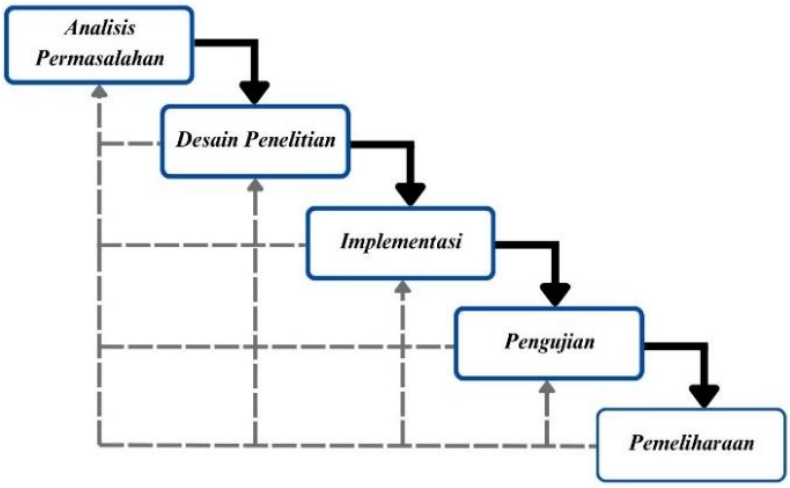
BAB III
METODE PENELITIAN

Pada pengembangan sistem ini digunakan model *Software Development Cycle* (SDLC). Model penelitian ini digunakan oleh para pengembang perangkat lunak untuk merancang, mengembangkan, menguji dan melakukan pemeliharaan perangkat lunak. Jenis metode SDLC yang dipakai pada penelitian ini adalah SDLC jenis *waterfall*, metode ini menekankan fase-fase yang berurutan serta sistematis dalam mengembangkan perangkat lunak sehingga dalam metode ini fase selanjutnya bisa dijalankan setelah fase sebelumnya telah diselesaikan (Riski Ramadan, 2023). Adapun tahapan yang dilalui oleh model SDLC *waterfall* meliputi 5 fase seperti yang ditunjukkan oleh Gambar 3.1.



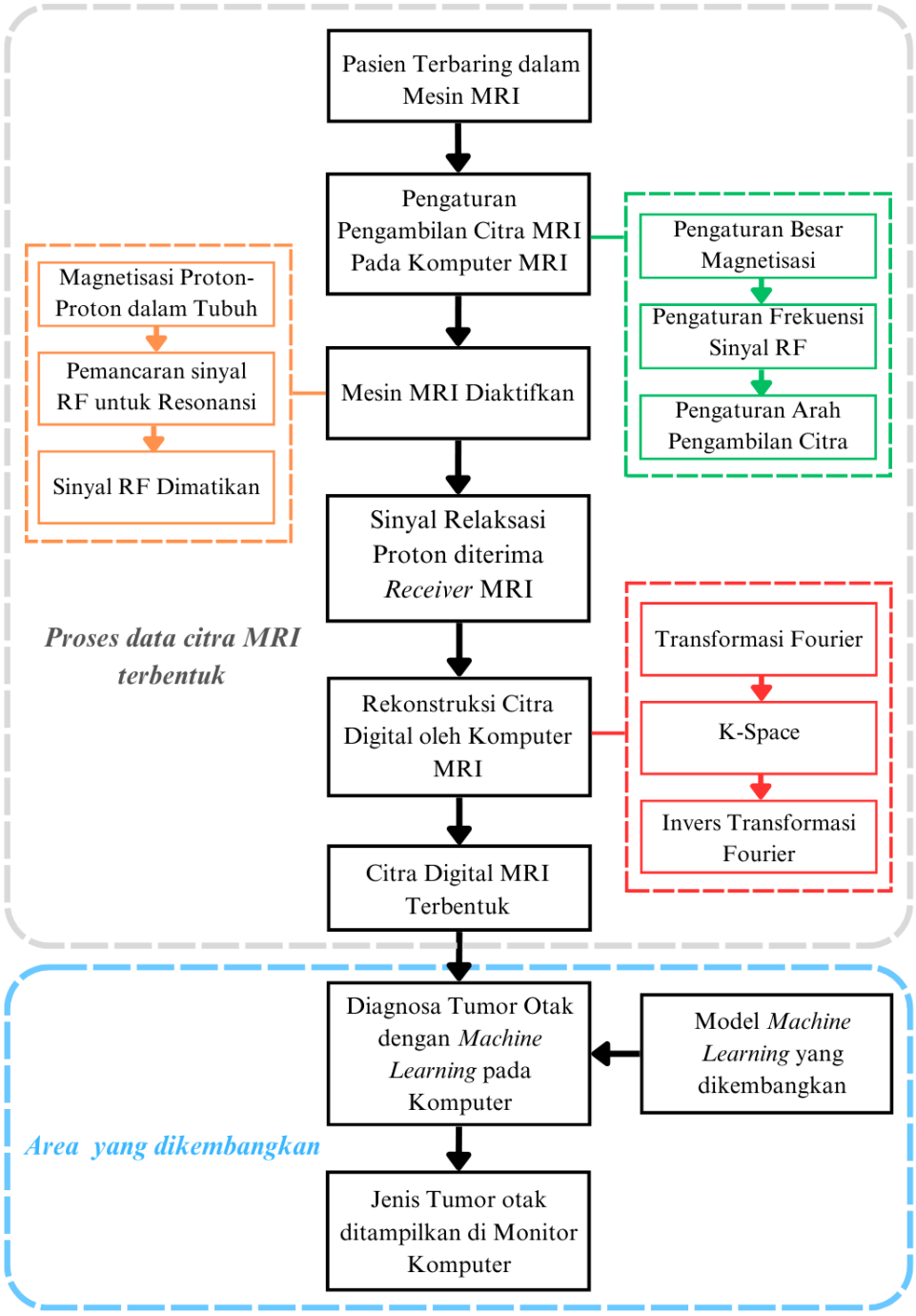
Gambar 3.1 Struktur penelitian SDLC *waterfall*

Keterangan :

1. Fase Analisis : Fase yang melibatkan pengumpulan serta analisis persyaratan dari pengguna
2. Fase Desain : Fase yang melibatkan pembuatan desain sangat rinci dari perangkat lunak
3. Fase Implementasi: Fase yang melibatkan pengkodingan pembuatan sistem dari perangkat lunak
4. Fase Pengujian : Fase untuk menguji perangkat lunak serta memastikan perangkat lunak yang dibuat memenuhi persyaratan

5. Fase Pemeliharaan : Fase yang melibatkan perbaikan masalah yang ada pada sistem perangkat lunak dan pembaruan dari perangkat lunak sesuai kebutuhan.

3.1 Fokus Penelitian



Gambar 3.2 Diagram keseluruhan proses yang bekerja dalam deteksi tumor otak

Gambar 3.2 menunjukkan alur diagram keseluruhan proses bagaimana mesin MRI dapat bekerja untuk mendiagnosa tumor otak jika *machine learning* dijalankan dalam komputer MRI. Pada penelitian ini berfokus pada bagian pengembangan model *machine learning* yang akan dijalankan pada komputer MRI untuk membantu mendiagnosa jenis tumor otak dari hasil citra MRI yang telah terbentuk.

Tahapan pada diagram ini dimulai dari sebelum mesin MRI bekerja, harus ada beberapa prosedur yang harus dilakukan, pasien harus terlebih dahulu berada dalam mesin MRI dengan telah memenuhi prosedur yang dijelaskan pada subbab 3.2 bagian 1, kemudian dilakukan pengaturan citra yang ingin diperoleh pada mesin MRI dengan langkah menentukan besar medan magnet dari mesin MRI. Ditentukan juga terlebih dahulu rentang frekuensi sinyal RF untuk membatasi area tubuh mana saja yang ingin dilakukan pencitraan MRI, setelah itu ditentukan juga arah yang ingin dilakukan pencitraan, pada MRI terdapat pilihan arah sagital, coronal dan aksial, dalam penelitian ini arah pencitraan yang dipakai adalah arah aksial yang menghasilkan citra MRI otak dari arah atas.

Setelah pengaturan pengambilan citra MRI telah dilakukan, selanjutnya mesin MRI dapat diaktifkan, saat mesin MRI diaktifkan, sejumlah rangkaian proses pengambilan citra pada mesin MRI bekerja yang diantaranya magnetisasi proton dalam tubuh untuk menyelaraskan arah spin dari proton, kemudian pemancaran sinyal RF yang dilakukan untuk merubah arah magnetisasi proton dan penonaktifkan sinyal RF untuk mendapatkan sinyal relaksasi dari proton-proton dalam tubuh yang akan diterima oleh *receiver* dari mesin MRI yang lebih spesifiknya proses ini dijelaskan pada subbab 2.2 tentang pencitraa MRI dalam tubuh manusia.

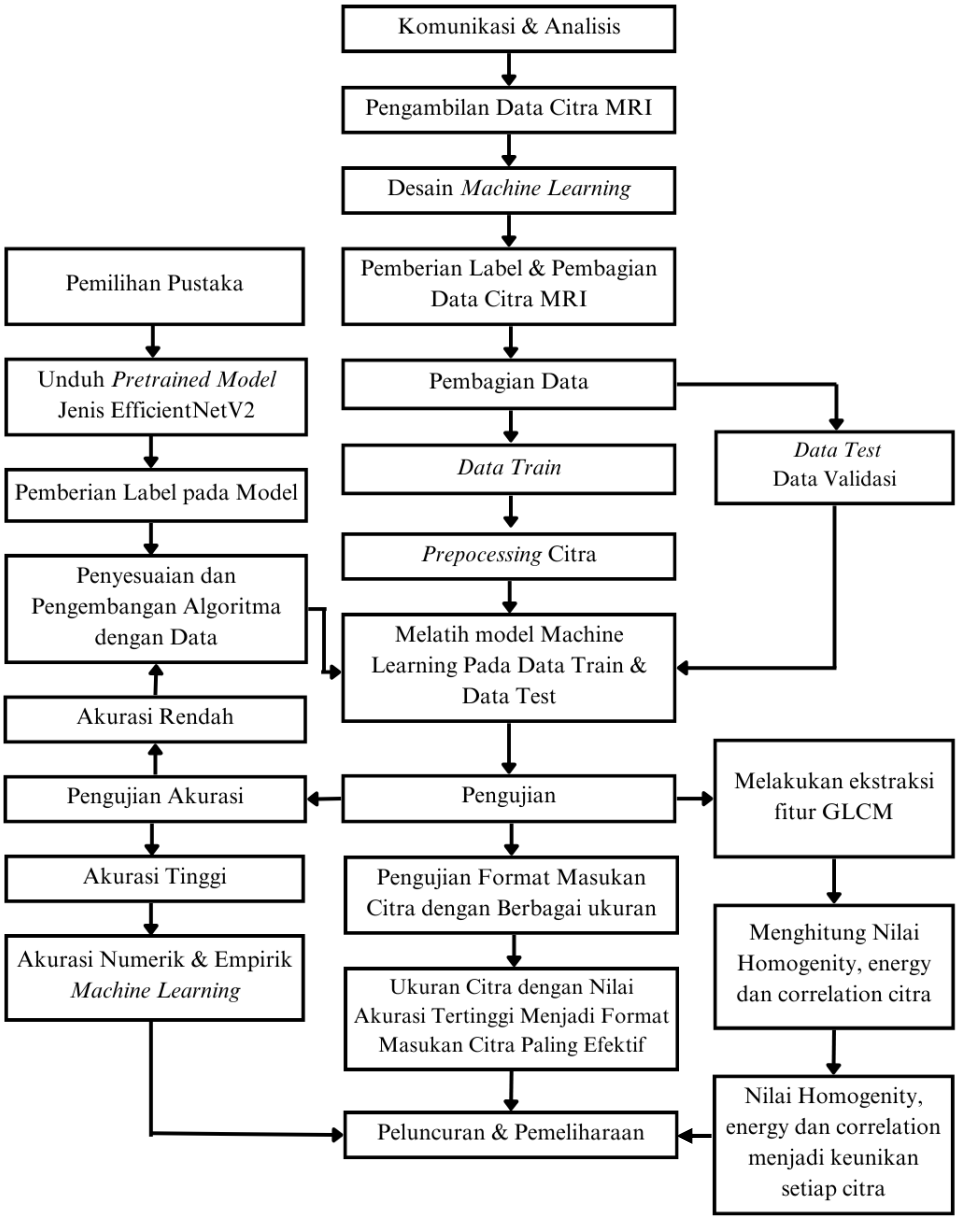
Sinyal relaksasi proton-proton yang diterima oleh mesin MRI selanjutnya dilakukan pengolahan sinyal dalam tahap rekonstruksi citra yang dilakukan oleh komputer MRI, sejumlah proses yang terjadi dalam proses ini adalah dimulai dari pengubahan domain sinyal yang dilakukan oleh transformasi fourier sehingga diperoleh data-data dari gelombang yang data-data tersebut yang disimpan dalam matriks K-space dan dilakukan invers transformasi fourier kembali untuk

mengembalikan domain dari sinyal sehingga diperoleh citra MRI 2D, proses ini lebih spesifiknya dijelaskan pada subbab 2.3 tentang pengolahan dan rekonstruksi citra MRI 2D

Citra otak digital yang telah terbentuk dalam bentuk file gambar selanjutnya dilakukan prediksi jenis tumor otak oleh *machine learning* sebagai diagnosa awal sebelum selanjutnya di verifikasi oleh ahli radiologi atau dokter. Diagnosa tumor otak oleh *machine learning* ini dapat bekerja pada komputer MRI dengan menjalankan model *machine learning* yang dikembangkan pada penelitian ini di *software machine learning* yang dikembangkan, hal ini bisa terjadi karena model *machine learning* yang dikembangkan pada penelitian ini dapat disimpan dalam bentuk file HDF5 yang bisa dijalankan pada kode editor di komputer, setelah citra dilakukan prediksi oleh *machine learning*, jenis tumor otak dapat ditampilkan pada monitor komputer.

3.2 Desain Penelitian

Dalam penelitian ini Gambar 3.3 menunjukkan proses yang akan dilalui dari dalam penelitian pengembangan *Machine learning* untuk mendiagnosa serta mengklasifikasikan penyakit otak dari hasil citra MRI. Langkah awal dalam proses ini adalah menganalisis karakterisik apa saja yang diperlukan oleh *Machine learning* dalam mengolah citra MRI untuk dapat mendiagnosa jenis penyakit tumor otak.



Gambar 3.3 Desain penelitian yang akan dilakukan

Karakteristik pertama yang dibutuhkan dalam penelitian ini adalah keunikan citra, yang memungkinkan citra dapat diproses oleh *Machine learning*. Setiap label atau setiap jenis penyakit tumor otak tentunya memiliki ciri khas sendiri, terutama dalam array pixel masing-masing label. Karakteristik kedua yang dibutuhkan yaitu akurasi prediksi dari *Machine learning* yang dikembangkan dalam mengklasifikasikan penyakit tumor otak, karakteristik ini menjadi indikator apakah algoritma *machine learning* tersebut layak untuk digunakan. Model *machine*

learning yang digunakan pada penelitian ini merupakan model yang sudah dibuat sebelumnya, namun model *Machine learning* yang digunakan pada penelitian ini memiliki efektivitas serta akurasi yang masih dapat ditingkatkan.

3.3 Alat dan Bahan

Untuk melakukan pengembangan algoritma *machine learning,* pada penelitian ini digunakan sejumlah alat dan bahan yang diantaranya tercantum dalam sub-subab berikut ini :

3.3.1 Mesin MRI

Pada penelitian ini mesin MRI menjadi alat yang digunakan untuk menghasilkan citra dari otak manusia. Prosedur yang dilalui dalam mengambil data citra otak manusia dimulai dari proses wawancara terhadap pasien untuk mengetahui keluhan dari pasien dan menentukan pasien tersebut layak dilakukan pemeriksaan MRI atau tidak, selanjutnya pasien tersebut akan diminta untuk berbaring di meja pasien, setelah posisi pasien sudah benar, selanjutnya pasien dimasukkan ke dalam mesin MRI, kemudian mesin MRI dinyalakan. Ketika pasien di dalam mesin MRI, pemeriksa atau dokter mengatur citra yang akan ditangkap pada komputer, pemeriksa mengatur jumlah *slice* yang akan digunakan, juga menentukan koordinat yang akan diambil citranya masing-masing pada arah sagittal, koronal dan aksial, setelah itu diperoleh file citra MRI berupa citra jenis T1,T2 dan T1C+ pada arah sagittal, koronal dan aksial. Hasil dari citra ini kemudian diperiksa oleh dokter atau ahlinya untuk menentukan kondisi dari otak pasien, dalam menentukan kondisi dari otak, pengembangan fitur pada komputer MRI dapat dilakukan untuk membantu diagnosa kondisi otak manusia.

3.3.2 Pemrograman *python*

Alat lain yang digunakan pada penilitian ini adalah bahasa pemrograman *Python,* yang digunakan sebagai bahasa pemrograman dalam membangun sistem *Machine learning* dalam penelitian ini. Alasan digunakan bahasa pemrograman *Python* adalah karena memiliki sejumlah

keunggulan dalam pengembangan model *Machine learning* seperti fleksibilitas yang tinggi, keanekaragaman pustaka, dan dukungan yang baik terhadap analisis serta visualisasi data (Fardilasoliha, 2023). Versi python yang digunakan pada penelitian ini adalah 3.10.12. Ada beberapa pustaka yang digunakan pada penelitian ini, diantaranya:

- Pandas : Pustaka untuk memanipulasi serta analisis data dalam bentuk DataFrame dan series
- Numpy : Pustaka untuk memnipulasi serta analisis data dalam bentuk array dan matriks
- Matplotlib : Pustaka untuk melakukan visualisasi data
- Keras : Pustaka untuk membangun dan melatih model *neural network*. Merupakan high-level neural network API yang berjalan di atas TensorFlow
- Scikit-Learn : Pustaka untuk pemodelan data dan analisis statistik
- Tensorflow : Pustaka untuk mengembangkan dan melatih *machine learning* yang menyediakan berbagai alat dan pustaka untuk membangun dan melatih model *deep learning*

Google colab dipakai pada penelitian ini untuk menjalankan pemrograman python. Google colab merupakan sebuah layanan notebook online yang memungkinkan penggunanya untuk menulis, menjalankan serta membagikan kode python di browser web, dengan menggunakan konsep *virtual machine* sehingga perangkat hardware seperti CPU dan GPU disediakan secara online oleh Google cloud server untuk menjalankan kode yang dibuat oleh pengguna. Platform ini merupakan sebuah patform yang sangat familiar dipakai oleh para pengembang *machine learning.*

3.3.3 Dataset pencitraan penyakit otak MRI

Dalam penelitian ini untuk membangun model *machine learning* untuk klasifikasi penyakit otak digunakan dataset pencitraan penyakit otak MRI sebagai data pelatihan, data untuk testing model yang dibangun serta data untuk validasi model yang dibangun.

Dataset primer yang diambil dalam penelitian ini berasal dari situs Kaggle, dipublikasi oleh ahli teknisi radiologi di GRS Imagem Cruz Alta Brazil yang ditujukan untuk penelitian, yang dalam keterangan metadatanya menunjukkan bahwa dataset ini telah diuji serta ditafsirkan oleh ahli radiologi, kemudian dataset ini memiliki sertifikasi lisensi sehingga kredibilitasnya bisa dipertanggungjawabkan. Kemudian dataset sekunder digunakan pada penelitian ini untuk mencegah *imbalance data*, penambahan citra otak normal ditambahkan pada dataset penelitian ini dengan menggunakan dataset dari SARTAJ yang sudah memiliki sertifikasi lisensi serta juga telah digunakan pada berbagai penelitian, dan pengurangan data terhadap data kelas yang *oversampling* juga dilakukan pada penelitian ini untuk menyeimbangkan jumlah data yang akan dilakukan *training* pada *machine learning*.

Dataset ini berisi hasil citra tumor otak dengan 5 kelas yaitu Glioma, Neurocytoma, Meningioma, Schawannoma dan penyakit otak lainnya dengan hasil citra MRI jenis T1, T2 dan T1C+ kemudian hasil citra yang digunakan merupakan citra yang berarah aksial. Jumlah dataset yang digunakan pada penelitian ini berjumlah 4219 data yang pembagian datanya ada pada Tabel 3.1 :

Tabel 3.1 Jumlah data masing-masing penyakit yang akan dipakai pada penelitian

No	Jenis Tumor Otak	Jumlah
1	Glioma	1231
2	Meningioma	1226
3	Neurocytoma	527
4	Normal	489
5	Schwannoma	446
Jumlah		**4219**

3.4 Desain Algoritma *Machine learning* Untuk Klasifikasi Jenis Tumor Otak hasil MRI

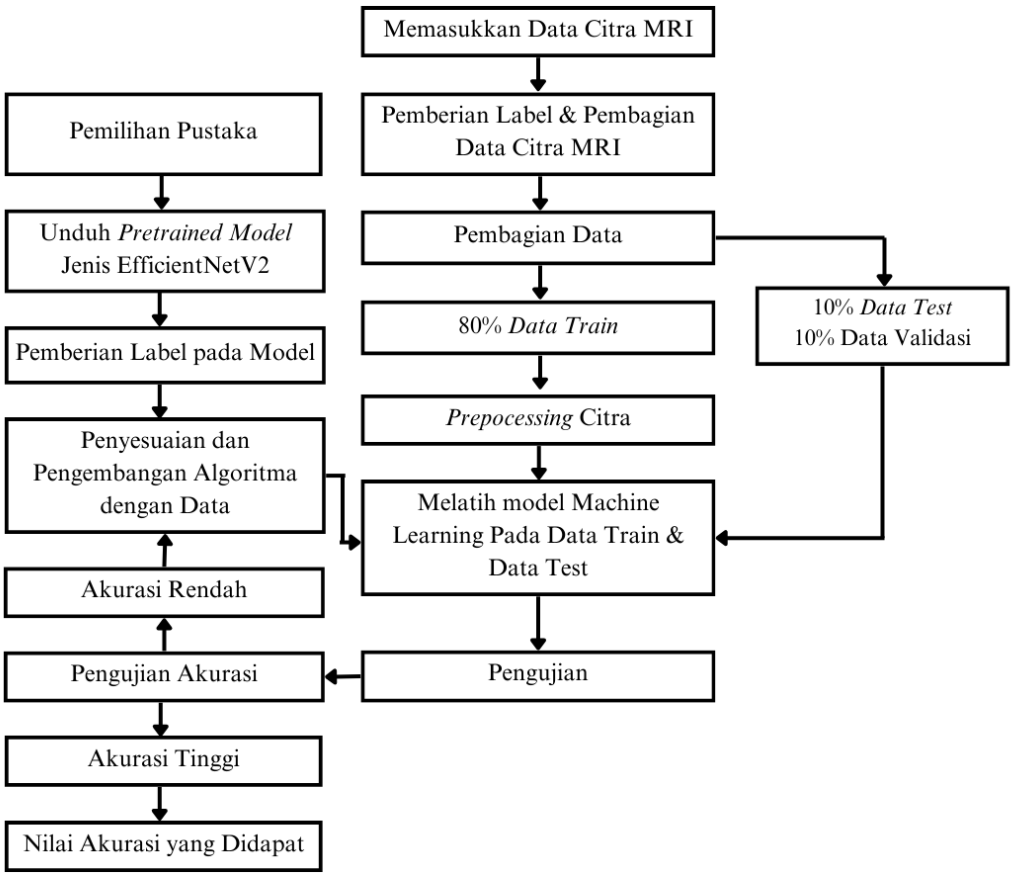
Pada subbab ini akan dijelaskan bagaimana proses perencanaan yang akan dilakukan setiap tahapannya dalam membangun algoritma *Machine learning* pada citra hasil MRI untuk diagnosa penyakit tumor otak. Proses ini dimulai dari ketika citra MRI dari otak manusia diperoleh sampai dengan *Machine learning* dapat mendiagnosa keaadaan otak manusia dari hasil MRI.

3.4.1 Desain Algoritma *Machine learning* untuk Mengetahui Keunikan Citra dari Setiap Label

Setiap hasil citra MRI otak manusia memiliki ukuran atau resolusi gambar yang tersusun atas pixel-pixel yang membentuk struktur gambar otak manusia. Setiap pixel citra memiliki nilai kedalaman warna yang menjadi informasi bagi *machine learning* untuk mengklasifikasikan citra. Pada penelitian ini langkah awal yang dilakukan agar keunikan citra dapat diketahui adalah dengan melakukan analisa ekstraksi fitur GLCM (*Gray Level Cooccurrence Matrix*) untuk mengetahui pola persebaran pixel pada setiap citra jenis tumor otak sehingga citra setiap jenis tumor otak dapat diketahui nilai khasnya untuk bisa dibedakan.

3.4.2 Desain Algoritma untuk Mengetahui Akurasi *Machine learning* Pengolahan Citra MRI untuk Klasifikasi Jenis Tumor Otak

Dalam penelitian ini untuk mengetahui akurasi akhir dari algoritma *Machine learning*, dilalui sejumlah tahapan yang secara garis besar dimulai dari data berupa citra hasil MRI dimasukkan kedalam *google drive* agar data disimpan dalam *cloud* sehingga lebih efisen ketika data akan dipakai dalam pengembangan model *machine learning*, *prepocessing* data dengan melakukan beberapa pengaturan terhadap data citra sebelum diolah, pemilihan pustaka algoritma dasar yang akan dipakai, *processing* data dengan memasangkan algoritma yang dikembangkan terhadap citra MRI sampai dengan dilakukan pengujian untuk mengetahui akurasi terbaik yang dicapai dalam penelitian ini, secara lebih detailnya tahapan yang akan dilalui terdapat pada Gambar 3.4.



Gambar 3.4 Alur untuk mengetahui nilai akurasi terbaik pada machine model *machine learning* yang dibuat

3.4.3 Desain Algoritma untuk Menentukan Resolusi Citra MRI Paling Efektif dalam pengolahan *Machine learning* untuk Klasifikasi Jenis Tumor Otak

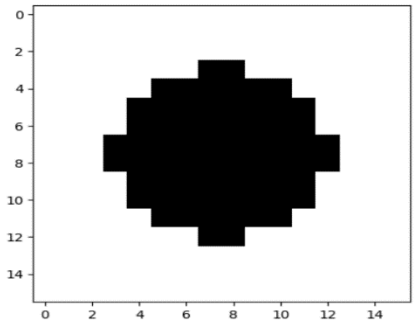
Untuk melihat format ukuran citra yang paling efektif sehingga memperoleh akurasi *machine learning* yang tinggi diperlukan pengujian dengan melakukan berbagai variasi ukuran citra yang akan coba dilatih oleh model *machine lerning* yang dibuat dan dilihat ukuran citra mana yang menghasilkan nilai akurasi serta nilai validasi yang tinggi. Dalam penelitian ini akan dilakukan uji coba dengan mengubah ukuran citra dengan ukuran 128x128, 256x256, 384x384 dan 512x512 pixel. Hasil pengujian ini akan ditampilkan ke dalam sebuah tabel untuk mempermudah analisa pengaruh ukuran citra terhadap model *machine learning* yang dikembangkan.

3.5 Pembangunan *Machine learning* Pengolah Citra MRI dalam Klasifikasi Jenis Tumor Otak

Pada subbab ini akan dijelaskan bagaimana implementasi secara teknis yang akan dilakukan dalam pembuatan algoritma *machine learning* dari desain penelitian yang telah dijelaskan pada subbab 3.3. Sintaks pemrograman untuk membangun *machine learning* setiap tahapannya akan ditampilkan pada subbab ini.

3.5.1 Proses Memperoleh Pola Persebaran *Array* yang Menjadi Keunikan Citra dari Setiap Label

Langkah untuk mengetahui pola persebaran pixel pada citra yang khas pada setiap jenis tumor otak dimulai dengan mengkonversi kedalaman warna ke dalam *array* berbentuk matriks sehingga dapat diolah oleh komputer seperti pada gambar 3.5, setelah itu dilakukan *prepocessing* data terlebih dahulu dengan menerapkan filter *threshold* untuk membersihkan citra tumor otak dengan area otak disekitarnya yang bekerja dengan memasang nilai ambang batas nilai *array* sehingga pixel yang memiliki *array* yang melebihi ambang batas akan terseleksi, jika area tumor sudah terpisahkan maka dilanjutkan dengan menganalisa pola persebaran pixel dengan menganalisa fitur-fitur pada metode ekstraksi fitur GLCM, adapun sintaks untuk melakukan ekstraksi fitur ini adalah sebagai berikut




```
[[255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255]
 [255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255]
 [255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255]
 [255 255 255 255 255 255 255   0   0 255 255 255 255 255 255 255]
 [255 255 255 255 255   0   0   0   0   0   0 255 255 255 255 255]
 [255 255 255 255   0   0   0   0   0   0   0   0 255 255 255 255]
 [255 255 255 255   0   0   0   0   0   0   0   0 255 255 255 255]
 [255 255 255   0   0   0   0   0   0   0   0   0 255 255 255 255]
 [255 255 255   0   0   0   0   0   0   0   0   0 255 255 255 255]
 [255 255 255 255   0   0   0   0   0   0   0   0 255 255 255 255]
 [255 255 255 255   0   0   0   0   0   0   0   0 255 255 255 255]
 [255 255 255 255 255   0   0   0   0   0   0 255 255 255 255 255]
 [255 255 255 255 255 255 255   0   0 255 255 255 255 255 255 255]
 [255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255]
 [255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255]
 [255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255]]
Ukuran matriks :  (16, 16)
```

Gambar 3.5 Konversi warna ke dalam matriks pada *machine learning*

Sourcode untuk ekstraksi fitur citra adalah sebagai berikut :

```
import numpy as np
import matplotlib.pyplot as plt
import cv2
from skimage.feature import greycomatrix, greycoprops

# Memuat gambar MRI
img_mri = cv2.imread('File Citra MRI.jpeg',
cv2.IMREAD_GRAYSCALE)

# Menentukan ambang batas
threshold = 160

# Binarisasi gambar
img_bw = np.where(img_mri > threshold, 255,
0).astype(np.uint8)

# Hitung GLCM
distances = [1]  # Jarak piksel untuk membuat GLCM
angles = [0]  # Sudut untuk membuat GLCM
glcm = greycomatrix(img_bw, distances=distances,
angles=angles, levels=256, symmetric=True, normed=True)

# Menghitung fitur GLCM
properties = ['ASM', 'contrast', 'dissimilarity',
'homogeneity', 'energy', 'correlation']

features = [greycoprops(glcm, prop).ravel() for prop in
properties]

# Menampilkan hasil
for prop, feature in zip(properties, features):
    print(f'{prop.capitalize()}:', feature)
```

3.5.2 Proses Memperoleh Akurasi *Machine learning* Pengolahan Citra MRI dalam Klasifikasi Jenis Tumor Otak

Seperti pada Gambar 3.4, untuk mengetahui nilai akurasi akhir dari pengembangan *machine learning* yang dibuat ini, diperlukan beberapa tahap dari hasil citra masuk sampai dengan diperoleh nilai akurasinya, penjelasan

mengenai implementasi dan bagaimana komputer mengolah citra pada setiap tahapannya seperti pada Gambar 3.4 adalah sebagai berikut :

1. Pemilihan pustaka

   Pustaka dalam konteks *machine learning* merupakan sebuah perpustakaan kode yang di dalamnya memuat beberapa fungsi dana algoritma yang sudah jadi sehingga lebih efisien dalam pemakainnya. Diperlukan pemanggilan pustaka diawal ketika memprogram supaya isi dari pustaka dapat dipakai. Berikut adalah bagaimana pemanggilan pustaka pada penelitian ini dilakukan :

```
import tensorflow as tf
from tensorflow.keras.applications.efficientnet_v2
import EfficientNetV2S
from tensorflow.keras.preprocessing.image import
ImageDataGenerator
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import os
import cv2
from tqdm import tqdm
import glob
from imblearn.over_sampling import SMOTE
```

2. Pemberian label

   Machine learning bekerja dengan mempelajari data yang sudah ada, untuk itu pemberian label nama penyakit otak pada setiap citra harus dilakukan, pada penelitian ini label nama diklasifikasikan menjadi 5 jenis yaitu Normal, *Schwannoma*, *Meningioma, Neurocytoma* dan *Glioma*, Adapun sintaks untuk pemberian label nama adalah sebagai berikut :

```
data_train = "/content/drive/MyDrive/Skripsi/All
Data/Data Training & Data Testing"
classes = sorted(os.listdir(data_train))
jenis_tumor = 1 , 2 , 3 , 4, 5
for i in jenis_tumor :
   print(classes[i])
```

3. Pembagian data

Pada proses ini data dibagikan menjadi data train, data test dan data validasi. Adapun pembagian data pada penelitian ini secara lebih jelas ada pada tabel berikut :

Jenis	Jumlah	keterangan
Data Train	80%	Data untuk dilatih dengan model algoritma yang dibuat
Data Test	10%	Data yang tidak dilatih, dan untuk menguji akurasi *machine learning*
Data Validation	10%	Termasuk data yang dilatih, digunakan untuk menguji akurasi

Tabel 3.2 Pembagian data yang akan digunakan pada penelitian

Berikut adalah sintaks untuk melakukan pembagian data :

```
train_ratio = 0.8
test_ratio = 0.1
validation_ratio = 0.1

X_train, X_test_val, y_train, y_test_val =
train_test_split(image_paths, labels,
test_size=test_ratio + validation_ratio,

stratify=labels, random_state=42)
validation_ratio = validation_ratio / (test_ratio +
validation_ratio)

X_test, X_val, y_test, y_val =
train_test_split(X_test_val, y_test_val,
test_size=validation_ratio,
stratify=y_test_val, random_state=42)
```

4. *Prepocessing data*

Sebelum citra MRI dilatih, citra MRI dilakukan penyesuaian gambar terlebih dahulu supaya bisa memperoleh nilai akurasi yang tinggi serta juga lebih cepat dalam pelatihan model. *Prepocessing data* pada penelitian ini terdiri atas *Resizing* yaitu merubah ukuran format resolusi seluruh citra menjadi satu resolusi, pada penelitian ini resolusi yang dipakai adalah 256 x 256 pixel, kemudian melakukan normalisasi data agar berada pada rentang kedalaman warna yang sama yaitu 0 sampai 255, selanjutnya adalah melakukan konversi warna, pada penelitian ini digunakan konversi warna bertipe *grayscale* dibandingkan dengan RGB, karena citra MRI memiliki

warna hitam putih saja sehingga tidak diperlukan data warna lainnya, selain itu konversi warna jenis *grayscale* lebih cepat dalam pelatihan model karena data yang diolah lebih sedikit dibandingkan dengan konversi warna jenis RGB.

Adapun sintaks dalam tahap *preprocessing data* adalah sebagai berikut

```
# Menentukan ukuran gambar
image_size = 256

# Memuat data latih
X_train = []
y_train = []

for i, cls in enumerate(classes):
    # Mendapatkan daftar file gambar
    image_paths = glob.glob(os.path.join(data_train,
cls,"*.*"))

    for img_path in tqdm(image_paths):
        # Membaca gambar
        img = cv2.imread(img_path)

        # Mengubah ukuran gambar
        img = cv2.resize(img, (image_size, image_size))

        # Menambahkan data gambar dan label
        X_train.append(img)
        y_train.append(i)

# Mengubah data menjadi array NumPy
X_train = np.array(X_train)
y_train = np.array(y_train)

# Mencetak informasi data
print('Data latih:', X_train.shape, y_train.shape)

datagen = ImageDataGenerator(
    rotation_range= 30,
    width_shift_range = 0.1,
    height_shift_range = 0.1,
    zoom_range = 0.2,
    horizontal_flip = True)

datagen.fit(X_train)
X_train.shape
```

5. Pelatihan model

Pada tahap ini dilakukan pelatihan data citra otak citra MRI pada algoritma *machine learning* yang sudah dikembangkan, jenis *algoritma machine learning* yang digunakan pada penelitian ini adalah algoritma EfficientNetV2, digunakan algoritma ini dengan alasan karena algoritma ini merupakan algoritma paling baru saat ini yang lebih cepat serta efisien dalam pelatihan model.

Adapun sintaks untuk pelatihan model adalah sebagai berikut :

```
base_model = EfficientNetV2S(
    include_top=False,
    weights="imagenet",
    input_shape=(image_size,image_size,  3)
)
from keras.optimizers import Nadam

opt = Nadam(lr=0.002, beta_1=0.9, beta_2=0.999)

base_model.compile(optimizer=opt,
loss="categorical_crossentropy", metrics=["accuracy"])

h = model_new.fit(X_train, y_train, epochs=10,
validation_data=(X_vall, y_vall), batch_size=50,
verbose=1, shuffle=1)
```

6. Pengujian

Pengujian dilakukan untuk mengetahui seberapa akurat algoritma *machine learning* dapat memprediksi penyakit dari hasil citra MRI. Pada tahap ini digunakan *confussion matrix* yaitu sebuah tabel yang digunakan untuk mengevalusai hasil dari pelatihan model, *confussion matriks* dari penelitian ini digambarkan pada Gambar 3.6

Tabel 3.2 Confussion Matriks

Label sebenarnya	Label 1	Label 2	Label 3	Label 4	Label 5
Label 1	**TP1**	FP2	FP3	FP4	FP5
Label 2	FN1	**TP2**	FP3	FP4	FP5
Label 3	FN1	FN2	**TP3**	FP4	FP5
Label 4	FN1	FN2	FN3	**TP4**	FP5
Label 5	FN1	FN2	FN3	FN4	**TP5**

Keterangan :

- TN (*True Negative*) : Output label *negative* yang berhasil ditebak sebagai label *negative*
- FN (*False Negative*) : Output label *positive* yang salah ditebak sebagai kelas *Negative*
- FP (*False Positive*) : Output label *negative* yang salah ditebak sebagai kelas *positive*

Dari tabel *confussion matrix* kita dapat menghitung beberapa metrik evaluasi untuk menilai kinerja dari pelatihan model yang diantaranya

- Akurasi : (TP1+TP2+TP3+TP4+TP5) / Total
- Presisi : TP / (TP + FP) untuk setiap kelas
- Recall : TP / (TP + FN) untuk setiap kelas
- F1-Score : 2 * (Presisi *Recall) / (Presisi + Recall) untuk setiap kelas

Adapun sintaks untuk membuat *confussion matriks* dan metrik-metrik evaluasi adalah sebagai berikut :

```
# Make predictions on the test data
y_pred = model_new.predict(X_vall)

# Get the class labels
class_names = classes

# Calculate the confusion matrix
cm = confusion_matrix(y_vall.argmax(axis=1),
y_pred.argmax(axis=1))

# Option 1: Using Matplotlib
# Normalize the confusion matrix for better
visualization
cm_norm = cm.astype('float') / cm.sum(axis=1)[:,
np.newaxis]

plt.figure(figsize=(8, 6))
plt.imshow(cm_norm, cmap=plt.cm.Blues)
plt.colorbar()

# Add labels and title
plt.xticks(range(len(class_names)), class_names,
rotation=45)
plt.yticks(range(len(class_names)), class_names)
plt.title('Confusion Matrix')

# Add text for each cell
for i in range(len(cm)):
  for j in range(len(cm)):
    plt.text(j, i, cm[i, j], ha='center',
va='center',fontsize=10)

plt.tight_layout()
plt.show()

from sklearn.metrics import accuracy_score,
precision_score, recall_score, f1_score

# Calculate accuracy, precision, recall, and F1-score
accuracy = accuracy_score(y_vall.argmax(axis=1),
y_pred.argmax(axis=1))
precision = precision_score(y_vall.argmax(axis=1),
y_pred.argmax(axis=1), average=None)
```

```
recall = recall_score(y_vall.argmax(axis=1),
y_pred.argmax(axis=1), average=None)
f1 = f1_score(y_vall.argmax(axis=1),
y_pred.argmax(axis=1), average='weighted')

# Print the results
print("Accuracy:", accuracy)
print("F1-Score:", f1)

# Print the results
for i in range(len(classes)):
  print("Class {}: Precision = {}, Recall =
{}".format(classes[i], precision[i], recall[i]))
```

3.5.3 Proses memperoleh Resolusi Citra MRI yang Efektif pada Pengembangan *Machine learning* Klasifikasi Jenis Tumor Otak

Pada proses ini dilakukan pengujian beberapa variasi ukuran citra dengan ukuran 128x128, 256x256, 384x384 dan 512x512 yang akan dilatih dengan *machine learning* yang dibuat dengan variable terikatnya nilai akurasi yang dicapai. Sintaks yang digunakan umumnya sama dengan sintaks untuk mengetahui nilai akurasi dan nilai validasi model *machine learning* yang dikembangkan, namun variabel "image_size" pada tahap *prepocessing* divariasikan satu satu dengan ukuran yang telah ditetapkan, berikut adalah bagian sintaks yang dilakukan variasi :

```
# Menentukan ukuran gambar
image_size = 256

# Memuat data latih
X_train = []
y_train = []

for i, cls in enumerate(classes):
    # Mendapatkan daftar file gambar
    image_paths = glob.glob(os.path.join(data_train,
cls,"*.*"))

    for img_path in tqdm(image_paths):
        # Membaca gambar
```

```
        img = cv2.imread(img_path)

        # Mengubah ukuran gambar
        img = cv2.resize(img, (image_size, image_size))

        # Menambahkan data gambar dan label
        X_train.append(img)
        y_train.append(i)

# Mengubah data menjadi array NumPy
X_train = np.array(X_train)
y_train = np.array(y_train)

# Mencetak informasi data
print('Data latih:', X_train.shape, y_train.shape)

datagen = ImageDataGenerator(
    rotation_range= 30,
    width_shift_range = 0.1,
    height_shift_range = 0.1,
    zoom_range = 0.2,
    horizontal_flip = True)

datagen.fit(X_train)
X_train.shape plt.xlabel("Image Size")
plt.ylabel("Accuracy")
plt.legend()
plt.show()
```

3.6 Pengujian dan Pengolahan Data untuk Memperoleh Karakteristik *Machine learning* Pengolahan Citra MRI dalam klasifikasi Jenis Tumor Otak

Dalam subbab ini akan dijelaskan bagaimana pengolahan data yang akan dilakukan untuk memperoleh karakteristik dari pengembangan model *machine learning* yang dibuat untuk mengklasifikasikan jenis tumor otak.

3.6.1 Analisa Data Hasil Pelatihan Model Untuk Memperoleh Akurasi *Machine learning* Terbaik dalam Klasifikasi Penyakit Tumor Otak

Hasil pengolahan data dalam pengembangan pembuatan model *machine learning* berupa data nilai akurasi keseluruhan, nilai validasi keseluruhan serta nilai presisi dan recall untuk setiap penyakit. Dari data-data tersebut akan menjadi acuan penulis dalam melakukan pengembangan lebih lanjut untuk menemukan format paling baik dalam pelatihan model *machine learning* sehingga memperoleh nilai akurasi, validasi, presisi dan recall yang tinggi. Penulis dapat melakukan variasi terhadap jumlah iterasi kemudian penambahan layer atau pengurangan layer pada algoritma CNN yang dibuat, dan membandingkan variasi yang dilakukan tersebut sehingga mmperoleh nilai akurasi, validari, presisi, recall yang tinggi.

3.6.2 Pengolahan Data untuk memperoleh resolusi citra yang paling efektif

Hasil dari penerapan variasi format ukuran citra adalah berupa nilai akurasi *machine learning* dapat melakukan klasifikasi jenis penyakit tumor otak. Hasil dari pengujian ini akan ditampikan ke dalam sebuah grafik yang nantinya akan dianalisa ukuran format citra mana yang paling efektif untuk dipakai pada model *machine learning* yang dibuat ini.