BAB III
METODOLOGI PENELITIAN

3.1 ALAT YANG DIGUNAKAN

Dalam penelitian ini perangkat keras yang digunakan yaitu satu buah PC (*Personal Computer*). Penelitian ini menggunakan satu buah PC dengan menggunakan sistem operasi *Windows* 10 64-Bit. PC memiliki spesifikasi seperti pada Tabel 3.1.

Tabel 3.1 Spesifikasi PC

Processor	Intel core i3 2.00 GHz
System memory	8 GB
Hard Disk Space	500 GB
Display	1366 x 768 *Resolution*

Dalam pengerjaan sistem klasifikasi tingkat keparahan jerawat ini dibutuhkan perangkat lunak (*software*) untuk menunjang hasil yang maksimal. Perangkat lunak yang dibutuhkan adalah sebagai berikut:

1. *Google Colaboratory*
2. *Python Libraries* (Tensorflow, Keras, Pandas, dll)

3.2 ALUR PENELITIAN

Secara garis besar, penelitian ini ditujukan untuk melakukan perancangan sistem klasifikasi tingkat keparahan jerawat dalam tiga kelas yaitu kelas 0 dengan tingkat keparahan ringan dengan jerawat 1 sampai 2, kelas 1 dengan tingkat keparahan sedang dengan jerawat 3 sampai 7, kelas 2 dengan tingkat keparahan berat dengan jerawat 8 sampai 15 bahkan lebih. tingkat keparahan jerawat diklasifikasikan menggunakan metode CNN dengan arsitektur GoogleNet.

Pada tahap awal dari alur penelitian ini yaitu studi literatur mencari informasi yang berfungsi untuk mencari wawasan dan pengetahuan dan juga sebagai pendukung pada penelitian ini. dengan menggunakan sumber berdasarkan jurnal ilmiah, buku, website dan sumber terpercaya lainnya.

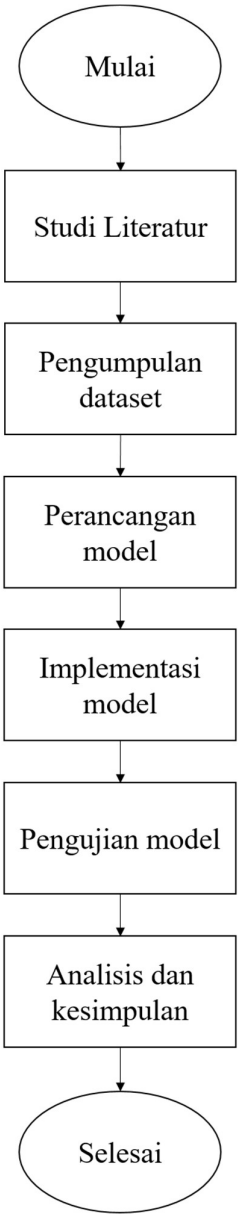
Pada tahap pengumpulan dataset, dataset yang digunakan diambil dari data publik jerawat yang diperoleh secara *online melalui https://github.com/xpwu95/LDL*[1]. Data latih berjumlah sebanyak 100 citra dari masing-masing kelas jerawat, dari jumlah keseluruhan dataset.

Pada tahap perancangan model merupakan proses Perancangan model untuk mengklasifikasikan tingkat keparahan jerawat yang akan digunakan yaitu GoogleNet. Masukan dalam proses latih untuk model klasifikasi ini menggunakan CNN dengan arsitektur GoogleNet. Peneliti menentukan *hyperparameter (optimizer*, *epoch*, dan *learning rate)* dalam proses *training* nilai akurasi yang sesuai dengan keinginan.

Pada tahap implementasi merupakan proses menentukan bahasa pemrograman yang akan digunakan. Proses evaluasi menggunakan Google Colab sebagai editor sekaligus *platform* untuk menjalankan program. Selanjutnya menentukan *library* yang akan digunakan.

Pada tahap pengujian, sebelum citra jerawat tersebut diolah, dilakukan *preprocessing* diperlukan untuk melakukan klasifikasi, seperti *rescale,* normalisasi, dan augmentasi. Hasil dari *preprocessing* akan dibagi menjadi dua data baru yaitu data latih dan data uji dengan proporsi pembagian 80% dan 20%. Model hasil proses *training* diuji menggunakan data uji untuk mendapatkan nilai akurasi beserta parameter hasil evaluasi lainnya (presisi, *recall*, f-1 *score*, dan *loss function)*. Model yang memiliki hasil pengujian yang sesuai target akan disimpan sebagai hasil penelitian.

Pada tahap terakhir yaitu analisis dan kesimpulan merupakan proses menganalisis performansi dari klasifikasi model GoogleNet menggunakan *confusion matrix.* Beberapa parameter yang akan dianalisis yaitu parameter akurasi, presisi, *recall*, f-1 *score*, dan *loss function.*



Gambar 3.1 Alur penelitian

3.2.1 Perancangan Sistem

Perancangan sistem pada penelitian ini digunakan untuk mengklasifikasikan tingkat keparah jerawat ke dalam tiga kelas yaitu *level* 0, *level* 1, dan *level* 2 dengan menggunakan *Deep Learning yang* berbasis CNN. Arsitektur CNN yang digunakan untuk membuat model *Deep Learning* untuk mengklasifikasikan tingkat keparahan jerawat pada Tugas Akhir ini menggunakan arstektur GoogleNet. Dalam perancangannya, model yang berbasis bahasa pemograman python dirancang di

Google colab untuk mempermudah akses dan memiliki *resources* yang besar. Sebelum melanjutkan proses perancangan model, data yang akan digunakan sebagai dataset untuk proses latih dan uji perlu diakuisisi terlebih dahulu dalam Google colab. Dataset yang digunakan akan menggunakan dataset dari Paper dengan sumber berikut [1] terdiri dari 1457 citra berformat jpg yang dibagi menjadi empat kelas. Kemudian, dataset perlu dirapihkan dan diseleksi terlebih dahulu dengan cara memilah gambar untuk masih-masing kelas kedalam folder terpisah untuk setiap kelasnya. Dataset yang telah dirapihkan dan diseleksi menjadi tiga kelas dengan jumlah 413 citra. Setelah dataset sudah terstuktur dengan baik, untuk dapat menjalankan *source code* python untuk perancangan model perlu menyiapkan *environment* terlebih dahulu dengan cara *import library* yang diperlukan di Google Colab agar perintah dapat dijalankan dengan baik.

Ketika konfigurasi *environment* telah selesai, skenario pertama akan melakukan *preprocessing* yang dilakukan sebelum pelatihan model dengan melakukan proses *resize* untuk menyetarakan ukuran piksel yang berbeda-beda pada citra menjadi satu ukuran yang sama untuk mempermudah sistem dalam melakukan proses deteksi klasifikasi citra. Setelah dilakukan proses *preprocessing* maka dilakukan pelatihan model pada dataset yang bertujuan agar sistem mengenali dan mengklasifikasikan objek sesuai dengan kelasnya masing-masing. Selanjutnya melakukan perancangan model GoogleNet yang mengikuti referensi berdasar pada buku[17], lalu menentukan *hyperparameter (optimizer*, *epoch*, dan *learning rate)* dalam proses *training* nilai akurasi yang sesuai dengan keinginan. *Optimizer* yang akan digunakan untuk menguji sistem ini adalah Adam dan SGD *optimizer*. Nantinya akan dibandingkan *optimizer* yang memiliki nilai terbaik dan akan digunakan sebagai model berikutnya. Contoh jumlah *epoch* biasanya diatur ke 10, 100, 500, 1000, dan lebih besar. Untuk mendapatkan model terbaik, nilai *learning rate* yang akan diuji pada penelitian ini adalah 0.1, 0.01, 0.001, 0.0001. Pelatihan model dilakukan dengan memberikan data latih kepada sistem dengan tujuan agar mendapatkan hasil klasifikasi yang maksimal. Klasifikasi merupakan keluaran pelatihan model dalam mengenali objek untuk diklasifikasi sesuai dengan kelas yang telah ditentukan.

3.2.2 Implementasi

Tugas Akhir ini dalam proses implementasinya menggunakan bahasa pemograman python, karena mayoritas *library* untuk pembuatan model dan arsitektur *Deep Learning* menggunakan python. Untuk mempermudah proses konfigurasi *environment* dan *resources* kapabilitas *hardware* untuk menjalankan program untuk merancang arsitektur, proses latih, dan proses evaluasi Tugas Akhir ini menggunakan Google Colab sebagai editor sekaligus *platform* untuk menjalankan program, hal ini dikarenakan kemudahan, fleksibilitas, dan *resources* yang ditawarkan. Dalam prosesnya, Google Colab lebih mudah untuk melakukan konfigurasi *environment* karena Google Colab berbasis Linux dan memiliki spesifikasi seperti GPU NVIDIA K80s, T4s, P4s dan P100s, memiliki RAM sebesar 13GB, mampu menyediakan penyimpanan sebesar 130 GB, dan akses ini diberikan secara gratis untuk proses *running* selama 6 jam. Apabila dibandingkan dengan dijalankan di komputer pribadi yang jauh lebih rumit dan *resources* yang terbatas.

Program python yang digunakan untuk menyusun arsitektur *Deep Learning* menggunakan beberapa *library* pendukung untuk mempermudah proses penyusunan, antara lain: sklearn, keras, tensorflow, dan matplotlib. Sklearn memiliki beberapa fungsi untuk melakukan proses *preprocessing* dataset seperti untuk pemisahan dataset menjadi data latih dan data uji. Keras berfungsi beberapa fungsi untuk membentuk *layer-layer* yang berfungsi untuk menyusun arsitektur yang akan digunakan di Tugas Akhir ini yait Google-Net. Tensorflow berfungsi untuk menentukan *hyperparameter* dari proses latih, menjalankan proses latih, dan proses uji dalam bentuk sebuah fungsi. Matplotlib adalah *library* yang menyediakan fungsi untuk menampilkan grafik untuk merepresentasikan data yang ada dalam sebuah bahasa pemograman python, sehingga grafik hasil dari proses latih dan evaluasi dapat ditampilkan menggunakan fungsi dari matplotlib.

Source code untuk model googlenet penelitian ini dapat dilihat pada penjelasan dibawah berikut:

```
import numpy as np
from PIL import Image
from imutils import paths
```

```
kumpl_gmbr = []
kumpl_lbl = []
imgpths = paths.list_images('Classroom/Dataset_Acne_Rapi')

for dirimg in imgpths:
  gmbr = Image.open(dirimg)
  gmbrz = np.array(gmbr.resize((224,224))) / 255.0
  kumpl_gmbr.append(gmbrz)

  lbl_kls = dirimg.split(os.path.sep) [-2]
  kumpl_lbl.append(lbl_kls)

kelas_asli = unique(kumpl_lbl)
kelas_asli
```

Fungsi dari *source code* diatas untuk memasukkan gambar yang akan menjadi dataset untuk perancangan model.

```
from sklearn.model_selection import train_test_split

ratio_latih = 0.8
ratio_uji = 0.2

(latihX, ujiX, latihY, ujiY) =
train_test_split(np.array(kumpl_gmbr),np.array(kumpl_lbl),
test_size=ratio_uji, random_state=7080)

print(latihX.shape)
print(latihY.shape)
print(ujiX.shape)
print(ujiY.shape)
```

Fungsi *source code* diatas untuk pembagian data latih dan uji dengan rasio 80% untuk data latih dan 20% untuk data uji.

```
from tensorflow.keras.preprocessing.image import
ImageDataGenerator

datagen = ImageDataGenerator(
      rotation_range=40,
      width_shift_range=0.2,
      height_shift_range=0.2,
      shear_range=0.2,
      zoom_range=0.2,
```

```
      horizontal_flip=True,
      fill_mode='nearest')
```

Source code diatas adalah fungsi augmentasi.

```
i=0

banyak_lth=latihX.shape
aug=[]
for jj in range(banyak_lth[0]):
  x = latihX[jj]
  x = x.reshape((1,) + x.shape)

  for aug in datagen.flow(x, batch_size=1):
      latihX = np.append(latihX, aug ,axis=0)
      label = latihY[jj]
      label = label.reshape((1,) + label.shape)
      latihY = np.append(latihY, label,axis=0)
      if i == 2:
        break
      i += 1
```

Fungsi *source code* diatas untuk proses augmentasi untuk setiap data latih dengan menggunakan fungsi augmentasi yang sudah dideklarasikan sebelumnya.

```
import tensorflow as tf
from tensorflow.keras.layers import BatchNormalization,
Conv2D, MaxPooling2D, AveragePooling2D, Dense, Activation,
Dropout, Flatten, Input, concatenate
from tensorflow.keras.models import Model
from tensorflow.keras.regularizers import l2

class GoogLeNet:
    @staticmethod
    def conv_module(x, K, kX, kY, stride, chanDim,
        padding="same", reg=0.0005, name=None):
        (convName, bnName, actName) = (None, None, None)

        if name is not None:
            convName = name + "_conv"
            bnName = name + "_bn"
            actName = name + "_act"
```

```
        x = Conv2D(K, (kX, kY), strides=stride,
padding=padding,
            kernel_regularizer=l2(reg), name=convName)(x)
        x = BatchNormalization(axis=chanDim, name=bnName)(x)
        x = Activation("relu", name=actName)(x)

        return x

    @staticmethod
    def inception_module(x, num1x1, num3x3Reduce, num3x3,
num5x5Reduce,
        num5x5, num1x1Proj, chanDim, stage, reg=0.0005):
        first = GoogLeNet.conv_module(x, num1x1, 1, 1, (1, 1),
            chanDim, reg=reg, name=stage + "_first")

        second = GoogLeNet.conv_module(x, num3x3Reduce, 1, 1,
(1, 1),
            chanDim, reg=reg, name=stage + "_second1")
        second = GoogLeNet.conv_module(second, num3x3, 3, 3,
(1, 1),
            chanDim, reg=reg, name=stage + "_second2")

        third = GoogLeNet.conv_module(x, num5x5Reduce, 1, 1,
(1, 1),
            chanDim, reg=reg, name=stage + "_third1")
        third = GoogLeNet.conv_module(third, num5x5, 5, 5, (1,
1),
            chanDim, reg=reg, name=stage + "_third2")

        fourth = MaxPooling2D((3, 3), strides=(1, 1),
padding="same",
            name=stage + "_pool")(x)
        fourth = GoogLeNet.conv_module(fourth, num1x1Proj, 1,
1, (1, 1),
            chanDim, reg=reg, name=stage + "_fourth")

        x = concatenate([first, second, third, fourth],
axis=chanDim,
            name=stage + "_mixed")

        return x

    @staticmethod
    def build(width, height, depth, classes, reg=0.0005):
        inputShape = (height, width, depth)
        chanDim = -1
```

```
        if tf.keras.backend.image_data_format() ==
"channels_first":
            inputShape = (depth, height, width)
            chanDim = 1

        inputs = Input(shape=inputShape)
        x = GoogLeNet.conv_module(inputs, 64, 5, 5, (1, 1),
            chanDim, reg=reg, name="block1")
        x = MaxPooling2D((3, 3), strides=(2, 2),
padding="same",
            name="pool1")(x)
        x = GoogLeNet.conv_module(x, 64, 1, 1, (1, 1),
            chanDim, reg=reg, name="block2")
        x = GoogLeNet.conv_module(x, 192, 3, 3, (1, 1),
            chanDim, reg=reg, name="block3")
        x = MaxPooling2D((3, 3), strides=(2, 2),
padding="same",
            name="pool2")(x)

        x = GoogLeNet.inception_module(x, 64, 96, 128, 16, 32,
32,
            chanDim, "3a", reg=reg)
        x = GoogLeNet.inception_module(x, 128, 128, 192, 32,
96, 64,
            chanDim, "3b", reg=reg)
        x = MaxPooling2D((3, 3), strides=(2, 2),
padding="same",
            name="pool3")(x)

        x = GoogLeNet.inception_module(x, 192, 96, 208, 16,
48, 64,
            chanDim, "4a", reg=reg)
        x = GoogLeNet.inception_module(x, 160, 112, 224, 24,
64, 64,
            chanDim, "4b", reg=reg)
        x = GoogLeNet.inception_module(x, 128, 128, 256, 24,
64, 64,
            chanDim, "4c", reg=reg)
        x = GoogLeNet.inception_module(x, 112, 144, 288, 32,
64, 64,
            chanDim, "4d", reg=reg)
        x = GoogLeNet.inception_module(x, 256, 160, 320, 32,
128, 128,
            chanDim, "4e", reg=reg)
        x = MaxPooling2D((3, 3), strides=(2, 2),
padding="same",
            name="pool4")(x)
```

```
        x = GoogLeNet.inception_module(x, 256, 160, 320, 32,
128, 128,
            chanDim, "5a", reg=reg)
        x = GoogLeNet.inception_module(x, 384, 192, 384, 48,
128, 128,
            chanDim, "5b", reg=reg)

        x = AveragePooling2D((7, 7), name="pool5")(x)
        x = Dropout(0.4, name="do")(x)

        x = Flatten(name="flatten")(x)
        x = Dense(classes, kernel_regularizer=l2(reg),
            name="labels")(x)
        x = Activation("softmax", name="softmax")(x)

        model = Model(inputs, x, name="googlenet")

        return model
```

Fungsi dari *source code* diatas adalah *fucntion source code* penyusun dari arsitektur GoogleNet.

```
GNet_model=GoogLeNet.build(width = 224, height = 224, depth =
3, classes = len(kelas_asli),reg=0.0005)
```

Fungsi *source code* diatas untuk deklarasi model dengan menggunakan arsitektur GoogleNet.

```
from keras.callbacks import ReduceLROnPlateau, EarlyStopping
from keras.optimizers import SGD
from keras import regularizers

reg = 0.0005

initial_learning_rate = 0.2
optimizer = SGD(learning_rate=initial_learning_rate)
GNet_model.compile(loss="categorical_crossentropy",
optimizer=optimizer, metrics=["accuracy"])

reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.5,
patience=5, min_lr=0.00001, verbose=1)
```

```
early_stop = EarlyStopping(monitor='val_accuracy',
patience=20, restore_best_weights=True, min_delta=0.01)

target_accuracy = 0.79
epochs = 100
batch_size = 32
current_accuracy = 0.0
while current_accuracy < target_accuracy:
    history = GNet_model.fit(latihX, latihY,
validation_data=(ujiX, ujiY), epochs=epochs,
batch_size=batch_size, callbacks=[reduce_lr])
    current_accuracy = history.history['val_accuracy'][-1]
    epochs += 10

test_loss, test_accuracy = GNet_model.evaluate(ujiX, ujiY)
print("Test Loss:", test_loss)
print("Test Accuracy:", test_accuracy)
```

Fungsi dari *source code* diatas untuk melakukan proses *adaptive training* dari model GoogleNet yang telah dirancang dengan menggunakan data latih teraugmentasi dan data uji.

```
from sklearn.metrics import classification_report

print("[INFO] evaluating network...")
predictions = GNet_model.predict(ujiX, batch_size=32)
print(classification_report(ujiY.argmax(axis=1),
predictions.argmax(axis=1), target_names=lb.classes_))
```

Fungsi *source code* diatas untuk menghasilkan dan menampilkan nilai evaluasi dari

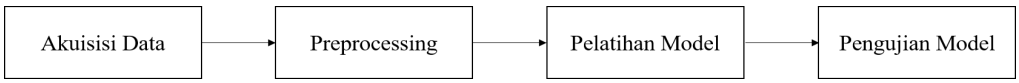
model seperti akurasi, *precision*, *recall*, dan f1-*score.*

```
from sklearn.metrics import confusion_matrix
predict = GNet_model.predict(ujiX)
rounded_pred=np.argmax(predict, axis=1)
rounded_ujiY=np.argmax(ujiY, axis=1)
print(confusion_matrix(rounded_ujiY, rounded_pred))
```

Fungsi *source code* diatas untuk menghasilkan dan menampilkan *confusion matrix.*

3.3 DIAGRAM BLOK

Penelitian ini memiliki diagram blok yang dimana diagram tersebut menjelaskan terjadinya suatu proses yang terjadi pada suatu rancangan yang akan dibuat.



Gambar 3.2 Diagram blok

Gambar 3.2 merupakan gambaran umum sistem klasifikasi tingkat keparahan jerawat yang akan dibuat. Proses akan dimulai dengan akuisisi data yaitu proses menangkap atau memproses citra analog sehingga diperoleh citra digital. Sumber dataset dari data publik jerawat yang diperoleh secara *online* melalui *https://github.com/xpwu95/LDL,* data tersebut dipublikasikan oleh xiaoping wu adalah seorang dokter perawatan paru dan kritis yang minat klinisnya meliputi fibrosis paru dan keterlibatan paru-paru dari penyakit jaringan ikat. Dia menggunakan pendekatan multi-disiplin untuk menavigasi kompleksitas penyakit paru-paru lanjut dan memberikan perawatan yang disesuaikan dengan kebutuhan masing-masing pasien.

Pada langkah *preprocessing* akan dilakukan:

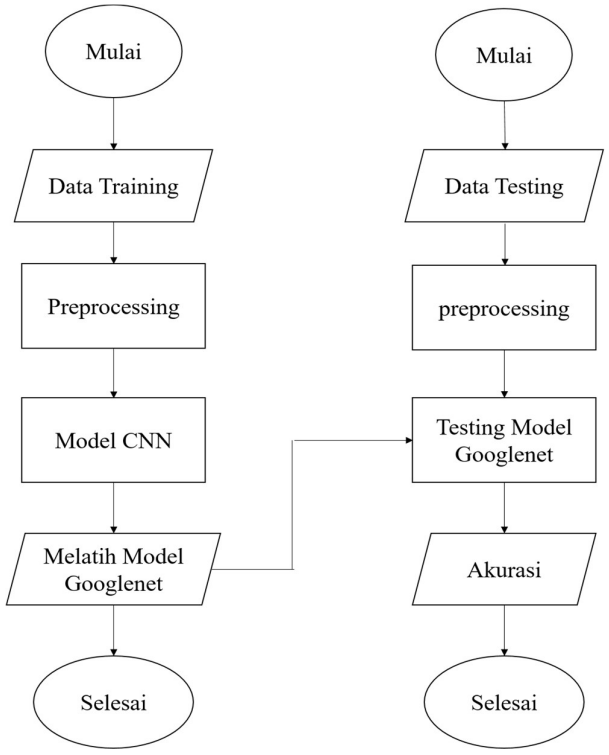
1. *Resize*: suatu proses untuk mengurangi ukuran citra, misalnya ukuran asli citra 512 piksel akan di *rescale* menjadi 224 piksel
2. *Rescale* 1/255: suatu proses untuk mengubah *range* dari setiap piksel [1,255] menjadi [0,1] agar rentang setiap gambar sama dan seimbang, sehingga tidak ada gambar yang memiliki rentang piksel yang terlalu tinggi atau terlalu rendah.
3. *Grayscale*: suatu proses untuk mengubah citra RGB menjadi citra hitam putih dengan mengganti dimensi matriks pada citra tersebut.
4. *S*egmentasi: proses mempartisi citra menjadi beberapa segmen dan merupakan proses untuk membedakan atau memisahkan objek-objek yang ada dalam suatu citra seperti memisahkan objek dengan latar belakangnya.
5. Representasi ini mengikuti *output* dari tahap segmentasi, memilih representasi hanya solusi untuk mengubah data mentah menjadi data yang diproses.

Setelah melakukan proses preprocessing untuk memaksimalkan infromasi dari dataset citra, maka selanjutnya akan dilakukan pelatihan model pada sistem untuk dapat mengenali, mengindentifikasi dan mengklasifikasikan citra berdasarkan kelasnya. Metode yang digunakan untuk melatih model ini adalah CNN dengan arsitektur GoogleNet. Pelatihan model akan menggunakan 80% dari dataset training dan akan dilakukan validation untuk pengujian model dengan menggunakan 20% dari data training yang akan dipilih secara acak. Selanjutnya melakukan Pengukuran performansi sistem dengan menggunakan lima parameter yaitu akurasi, presisi, *recall*, f-1 *score*, dan *loss function*.

3.3.3 Pengujian

Tabel 3.2 Pembagian dataset

	Level 0	Level 1	Level 2	Total
	138	139	136	413
Data latih	110	110	110	330
Data Uji	28	28	27	83



Gambar 3.3 Flowchart pengujian model

Tugas Akhir ini melakukan pengujian perancangan model Deep Learning dengan menggunakan arsitektur GoogleNet menggunakan dataset sebanyak 413 gambar atau citra. Dataset sebanyak 413 dibagi menjadi 2, sebagai data latih dan data uji dengan masing-masing proporsinya adalah 80% dan 20%. Gambar atau citra pada data latih dan data uji mengalami proses *preprocessing* terlebih dahulu agar proses perancangan model lebih optimal dan efisien. Tahap preprocessing gambar meliputi: *rescaling* dan *normalization,* khusus pada data latih melewati proses augmentasi untuk masing gambar. Kemudian, melakukan perancangan model CNN dengan menggunakan arsitektur GoogleNet. Model yang telah dirancang melalui proses *training* dengan input dari data latih yang sudah melewati proses *preprocessing,* sehingga diharapkan model dapat memiliki fungsi pengklasifikasi yang diharapkan. Model yang telah dilatih melalui proses pengujian dengan menggunakan data uji untuk mendapatkan nilai-nilai evaluasi model CNN seperti: akurasi, *loss*, *recall*, *precision*, dan f1-*score.* Apabila akurasi telah mencapai target yang diinginkan maka model CNN yang telah dirancang telah dapat menjalankan fungsi yang diharapkan pada Tugas Akhir ini.

3.3.4 Analisis

Tabel 3.3 *Confusion matrix*

		Predicted Classification		
Actual Classification	Class	Level 0	Level 1	Level 2
	Level 0	T_{0_0}	F_{1_0}	F_{2_0}
	Level 1	F_{0_1}	T_{1_1}	F_{2_1}
	Level 2	F_{0_2}	F_{1_2}	T_{2_2}

T_{0_0} : Kondisi dimana hasil prediksi benar. Hasil prediksi adalah *level 0* dan kelas sebenarnya *level 0*

F_{1_0} : Kondisi dimana hasil prediksi salah. Hasil prediksi adalah *level 1* dan kelas sebenarnya *level 0*

F_{2_0} : Kondisi dimana hasil prediksi salah. Hasil prediksi adalah *level 2* dan kelas sebenarnya *level 0*

F_{0_1} : Kondisi dimana hasil prediksi salah. Hasil prediksi adalah *level 0* dan kelas sebenarnya *level 1*

T_{1_1} : Kondisi dimana hasil prediksi benar. Hasil prediksi adalah *level 1* dan kelas sebenarnya *level 1*

F_{2_1} : Kondisi dimana hasil prediksi salah. Hasil prediksi adalah *level 2* dan kelas sebenarnya *level 1*

F_{0_2} : Kondisi dimana hasil prediksi salah. Hasil prediksi adalah *level 0* dan kelas sebenarnya *level 2*

F_{1_2} : Kondisi dimana hasil prediksi salah. Hasil prediksi adalah *level 1* dan kelas sebenarnya *level 2*

T_{2_2} : Kondisi dimana hasil prediksi benar. Hasil prediksi adalah *level 2* dan kelas sebenarnya *level 2*

Pengukuran performansi dapat dilakukan dengan menggunakan *confusion matrix*. Pengujian dilakukan berdasarkan nilai *True Positive* (TP), *True Negative* (TN), *False Negative* (FN) dan *False Positive* (FP). Beberapa parameter yang dapat digunakan untuk pengujian performansi pada penelitian ini adalah sebagai berikut:

1. Akurasi

   Akurasi merupakan tolak ukur ketepatan sistem dan merupakan suatu rasio prediksi antara *true positive* dan *true negative* dari keseluruhan data. Untuk menguji akurasi dilakukan dengan menjumlahkan prediksi *true* dan membagi dengan seluruh prediksi seperti pada persamaan (2.3).

2. Presisi

   Presisi (*precission*) merupakan parameter untuk validasi data terhadap kedekatan hasil pengukuran sistem dengan data yang diinginkan. Persamaan *precission* dapat dilihat pada persamaan (2.4).

3. *Recall*

   Recall merupakan tolak ukur seberapa banyak sistem dapat mendeteksi data yang diberikan. Misal dari 100 penderita jerawat yang diuji terdapat 80 hasil yang positif, maka *recall*-nya adalah 80%. Secara matematis dapat dituliskan dengan persamaan (2.5).

4. F-1 *score*

F-1 *score* digunakan untuk membuat nilai precission yang rendah dan nilai *recall* yang tinggi atau sebaliknya menjadi seimbang. Persamaan f-1 *score* dapat dilihat pada persamaan (2.6).

5. *Loss function*

   Loss digunakan untuk mengetahui tingkat kesalahan identifikasi sistem. Dalam penelitian ini digunakan *categorical crossentropy* sebagai *loss function default* untuk digunakan pada model klasifikasi *multi-class*. *Crossentropy* menghitung skor yang merangkum rata-rata antara distribusi probabilitas aktual dan prediksi untuk semua kelas. Persamaan *Loss function* dapat dilihat pada persamaan (2.7).