BAB IV

HASIL DAN PEMBAHASAN

4.1. Load Dataset

Dataset yang telah di unduh dari situs kaggle terbagi menjadi dua folder, yaitu Training dan Testing denga total jumlah 3264 data. Implementasi untuk memuat dataset ini menggunakan kode Python dengan bantuan library OpenCV (cv2) dan NumPy. Prosesnya dimulai dengan menentukan label-label yang ada dalam dataset, seperti 'no_tumor', 'meningioma_tumor', 'glioma_tumor', dan 'pituitary_tumor'. Implementasi load dataset dapat dilihat pada gambar berikut.

```python
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
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))
        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))
        X_train.append(img)
        y_train.append(i)

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

Gambar 2. Implementasi load dataset

4.2. Pre-Processing

Pada tahap preprocessing dataset, langkah yang dilakukan yaitu pengurangan ukuran piksel gambar dari resolusi 512x512 menjadi 150x150. Pengurangan resolusi ini merupakan langkah penting yang bertujuan untuk meningkatkan efisiensi komputasi selama proses pelatihan data. Dengan mengurangi resolusi gambar, model dapat memproses data lebih cepat karena jumlah informasi yang harus diproses berkurang secara signifikan. Meskipun resolusi gambar dikurangi, informasi penting dari gambar tetap dipertahankan agar model tetap dapat belajar dan membuat prediksi yang akurat.

```
image_size = 150
```

Gambar 3. Resizing image

4.3. Data Splitting

Pada penelitian ini, dilakukan pembagian dataset menjadi tiga bagian: data latih, data validasi, dan data uji. Dataset yang digunakan dalam penelitian ini terdiri dari total 3264 data, yang dibagi menjadi 80% untuk data latih, 10% untuk data validasi, dan 10% untuk data uji.

```
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=101)
```

Gambar 3. Data Splitting

Data validasi digunakan selama proses pelatihan untuk memvalidasi model. Model dievaluasi pada data validasi untuk memantau performa dan menyesuaikan parameter, sehingga dapat mencegah overfitting. Sebanyak 10% data test digunakan untuk mengukur performa akhir model. Evaluasi pada data uji memberikan gambaran yang lebih akurat tentang performa model.

Pembagian data ini penting untuk memastikan model dilatih, divalidasi, dan diuji dengan benar untuk menilai kinerja dan generalisasi model. Parameter "*random_state*" digunakan untuk memastikan bahwa pembagian data dilakukan secara acak tetapi tetap dapat direproduksi.

4.4. Augmentasi Data

Augmentasi data adalah teknik untuk meningkatkan jumlah data dengan memodifikasi gambar asli sehingga memiliki penampilan yang bervariasi. Teknik ini berperan penting dalam meningkatkan kinerja klasifikasi dengan menyediakan variasi data yang lebih banyak. Penelitian ini menggunakan beberapa teknik augmentasi seperti rotasi, pergeseran lebar dan tinggi, zoom, serta horizontal flip.

```
datagen = ImageDataGenerator(
    rotation_range=20,
    width_shift_range=0.1,
    height_shift_range=0.1,
    zoom_range=0.2,
    horizontal_flip=True,
    validation_split=0.111,
)
```

Gambar 4. Augmentasi Data

Kelas ImageDataGenerator digunakan untuk mendefinisikan berbagai transformasi augmentasi data. Parameter rotation_range=20 menjadikan gambar diputar hingga 20 derajat, width_shift_range=0.1 dan height_shift_range=0.1 memungkinkan pergeseran gambar secara horizontal dan vertikal hingga 10% dari total lebar dan tinggi, zoom_range=0.2 melakukan zooming hingga 20%, dan horizontal_flip=True memungkinkan flipping gambar secara horizontal. Selain itu, parameter validation_split=0.111 digunakan untuk membagi data latih menjadi subset data latih dan data validasi, dengan sebagian data latih digunakan sebagai data validasi saat melatih model.

4.5. Implementasi Model EfficientNetV2B0

Implementasi model dilakukan menggunakan arsitektur *Squential*. Susunan arsitektur yang digunakan, yaitu: 1) GlobalAveragePooling2D, 2) Dropout, 3) Dense.

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dropout, Dense, Activation, GlobalAveragePooling2D
from tensorflow.keras.regularizers import l2

model = Sequential()
model.add(base_model)
model.add(GlobalAveragePooling2D())
model.add(Dropout(0.5))
model.add(Dense(256, kernel_regularizer=l2(0.01)))
model.add(Activation('relu'))
model.add(Dense(4, activation='softmax'))

model.summary()
```

Gambar 5. Model

Kerangka pada model ini menggunakan Sequential dari keras untuk merancang arsitektur yang efektif untuk tugas klasifikasi. Penggunaan base model (EfficientNetV2B0) merupakan model yang sudah dilatih sebelumnya menggunakan imagenet, yang bertujuan untuk mengekstraksi fitur tingkat tinggi dari dataset input dengan efisiensi yang tinggi. Selanjutnya GlobalAveragePooling2D ditambahkan untuk mengurangi dimensi spasial dari output fitur tanpa mengurangi informasi penting. Untuk mengurangi overfitting, diterapkan lapisan Dropout sebesar 0.5, yang secara acak menonaktifkan setengah dari unit-unit neuron dalam lapisan sebelumnya selama proses pelatihan.

Lapisan berikutnya adalah Dense layer dengan 256 neuron, yang menggunakan regularisasi L2 dengan faktor 0.01. Regularisasi ini membantu mengontrol kompleksitas model dan mencegah overfitting dengan membatasi norma L2 dari bobot. Setelah itu, model diakhiri dengan lapisan Dense yang memiliki 4 neuron dan menggunakan fungsi aktivasi softmax. Lapisan ini bertujuan untuk menghasilkan probabilitas output untuk setiap kelas yang ada, dengan model ini difokuskan untuk tugas klasifikasi multikelas dengan 4 kelas yang berbeda.

4.6. Pelatihan Model

```
with tf.device('/device:GPU:0'):
    history = model.fit(train_data,validation_data=val_data, epochs =20, verbose=1, batch_size=32,
                        callbacks=[checkpoint, lr_scheduler])
```

Gambar 5. Pelatihan Model

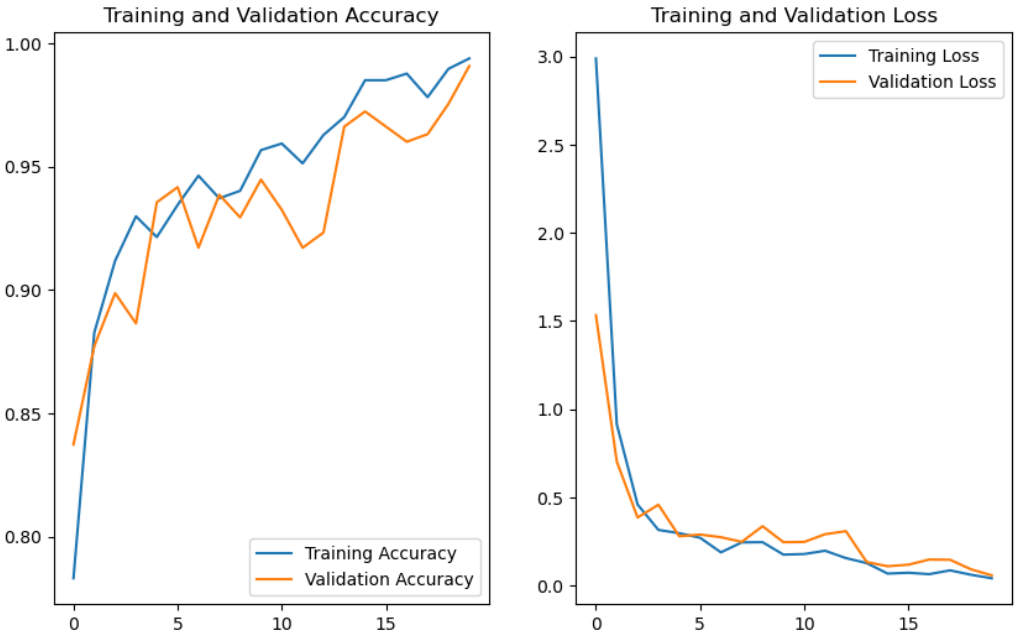
Model dilatih menggunakan data pelatihan dan divalidasi dengan data validasi selama 20 epoch. Proses pelatihan dilakukan menggunakan perangkat GPU untuk mempercepat komputasi, dengan batch size sebesar 32. Optimizer yang digunakan adalah "Adam", dan loss function yang digunakan adalah "Categorical Crossentropy". Callbacks seperti checkpoint dan lr_scheduler digunakan untuk menyimpan model terbaik selama pelatihan dan mengatur learning rate secara dinamis.

4.7. Evaluasi Model

Pada tahap ini, model yang telah dilatih menggunakan arsitektur EfficientNetV2B0 dievaluasi secara menyeluruh untuk mengukur kinerja objek. Evaluasi ini melibatkan serangkaian metrik dan pengujian pada dataset yang tidak pernah dilihat oleh model selama proses pelatihan. Hasil evaluasi dan pengujian ini diharapkan memberikan wawasan mendalam tentang keandalan dan kegunaan model EfficientNetV2B0 dalam konteks pengenalan objek. Analisis ini dapat membantu mengidentifikasi potensi perbaikan atau peningkatan model serta memberikan panduan untuk pengembangan model di masa depan.

4.7.1 Grafik Akurasi dan Loss Model

Pengujian model dilakukan dengan menggunakan grafik yang menunjukkan akurasi dan loss. Grafik ini dihasilkan dari data yang dikumpulkan selama proses pelatihan model. Sepanjang pelatihan, nilai akurasi dan loss dicatat untuk setiap iterasi atau epoch, sehingga memungkinkan untuk melihat bagaimana performa model berubah seiring waktu. Grafik akurasi memberikan informasi tentang seberapa baik model dapat memprediksi dengan benar, sedangkan grafik loss menunjukkan seberapa besar kesalahan yang dibuat oleh model. Hasil grafik akurasi dan loss model dapat dilihat pada gambar dibawah.



Gambar 7. Grafik Akurasi dan Loss

Berdasarkan grafik pada Training and Validation Accuracy, garis biru menggambarkan performa model pada data pelatihan, sedangkan garis oranye menunjukkan akurasi pada data validasi. Pada epoch pertama, akurasi pelatihan tercatat sebesar 78,32%, sementara akurasi validasi mencapai 83,74%. Pada epoch kedua, akurasi pelatihan meningkat menjadi 88,28% dan akurasi validasi naik menjadi 87,73%. Seiring berjalannya pelatihan, model terus menunjukkan peningkatan performa. Pada epoch ke-20, akurasi pelatihan mencapai nilai tertinggi yaitu 99,39%, sedangkan akurasi validasi mencapai 99,08%.

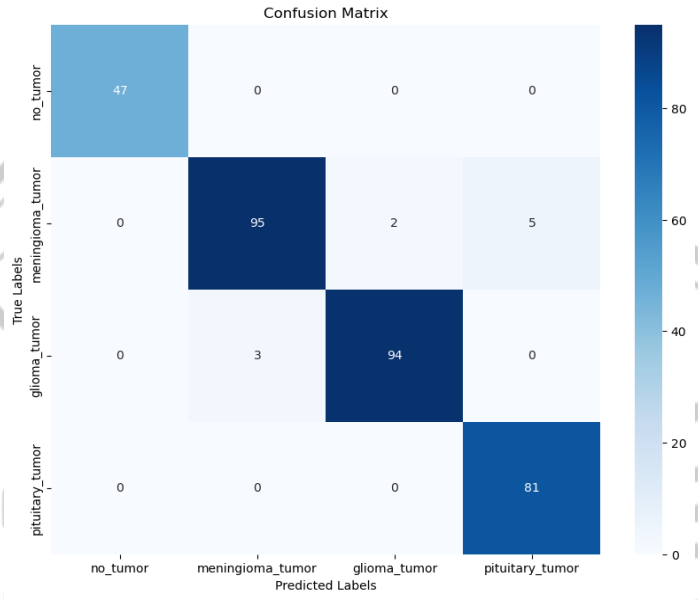
Grafik Training and Validation Accuracy diatas menunjukkan bahwa model tidak mengalami overfitting, karena peningkatan akurasi pada data pelatihan diimbangi dengan peningkatan akurasi pada data validasi. Meskipun akurasi pelatihan mencapai nilai yang sangat tinggi, tidak ada penurunan signifikan pada akurasi validasi, yang sering menjadi indikator overfitting.

Pada gambar grafik loss diatas, garis biru merepresentasikan loss pelatihan dan garis oranye menunjukkan loss validasi. Pada epoch pertama, loss pelatihan tercatat cukup tinggi yaitu sebesar 2,9892, namun hal ini berhasil ditangani pada epoch kedua dengan penurunan loss pelatihan menjadi 0,9157. Seiring berjalannya pelatihan, overfitting berhasil cukup

baik dikendalikan oleh model. Hingga akhir pelatihan, loss terbaik yang diperoleh adalah 0,0418 pada epoch ke-20. Dengan demikian, grafik menunjukkan bahwa model tidak hanya mempertahankan akurasi yang tinggi tetapi juga efisiensi yang baik dalam hal loss.

4.7.2 Confussion Matrix

Hasil evaluasi model menggunakan confussion matrix dapat dilihat pada gambar dibawah.



Gambar 9. Confussion Matrix

Confusion matrix di atas memberikan gambaran tentang performa model dalam mengklasifikasikan jenis penyakit pada tumor otak. Terdapat empat kelas yang berbeda, yaitu 'no_tumor', 'meningioma_tumor', 'glioma_tumor', dan 'pituitary_tumor'.

Model berhasil mengidentifikasi 47 sampel dari kelas 'no_tumor' dengan benar tanpa kesalahan, lalu dari 102 sampel kelas meningioma_tumor model berhasil mengklasifikasikan 95 sampel dengan benar, kemudian dari 97 sampel kelas glioma_tumor model mengklasifikasikan 94 sampel dengan benar, dan yang terakhir dari 81 sampel kelas pituitary_tumor berhasil diklasifikasikan dengan benar Secara keseluruhan, model menunjukkan performa yang sangat baik dalam mengklasifikasikan keempat jenis kelas tersebut, dengan tingkat akurasi yang tinggi pada setiap kelas dan jumlah kesalahan klasifikasi yang sangat rendah.

4.7.3 Perbandingan Penelitian Sebelumnya

Pada penelitian ini digunakan dataset yang sama seperti pada penelitian-penelitian sebelumnya. Peneitian ini menerapkan teknik augmentasi serta model EfficientNetV2B0 untuk klasifikasi citra tumor otak MRI. Classification Report dan hasil perbandingan penelitian ini dengan penelitian sebelumnya dapat dilihat pada pada tabel 3 dan tabel 4.

Tabel 3. Classification Report

	Precision	Recall	F1-Score
0	1.00	1.00	1.00
1	0.97	0.93	0.95
2	0.98	0.97	0.97
3	0.94	1.00	1.00
Accuracy	97%		

Tabel 4. Perbandingan Penelitian Sebelumnya

Model	Akurasi
Convolutional Neural Network	96%
EfficientNetV2B0	97%