BAB IV
HASIL DAN PEMBAHASAN

4.1 Load Dataset

Langkah awal dalam penelitian ini yaitu, mempersiapkan dataset sesuai dengan kebutuhan model. Untuk mencapai hal ini, gambar-gambar MRI otak dari empat kategori berbeda telah diunduh dari situs Kaggle. Kategori tersebut adalah 'no_tumor', 'meningioma_tumor', 'glioma_tumor', dan 'pituitary_tumor'. Dataset yang diunduh terbagi menjadi dua folder, yaitu Training dan Testing, dengan total 3264 data.

```
import cv2, os, numpy as np
from tqdm import tqdm

labels = ['no_tumor', 'meningioma_tumor', 'glioma_tumor', 'pituitary_tumor']

X_train = []
y_train = []
image_size = 150

# Fungsi untuk menerapkan CLAHE pada gambar
def apply_clahe(img):
    lab = cv2.cvtColor(img, cv2.COLOR_BGR2LAB)
    l, a, b = cv2.split(lab)
    clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8))
    cl = clahe.apply(l)
    limg = cv2.merge((cl, a, b))
    final_img = cv2.cvtColor(limg, cv2.COLOR_LAB2BGR)
    return final_img

for i in labels:
    folderPath = os.path.join('Training', i)
    for j in tqdm(os.listdir(folderPath)):
        img = cv2.imread(os.path.join(folderPath, j))
        img = cv2.resize(img, (image_size, image_size))
        img = apply_clahe(img)  # Menerapkan CLAHE
        X_train.append(img)
        y_train.append(i)

for i in labels:
    folderPath = os.path.join('Testing', i)
    for j in tqdm(os.listdir(folderPath)):
        img = cv2.imread(os.path.join(folderPath, j))
        img = cv2.resize(img, (image_size, image_size))
        img = apply_clahe(img)  # Menerapkan CLAHE
        X_train.append(img)
        y_train.append(i)

X_train = np.array(X_train)
y_train = np.array(y_train)
```

Gambar 1. Implementasi Load Dataset

4.2 Pre-Processing

Pada tahap ini, pre-prorocessing yang diterapkan yaitu resize image dari 512x512 pixel menjadi 150x150 pixel yang berguna untuk mengurangi perbedaan dalam data input dan beban komputasi, sehingga mempermudah model untuk belajar lebih efektif serta mengurangi risiko overfitting. Clahe atau Contrast Limited Adaptive Histogram Equalization diterapkan guna untuk meningkatkan kontras gambar sebelum input ke arsitektur model seperti InceptionResNetV2.

Dengan menggunakan CLAHE, gambar diproses untuk meningkatkan kejelasan dan detailnya, memungkinkan model untuk lebih efektif mempelajari fitur-fitur yang relevan. Kontras yang ditingkatkan sangat berarti dalam tugas-tugas seperti deteksi tepi dan segmentasi objek, di mana kemampuan model untuk membedakan antara berbagai bagian gambar sangat penting. Dengan demikian, penggunaan CLAHE sebagai langkah prapemrosesan dapat secara signifikan mendukung kinerja dan akurasi model seperti InceptionResNetV2 dalam mengatasi tugas-tugas pengolahan gambar yang kompleks.

```python
# Fungsi untuk menerapkan CLAHE pada gambar
def apply_clahe(img):
    lab = cv2.cvtColor(img, cv2.COLOR_BGR2LAB)
    l, a, b = cv2.split(lab)
    clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8))
    cl = clahe.apply(l)
    limg = cv2.merge((cl, a, b))
    final_img = cv2.cvtColor(limg, cv2.COLOR_LAB2BGR)
    return final_img

for i in labels:
    folderPath = os.path.join('Training', i)
    for j in tqdm(os.listdir(folderPath)):
        img = cv2.imread(os.path.join(folderPath, j))
        img = cv2.resize(img, (image_size, image_size))
        img = apply_clahe(img)  # Menerapkan CLAHE
        X_train.append(img)
        y_train.append(i)

for i in labels:
    folderPath = os.path.join('Testing', i)
    for j in tqdm(os.listdir(folderPath)):
        img = cv2.imread(os.path.join(folderPath, j))
        img = cv2.resize(img, (image_size, image_size))
        img = apply_clahe(img)  # Menerapkan CLAHE
        X_train.append(img)
        y_train.append(i)

X_train = np.array(X_train)
y_train = np.array(y_train)
```

Gambar 2. Preprocessing CLAHE

4.3 Data Splitting

Tahap data splitting dilakukan pada dataset yang digunakan dalam penelitian ini terdiri dari total 3264 data, untuk membagi data menjadi tiga bagian secara proporsional. Sebanyak 80% dari dataset digunakan untuk data training, 10% untuk data validation, dan 10% untuk data testing. Data validation digunakan untuk menguji dan memvalidasi model yang telah dilatih dengan dataset training. menggunakan fungsi train_test_split dari pustaka sklearn.model_selection untuk membagi dataset menjadi dua bagian utama: data training dan data testing. Dataset ini terdiri dari input (X_train) dan target (y_train), yang merupakan label dari dataset yang akan dibagi. Dengan menentukan test_size=0.1, 10% dari total data untuk menjadi data testing, sementara 90% sisanya digunakan sebagai data training

untuk melatih model. Penggunaan random_state=123 memastikan bahwa pembagian data dilakukan secara konsisten, sehingga hasilnya dapat diprediksi ulang dengan cara yang sama pada setiap percobaan. Data yang sudah terbagi, yaitu X_train (input untuk training), X_test (input untuk testing), y_train (label untuk training), dan y_test (label untuk testing), akan digunakan untuk mengembangkan dan menguji performa model dengan data yang belum pernah dilihat sebelumnya.

```
from sklearn.model_selection import train_test_split

X_train,X_test,y_train,y_test = train_test_split(X_train,y_train, test_size=0.1,random_state=123)
```

Gambar 3. Data Splitting

4.4 Data Augmentation

Proses augmentasi data bertujuan untuk memperluas dan memperkaya dataset dengan cara memodifikasi gambar asli menjadi berbagai variasi yang berbeda. Sehingga, menjadikan performa klasifikasi model dapat meningkat secara signifikan karena model mampu mengenali pola dan fitur yang lebih beragam dalam data latihannya. Hal ini sangat penting dalam meningkatkan akurasi dan generalisasi model, sehingga mampu memberikan hasil yang lebih baik pada data yang belum pernah dilihat sebelumnya.

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
# Data augmentation
datagen = ImageDataGenerator(
    rotation_range=25,
    width_shift_range=0.01,
    height_shift_range=0.01,
    zoom_range=0.1,
    horizontal_flip=True,
    validation_split=0.111,
)
```

Gambar 4. Augmentasi Data

Dalam penelitian ini, ImageDataGenerator dari pustaka Keras digunakan untuk melakukan augmentasi data pada gambar. Parameter rotation_range diterapkan untuk mengatur variasi rotasi gambar hingga 25 derajat, yang menjadikan variasi posisi sudut dari gambar asli. Selanjutnya, width_shift_range dan height_shift_range membuat jangkauan pergeseran horizontal dan vertikal pada 1% dari ukuran gambar, menjadikan gambar untuk diposisikan dengan sedikit

perbedaan dari posisi aslinya. zoom_range, yang diatur pada 0.1, memperbolehkan gambar untuk diperbesar atau diperkecil hingga 10% dari dimensi aslinya. Dengan mengaktifkan horizontal_flip=True, gambar dapat diputar secara horizontal secara acak, sehingga menambah variasi lebih lanjut. Terakhir, parameter validation_split=0.111 sebagai persentase data yang akan digunakan sebagai data validasi, yaitu 11.1% dari total dataset yang tersedia.

4.5 Model InceptionResNetV2

Pada penelitian ini, base_model (InceptionResNetV2) yang digunakan pada penelitian ini sebelumnya telah dilatih meggunakan data gambar dari data set ImageNet. Global Average Pooling 2D kemudian diterapkan setelah lapisan base model untuk mengurangi dimensi output, sehingga dapat digunakan untuk klasifikasi. Untuk mencegah overfitting, lapisan Dropout dengan dropout rate 0.5 diterapkan setelah Global Average Pooling, diikuti oleh Batch Normalization untuk normalisasi aktivasi. Lapisan-lapisan fully connected terdiri dari lapisan Dense dengan 1024 unit, diikuti oleh fungsi aktivasi ReLU untuk mengekstraksi fitur yang lebih abstrak. Dropout kembali diterapkan dengan rate 0.5 untuk mencegah overfitting, diikuti oleh lapisan Dense lebih kecil dengan 128 unit dan aktivasi ReLU. Lapisan output terakhir menggunakan Dense layer dengan 4 unit dan fungsi aktivasi softmax untuk menghasilkan probabilitas kelas pada tugas klasifikasi multi-kelas dengan 4 kelas yang ditentukan.

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import BatchNormalization, Dropout, Flatten, Dense, Activation, GlobalAveragePooling2D

model = Sequential()
model.add(base_model)
model.add(GlobalAveragePooling2D())
model.add(Dropout(0.5))
model.add(BatchNormalization())
model.add(Dense(1024))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(128))
model.add(Activation('relu'))
model.add(Dense(4, activation='softmax'))

model.summary()
```

Gambar 5. Arsitektur Model

4.6 Training Model

Pelatihan model dilakukan pada data latih (train_data) dan data validasi (val_data). Proses pelatihan ini berlangsung selama 20 epoch dengan ukuran batch sebesar 32. Saat pelatihan berlangsung, informasi akan ditampilkan secara verbal dengan tingkat detail (verbose) yang diatur ke 1. Selain itu, terdapat penggunaan callbacks yang terdiri dari checkpoint untuk menyimpan model terbaik dan reduce_lr untuk mengurangi learning rate secara otomatis selama pelatihan. Proses ini untuk mengoptimalkan model agar dapat memberikan performa terbaik dalam melakukan prediksi berdasarkan data yang tersedia.

```
history = model.fit(
    train_data,
    validation_data=val_data,
    epochs =20,
    batch_size=32,
    verbose=1,
    callbacks=[checkpoint,reduce_lr])
```

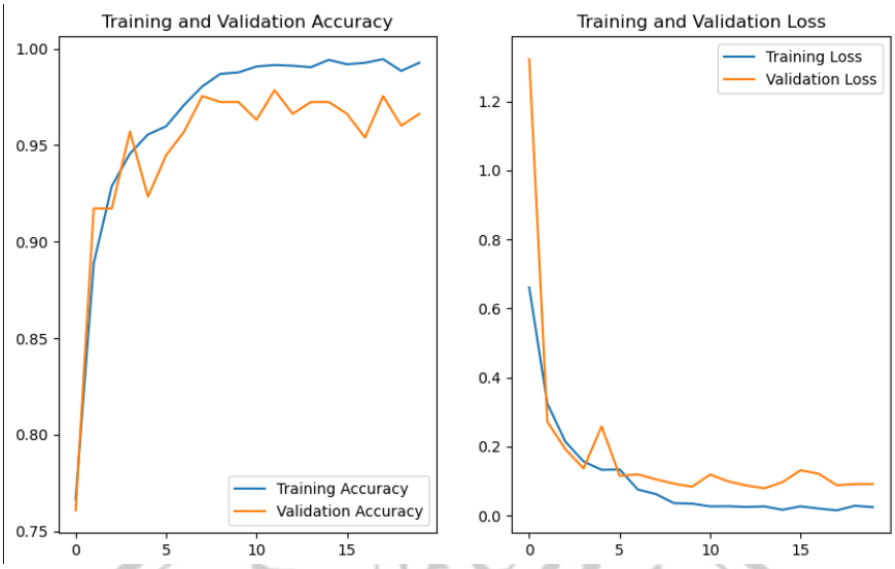
Gambar 6. Model

4.7 Evaluasi Model

Setelah proses pelatihan model selesai, langkah berikutnya adalah melakukan evaluasi menggunakan classification report, confusion matrix, grafik akurasi, dan loss model. Evaluasi ini bertujuan untuk mengukur kinerja model, membandingkannya dengan model lain, serta mengidentifikasi masalah seperti bias atau overfitting yang mungkin terjadi.

4.7.1 InceptionResNetV2 + CLAHE

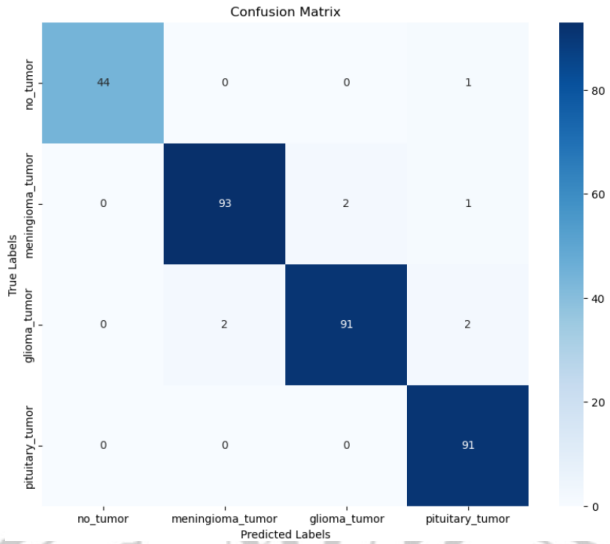
Pada skenario 1, pelatihan menggunakan model InceptionResNetV2 digabungkan dengan augmentasi serta penerapan teknik Contrast Limited Adaptive Histogram Equalization (CLAHE). Hasil evaluasi pada skenario ini menggunakan data uji dan mendapatkan hasil akurasi sebesar 97%. Hasil grafik akurasi dan loss dapat dilihat pada gambar 11 dibawah.



Gambar 7. Grafik Model Skenario 1

Pada grafik akurasi diatas, epoch pertama mencapai akurasi pelatihan sebesar 76.64% dan akurasi validasi sebesar 76.07%. Pada epoch ke-6 dan ke-10 terjadi pengurangan learning rate karena tidak ada peningkatan signifikan dalam akurasi validasi. Hal ini bertujuan untuk menghindari overfitting dan membantu model untuk belajar lebih baik pada data validasi. Akurasi pelatihan terus meningkat pada epoch berikutnya, mencapai 99.46% dan akurasi validasi sebesar 97.85% pada epoch ke-18.

Pada grafik loss, nilai loss pelatihan adalah sekitar 0.6610, sementara loss validasi awalnya sekitar 1.3225. Terjadi penurunan signifikan pada epoch kedua, di mana loss pelatihan turun menjadi sekitar 0.3246 dan loss validasi turun menjadi sekitar 0.2717. Selama proses pelatihan berlanjut, baik loss pelatihan maupun loss validasi mengalami penurunan secara bertahap. Pada epoch ke-20, model mencapai loss pelatihan sekitar 0.0245 dan loss validasi sekitar 0.0912, menunjukkan bahwa model telah belajar untuk mengoptimalkan prediksinya terhadap data pelatihan dan mampu melakukan generalisasi dengan baik terhadap data validasi.

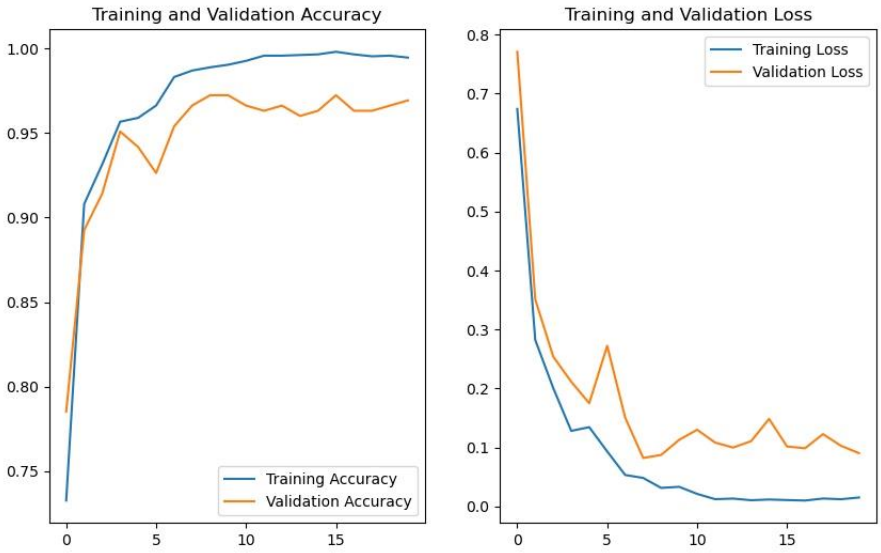


Gambar 8. Confussion Matrix

Confusion matrix tersebut memvisualisasikan performa model klasifikasi yang digunakan untuk mengidentifikasi empat kategori berbeda: "no_tumor", "meningioma_tumor", "glioma_tumor", dan "pituitary_tumor". Model berhasil mengidentifikasi "no_tumor" dengan benar sebanyak 44 Data. Untuk kategori "meningioma_tumor", model mengidentifikasi dengan akurat 93 data. Untuk "glioma_tumor", model mengenali 91 data dengan tepat. Pada kategori "pituitary_tumor", model berhasil mengklasifikasi 91 data dengan benar tanpa kesalahan. Model menunjukkan performa yang sangat baik dengan mayoritas prediksi yang benar terdapat di diagonal utama matriks.

4.7.2 InceptionResNetV2

Pada skenario 2, pelatihan menggunakan model InceptionResNetV2 digabungkan dengan augmentasi. Hasil evaluasi pada skenario ini menggunakan data uji dan mendapatkan hasil akurasi sebesar 96%. Hasil grafik akurasi dan loss dapat dilihat pada gambar 12 dibawah.



Gambar 9. Grafik Model Skenario 2

Pada grafik akurasi di atas, epoch pertama mencapai akurasi pelatihan sebesar sekitar 74.5% dan akurasi validasi sebesar sekitar 76.5%. Pada epoch ke-6 dan ke-10 terjadi pengurangan learning rate karena tidak ada peningkatan signifikan dalam akurasi validasi. Hal ini bertujuan untuk menghindari overfitting dan membantu model untuk belajar lebih baik pada data validasi. Akurasi pelatihan terus meningkat pada epoch berikutnya, mencapai sekitar 99.8% pada epoch ke-18, dan akurasi validasi mencapai sekitar 97.5%.

Pada grafik loss, nilai loss pelatihan adalah sekitar 0.75 pada epoch pertama, sementara loss validasi awalnya sekitar 0.76. Terjadi penurunan signifikan pada epoch kedua, di mana loss pelatihan turun menjadi sekitar 0.20 dan loss validasi turun menjadi sekitar 0.25. Selama proses pelatihan berlanjut, baik loss pelatihan maupun loss validasi mengalami penurunan secara bertahap. Pada epoch ke-20, model mencapai loss pelatihan sekitar 0.05 dan loss validasi sekitar 0.10.

4.8 Perbandingan Performa Model

Pada penelitian ini digunakan dataset yang sama seperti pada penelitian-penelitian sebelumnya. Adapun hasil dari model penelitian ini dapat dilihat pada gambar 14 dan 15 serta perbandingan performa model pada Tabel 3.

	precision	recall	f1-score	support
0	1.00	0.98	0.99	54
1	0.94	0.97	0.95	92
2	0.98	0.93	0.95	94
3	0.97	1.00	0.98	87
accuracy			0.97	327
macro avg	0.97	0.97	0.97	327
weighted avg	0.97	0.97	0.97	327

Gambar 10. Classification Report Skenario 1

	precision	recall	f1-score	support
0	1.00	0.98	0.99	45
1	0.98	0.97	0.97	96
2	0.98	0.96	0.97	95
3	0.96	1.00	0.98	91
accuracy			0.98	327
macro avg	0.98	0.98	0.98	327
weighted avg	0.98	0.98	0.98	327

Gambar 11. Classification Report Skenario 2

Tabel 3. Perbandingan Performa Model

Model	**Akurasi**
CNN	96%
InceptionResNetV2 + CLAHE	98%
InceptionResNet-V2	97%

Pada tabel perbandingan performa model diatas, model InceptionResNetV2 yang menerapkan teknik Contrast Limited Adaptive Histogram Equalization (CLAHE) mendapatkan akurasi tertinggi daripada penelitian sebelumnya dan dari model yang tidak menerapkan metode CLAHE.