].

c) Pandas

Pandas berarti sebuah *library open source* yang ada pada bahasa pemrograman *Python* yang sering digunakan untuk memproses data, mulai pembersihan data, manipulasi data, hingga melakukan analisis data. *Library* Pandas ini dibangun di atas dua pustaka inti *Python*, yaitu matplotlib untuk visualisasi data dan NumPy untuk operasi matematika [38]

d) Keras

Keras merupakan *library* jaringan syaraf tiruan tingkat tinggi yang ditulis dengan menggunakan Bahasa *Python* dan mampu berjalan di atas *Tensorflow*, CNTK, atau Theano. *Library* ini menyediakan fitur yang digunakan dengan fokus mempermudah pengembangan lebih dalam tentang *Deep Learning*. Keras memiliki 3 karakteristik yaitu *simple*, *flexible*, dan *powerful* [39].

e) *Matplotlib*

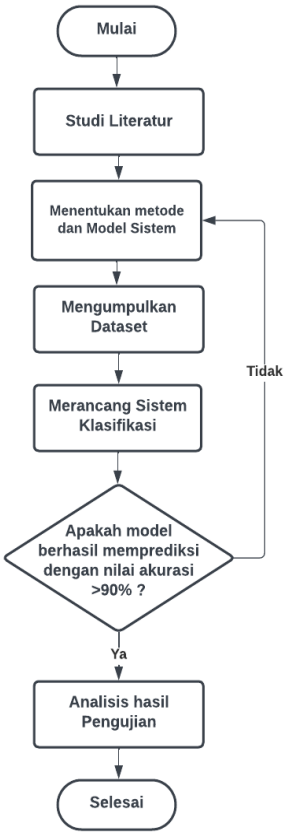
Matplotlib adalah suatu *library* atau *package* yang paling populer di bahasa *python* untuk melakukan visualisasi data seperti membuat plot grafik untuk satu sumbu atau lebih. Setiap sumbu memiliki sumbu horizontal (x) dan sumbu vertikal (y) [40].

f) *Seaborn*

Seaborn adalah *library* untuk membuat grafik dan statistik dengan menggunakan *Python*. *Library* ini dibangun berdasarkan *library Matplotlib* yang sudah ada. Kemudian terintegrasi dengan struktur data pada Pandas. *Seaborn* berorientasi pada fungsi *plotting* berdasarkan dataset yang beroperasi pada dataframe dan *array* yang berisi seluruh dataset yang secara internal [41].

3.2 ALUR PENELITIAN

Dalam sebuah perancangan penelitian ini dibutuhkan alur penelitian dengan tujuan agar dalam proses perancangan dapat tersistem dan berjalan sesuai rencana yang diharapkan sebelumnya seperti yang ditunjukan pada Gambar 3.4.



Gambar 3. 4 *Flowchart* Alur Penelitian

Berdasarkan Gambar 3.4 untuk kebutuhan tahapan penelitian ini peneliti melakukan studi literatur, menentukan metode dan model sistem, pengumpulan dataset, merancang sistem klasifikasi, apabila model berhasil mengklasifikasi dengan menampilkan hasil akurasi > 90% maka akan dilanjutkan pada analisis hasil pengujian model untuk klasifikasi citra *chest x-ray* tuberkulosis sedangkan apabila sistem klasifikasi tidak dapat memprediksi dengan nilai < 90% maka akan dilakukan pengecekan kembali pada model sistem.

3.2.1 Studi Literatur

Untuk mendapatkan informasi untuk memecahkan masalah dari studi literatur yang telah dilakukan ditemukan suatu solusi yaitu dapat membantu mempercepat identifikasi penyakit dalam proses diagnosa *tuberculosis* dengan mengklasifikasi citra menggunakan bantuan *computer aided diagnosis.* Pada penelitian tentang "Klasifikasi Tuberkulosis Berdasarkan Citra *Chest X-Ray* Menggunakan *Convolutional Neural Network* Dengan Tujuh Konvolusi *Layer*" beberapa tahapan di mulai mencari referensi sebanyak mungkin dari sumber *ebook*, jurnal, *paper*, dan referensi lainya.

3.2.2 Metode dan Model Sistem

Untuk menentukan metode dan model sistem yang akan digunakan pada penelitian ini yaitu menggunakan metode CNN (*Convolutional Neural Network)* dengan tujuh *layer* konvolusi. Proses perancangan model sistem dilakukan pada *google colaboratory* dengan menggunakan bahasa pemrograman *python* yang akan tersimpan dalam format ipynb.

3.2.3 Pengumpulan Dataset

Pengumpulan dataset didapatkan dari *National Library of Medicine* (NLM) yaitu TB NIAID (*National Institute of Allergy and Infectious Diseases*) sebanyak 4200 yang berukuran 512x512 *pixels* dengan format png . Dataset yang digunakan terdiri dari 2 kelas yaitu normal sebanyak 3500 gambar dan 700 gambar positif yang akan dilakukan untuk proses *training* dan *testing.* Sebelum data digunakan untuk *training* dan *testing* dilakukan *preprocessing* untuk

menyeimbangkan data dengan cara menambah jumlah data sebanyak 2800 sehingga total data setelah diaugmentasi yaitu sebanyak 7000 gambar. Berikut Tabel 3.3 merupakan jumlah dataset sebelum dan setelah diaugmentasi.

Tabel 3. 3 Jumlah Dataset

Jumlah Dataset Sebelum Augmentasi		**Jumlah Dataset Setelah Augmentasi**	
Normal	3500	Normal	3500
Tuberkulosis	700	Tuberkulosis	3500
Total	**4200**	**Total**	**7000**

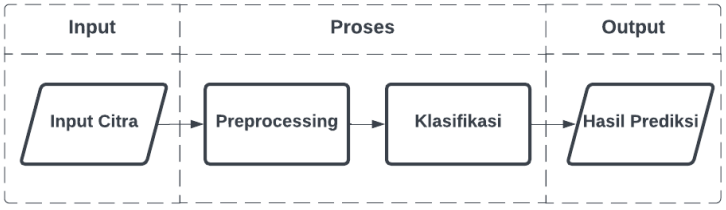
Tabel 3.4 merupakan pembagian dataset yang digunakan yaitu dengan rasio sebanyak 70% 10% 20% untuk *training* ,*validation*, dan *testing*. Model dilatih atau dibangun menggunakan *Training. Validation* digunakan untuk mengoptimalkan model saat sedang dilatih dan *Testing* digunakan untuk menguji model setelah proses pelatihan selesai.

Tabel 3. 4 Jumlah dataset dengan rasio 70% 10% 20%

Jumlah Data	
Training	4900
Validation	700
Testing	1400
Total	**7000**

3.2.4 Rancangan Sistem

Pada penelitian ini untuk merancang sistem klasifikasi memiliki alur pengerjaan yang dijelaskan melalui blok diagram yang digambarka pada Gambar 3.5.



Gambar 3. 5 Blok diagram perancangan sistem klasifikasi

Gambar 3.5 merupakan gambaran tahap proses perancangan yang akan dilakukan pada penelitian ini. Tahapan perancangan sistem yang terdapat pada penelitian ini telah dirancang dengan cermat untuk memastikan bahwa pengembangan sistem dilakukan dengan efisien dan sesuai dengan tujuan penelitian.

A. *Input* Citra

Input citra merupakan proses yang paling utama untuk melakukan klasifikasi pada sistem ini. *Input* yang digunakan berupa *citra chest x-ray* yang telah tersedia dari dataset dengan format png dan ukuran 512x512 piksel. Kemudian sistem akan memproses citra *chest x-ray* menggunakan metode *convulutional neural network* dalam melakukan klasifikasi penyakit tuberkulosis.

B. *Preprocessing*

a) Augmentasi data

Proses *preprocessing* yang pertama yaitu melakukan augmentasi data untuk meningkatkan jumlah data dalam dataset.

```
import Augmentor
import os
def perbanyak_(ini, sebanyak_ini):
source_dir = ini
output_dir = "."
p = Augmentor.Pipeline(source_directory=source_dir,
output_directory=output_dir)
p.random_distortion(probability=1, grid_width=4, grid_height=4,
magnitude=1)
p.rotate(probability=0.7, max_left_rotation=13,
max_right_rotation=13)
p.zoom_random(probability=0.5, percentage_area=0.9)
p.crop_random(probability=0.6, percentage_area=0.9)
p.resize(probability=1.0, width=64, height=64)
p.sample(sebanyak_ini)
perbanyak_("/content/drive/MyDrive/Dataset/Tuberculosis/",2800)
```

Gambar 3. 6 *Source code* augmentasi data

Gambar 3.6 merupakan *code* dari augmentasi data untuk memperbanyak jumlah data dalam dataset diawali dengan menginstal augmentor dengan menggunakan pip. Fungsi *perbanyak_* digunakan untuk memperbanyak gambar, *distortion random* yaitu fungsi untuk menambahkan distorsi acak pada gambar

dengan *probability=1* menunjukkan probabilitas 100% bahwa operasi ini akan diterapkan pada setiap gambar atau memutar balikan data secara acak, *grid_width* dan *grid_height* menentukan berapa banyak *grid* yang digunakan untuk mengubah gambar, dan magnitude mengatur seberapa besar distorsi yang akan diterapkan. Fungsi *p.rotate* akan memutar gambar dengan kemungkinan 70%. Fungsi *max_left_rotation dan max_right_rotation* mengatur batas rotasi ke kiri dan ke kanan (dalam derajat) yang akan diterapkan pada gambar. Pada fungsi *p.zoom_random(probability=0.5)* akan dilakukan secara acak memperbesar atau memperkecil gambar dengan kemungkinan 50% dan fungsi *percentage_area* menentukan rentang persentase perubahan ukuran gambar yang akan diterapkan. Fungsi *p.crop_random* merupakan fungsi yang akan memotong secara acak bagian dari gambar dengan *probability=0.6* kemungkinan 60% dengan *percentage_area=0.9* unutk menentukan rentang persentase gambar yang dipotong. Fungsi *p.resize* akan mengubah ukuran semua gambar menjadi lebar 64 piksel dan tinggi 64 piksel dengan probabilitas 100% dan *p.sample(sebanyak_ini)* berfungsi untuk memulai proses augmentasi dan menghasilkan gambar-gambar baru yang sudah di augmentasi.

b) *Resize*

Resize dilakukan pada citra untuk memastikan bahwa semua gambar masukan memiliki dimensi yang seragam dan dapat digunakan untuk melatih model dengan baik. Dalam penelitian ini digunakan 7000 citra *chest X-ray* dengan resolusi piksel 512x512 dan format png.

```
BATCH_SIZE = 32
IMG_HEIGHT = 224
IMG_WIDTH = 224
#rescale the images
trainGenerator = ImageDataGenerator(rescale=1./255.)
valGenerator = ImageDataGenerator(rescale=1./255.)
testGenerator = ImageDataGenerator(rescale=1./255.)
```

Gambar 3. 7 *Source code resize*

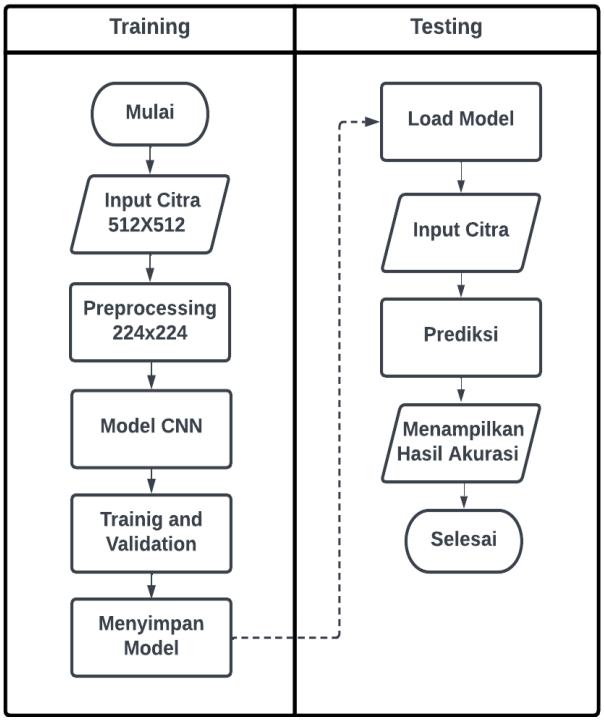
Untuk mempersiapkan data gambar dalam model seperti yang ditunjukkan pada Gambar 3.7 terdapat penggunaan parameter *BATCH_SIZE* dengan *batch size* 32 yang akan digunakan selama proses pelatihan dan pengujian model. Pada saat melakukan pelatihan, data gambar akan dipecah menjadi *batch-batch* kecil dengan

setiap *batch* berisi 32 gambar. Setelah setiap *batch*, model akan diperbarui sesuai dengan hasil pelatihan pada *batch* tersebut. Parameter *IMG_HEIGHT* dan *IMG_WIDTH* memiliki fungsi untuk menentukan dimensi tinggi dan lebar gambar yang akan digunakan dalam model. Proses selanjutnya melibatkan mengubah ukuran gambar asli menjadi ukuran yang baru yaitu 224x224 piksel.

Penggunaan parameter rescale=1/255 terlihat dalam penggunaan *ImageDataGenerator*. Fungsi ini bertujuan untuk melakukan penyekalaan terhadap nilai piksel pada gambar. Nilai piksel pada gambar asli yang semula berada dalam rentang 0 hingga 255 akan diubah menjadi nilai yang telah disesuaikan dalam rentang 0 hingga 1 setelah melalui proses penyekalaan.

C. Klasifikasi

Klasifikasi, yang merupakan tahapan penting dalam penelitian ini, memungkinkan untuk melakukan pengelompokan yang akurat pada citra chest x-ray. Proses ini berperan dalam mengidentifikasi pola-pola khas yang dapat membantu dalam diagnosis medis. Rincian mengenai langkah-langkah klasifikasi ini dapat ditemukan pada gambar 3.4 yang menggambarkan proses secara visual.



Gambar 3. 8 *Flowchar*t proses klasifikasi

Proses kalsifikasi *taining* dan *testing* digambarkan sebagai diagram alur seperti pada Gambar 3.8 yaitu proses *training* pada penelitian ini adalah untuk

melatih model untuk memahami dan membedakan antara citra *chest x-ray* yang menunjukkan kondisi normal dan kondisi tuberkulosis. Dataset yang digunakan untuk proses *training* dan *validation* ini terdiri dari dua kelas. Tujuan dari model proses *training* ini adalah untuk menghasilkan model terbaik dengan jumlah *loss* paling sedikit dan akurasi maksimum. Proses *training* dimulai dari *input* citra dengan ukuran 512x512 *pixel* kemudian dilakukan *preprocessing* dengan mengaugmentasi data untuk meningkatkan jumlah data dalam dataset, kemudian mere*size* gambar untuk menyeragamkan ukuran gambar dengan ukuran 224x224 *pixel.* Selannjutnya pada bagian model *Convulutional Neural Network* merupakan proses sistem multi *layer* yang memiliki koneksi antara setiap lapisan, pada penelitian ini digunakan tujuh lapisan konvolusional.

Lapisan konvolusional adalah lapisan pertama, sedangkan lapisan *pooling* adalah lapisan kedua.Setiap lapisan memiliki tujuan yang berbeda dan diproses pada lapisan berikutnya. Lapisan pertama yaitu *convolution layer* 2D untuk melakukan proses konvolusi antara *matrix* citra dengan *hyparameter* yang digunakan. Lapisan ke dua adalah *pooling layer* yang akan membandingkan hasil jenis *pooling layer* antara *Max Pooling* dan *Average Pooling*. Dalam penelitian ini fitur map akan diubah dari bentuk matriks menjadi vektor satu dimensi dengan memanfaatkan *flatten*. Setelah model dijalankan maka akan dilakukan *training* and *validation* dengan percobaan melatih model dengan menggunakan beberapa *hyparameter* diantaranya menggunakan kernel 3x3 dengan fungsi aktivasi ReLu, ukuran *batch size* yang berbeda yaitu 32, 64 dan 128. Filter 16, 16, 32, 32, 64, 64 dan 128, learning rate 0,0001 serta *epoch* 50.

```
model = keras.Sequential([
        keras.layers.InputLayer(input_shape=(IMG_HEIGHT,IMG_WIDTH, 3)),
        layers.Conv2D(16, 3, padding='same', activation='relu'),
        layers.MaxPooling2D(pool_size=(2,2)),
        layers.Conv2D(16, 3, padding='same', activation='relu'),
        layers.MaxPooling2D(pool_size=(2,2)),
        layers.Conv2D(32, 3, padding='same', activation='relu'),
        layers.MaxPooling2D(pool_size=(2,2)),
        layers.Conv2D(32, 3, padding='same', activation='relu'),
        layers.MaxPooling2D(pool_size=(2,2)),
        layers.Conv2D(64, 3, padding='same', activation='relu'),
        layers.MaxPooling2D(pool_size=(2,2)),
```

```
        layers.Conv2D(64, 3, padding='same', activation='relu'),
        layers.MaxPooling2D(pool_size=(2,2)),
        layers.Conv2D(128, 3, padding='same', activation='relu'),
        layers.MaxPooling2D(pool_size=(2,2)),
        Flatten(),
        Dense(512,activation = "relu"),
        BatchNormalization(),
        Dropout(0.3),
        Dense(512,activation = "relu"),
        BatchNormalization(),
        Dropout(0.3),
        Dense(2, activation='sigmoid')])

# Optimasi Model
model.compile(loss = 'sparse_categorical_crossentropy', optimizer
=Adam(learning_rate=0.0001), metrics= ['accuracy'])
```

Gambar 3. 9 *Source code* model *convolutional neural etwork*

Berdasarkan Gambar 3.9 merupakan stuktur model CNN dengan input gambar yang memiliki ukuran *(IMG_HEIGHT, IMG_WIDTH, 3)* sesuai dengan ukuran gambar dengan 3 *channel* warna (RGB). Model memiliki tujuh *layer* konvolusi dan *max pooling* yang diikuti oleh aktivasi ReLU. *Layer* konvolusi bertujuan untuk mengekstrak fitur-fitur dari gambar, sedangkan *max pooling* digunakan untuk mengurangi dimensi data dan mempertahankan fitur paling penting. Setelah rangkaian konvolusi dan *max pooling*, layer *flatten* digunakan untuk mengubah *output* dari *layer* konvolusi menjadi vektor satu dimensi sebelum dilanjutkan ke layer Dense. Model memiliki dua *layer* dense dengan fungsi aktivasi ReLU. Layer Dense bertujuan untuk menghubungkan setiap neuron dari *layer* sebelumnya dengan setiap neuron pada layer berikutnya. Fungsi aktivasi ReLU membantu memberikan sifat *non-linear* pada model. Penggunaan *BatchNormalization* digunakan untuk menormalisasi input pada setiap *batch* dan membantu meningkatkan stabilitas dan kecepatan konvergensi model. Untuk mengurangi *overfitting* digunakan *layer Dropout* untuk secara acak dengan menonaktifkan 30% dari neuron selama pelatihan. Untuk *output layer* menggunakan fungsi aktivasi sigmoid yang akan menghasilkan probabilitas untuk setiap kelas (2 kelas). Untuk menyelesaikan tugas klasifikasi dengan label yang benar dalam bentuk bilangan bulat (*integer*) pada optimasi model digunakan

fungsi *loss 'sparse_categorical_crossentropy'*. Penggunaan *optimizer* Adam dengan *learning rate* 0.0001 digunakan untuk mengoptimalkan model selama pelatihan kemudian model akan diukur kinerjanya menggunakan metrik akurasi.

```
# proses pelatihan
start = time.time()
history = model.fit(trainDataset,

steps_per_epoch=len(trainDataset.filenames) // BATCH_SIZE,
                    epochs=50,
                    validation_data=valDataset,

validation_steps=len(valDataset.filenames) // BATCH_SIZE,)
elapsed = time.time() - start
print('Computation time = ' + str(round(elapsed,2)) + 's')
```

Gambar 3. 10 *Code* proses pelatihan model

Gambar 3.10 merupakan proses pelatihan model yang diawali dengan merekam waktu saat pelatihan dimulai dengan menggunakan modul time. Metode model fit digunakan untuk melatih model dengan menggunakan data pelatihan (*trainDataset*) dan data validasi (*valDataset*) selama 50 *epoch*. Jumlah langkah per *epoch* diatur dengan *steps_per_epoch* dan jumlah langkah validasi diatur dengan *validation_steps*. Setelah pelatihan selesai waktu berakhirnya pelatihan dihitung dengan mengurangkan waktu sekarang dengan waktu mulai. Kemudian *output* mencetak waktu yang diperlukan untuk melatih model dalam detik dengan dua angka desimal. Apabila proses pelatihan telah selesai maka hasil dari evaluasi akan memberikan gambaran tentang seberapa baik model yang dilatih mampu melakukan klasifikasi pada data uji yang belum pernah dilihat sebelumnya.

```
#evaluasi model pada kumpulan data uji
loss, acc = model.evaluate(testDataset)

print('Loss:', loss)
print('Accuracy:', acc)
```

Gambar 3. 11 Model evaluasi

Gambar 3.11 merupakan *evaluate* pada model untuk mengevaluasi performa model terhadap data uji *(testDataset). Evaluate* akan memberikan kembali dua nilai yaitu *loss* dan *accuracy* dari model terhadap data uji. *testDataset* adalah

dataset yang berisi data uji dengan jumlah 1400 yang akan digunakan untuk mengevaluasi model. *Evaluate* akan memproses data uji melalui model dan menghitung nilai *loss* dan akurasi dari model berdasarkan data uji tersebut. Setelah proses evaluasi selesai, nilai *loss* akan disimpan dalam variabel *loss* dan nilai akurasi akan disimpan dalam variabel *acc*. Untuk melihat performa model terhadap data uji dapat mencetak nilai *loss (loss)* dan *accuracy (acc)*.

D. Hasil Prediksi

Hasil prediksi merupakan variable yang merepresentasikan kinerja yang digunakan untuk menilai tolak ukur keberhasilan model yang telah dibangun sehingga dapat mengklasifikasi kemudian didapatkan tingkat akurasi yang diharapkan.

```
# Melakukan klasifikasi
y_pred = model.predict(imagestotal)

y_pred

y_predint = np.argmax(y_pred, axis=1)

y_predint

labelstotalint = [np.where(r==1)[0] for r in labelstotal]
```

Gambar 3. 12 *Source code* proses klasifikasi

Gambar 3. 12 merupakan *code* model untuk melakukan prediksi pada data gambar *(imagestotal)* dan kemudian mengambil indeks dengan nilai tertinggi dari prediksi *(y_predint)*. Model *predict* akan menghasilkan probabilitas prediksi untuk setiap kelas untuk setiap gambar dalam data imagestotal. Fungsi *y_pred* merupakan variabel yang akan berisi hasil prediksi dalam bentuk probabilitas (nilai antara 0 dan 1) untuk setiap kelas untuk setiap gambar dalam data imagestotal dengan menggunakan fungsi *np.argmax*. Dengan ini akan didapatkan hasil prediksi kelas dalam bentuk *integer* untuk setiap gambar dalam data *imagestotal*. Selain itu dilakukan perubahan pada label kelas dari format *one-hot encoding* ke format *integer (labelstotalint)* menggunakan fungsi *np.where.* Fungsi ini akan mengembalikan indeks dari elemen array yang bernilai 1, yang sesuai dengan kelas yang benar untuk setiap gambar dalam dalam data *labelstotal*.

3.2.5 Analisis Hasil Pengujian

Analisis hasil pengujian dilakukan dengan menggunakan *confusion matrix* untuk mengevaluasi hasil dari setiap gambar yang diproses. *Confusion matrix* memberikan informasi tentang prediksi kelas yang benar dan salah pada sistem. Dari *confusion matrix* ini kemudian dihitung beberapa metrik evaluasi termasuk akurasi, presisi, recall, dan F1-*score*. Dengan menggabungkan informasi dari *confusion matrix* dan nilai-nilai metrik evaluasi ini maka dapat menilai seberapa baik sistem dalam melakukan prediksi dan mengklasifikasikan gambar-gambar tersebut serta tingkat keberhasilan keseluruhan sistem.

Tabel 3. 5 *Confusion matrix* pada klasifikasi tuberkulosis

		Actual	
		***True* (0)**	***False* (1)**
Predicted Class	***True* (0)**	*True Tuberculosis*	*False Tuberculosis*
	***False* (1)**	*False* Normal	*True* Normal

Tabel 3.5 adalah hasil dari *confusion matrix* yang digunakan untuk melakukan klasifikasi tuberkulosis dengan dua kelas, yaitu kelas Normal dan kelas Tuberkulosis. Pada tabel tersebut kelas Normal diberi label 0 sementara kelas positif Tuberkulosis ditandai dengan label 1. Pada Gambar 3.13 dibawah ini berisi *source code* untuk melakukan visualisasi *confusion matrix* dan menampilkan hasil prediksi.

```
import seaborn as sns
from sklearn.metrics import confusion_matrix, classification_report
cm = confusion_matrix(labelstotal, y_predint )
sns.heatmap(cm, annot=True, fmt='g')
plt.show()

print('Confusion Matrix :')
print()
print(confusion_matrix(labelstotal, y_predint))
print()
print('Classification Report :')
print()
print(classification_report(labelstotal, y_predint))
```

Gambar 3. 13 *Source vode* visualisasi *confusion matrix*

Berdasrkan Gambar 3. 13 merupakan *code* visualisasi untuk menunjukkan jumlah hasil prediksi yang benar dan salah untuk setiap kelas pada data yang diuji. Fungsi *confusion_matrix* digunakan untuk menghitung dan menghasilkan hasil prediksi model. Sedangkan *classification_report* merupakan fungsi yang digunakan untuk menghasilkan ringkasan berbagai metrik evaluasi untuk setiap kelas pada data yang diuji yang mencakup akurasi, presisi, *recall*, dan f1-*score*. Dengan menggunakan *confusion matrix* ini dapat mempermudah melakukan evaluasi kinerja model klasifikasi terhadap data yang diuji, sehingga dapat memahami sejauh mana model berhasil melakukan klasifikasi pada setiap kelas.