# BAB 3
# METODOLOGI PENELITIAN

## 3.1 Metodologi Penelitian

Pada bagian ini akan merangkum langkah langkah yang dilakukan dalam melakukan pembuatan dan penyusunan penelitian. Langkah langkah tersebut adalah telaah literatur, pengumpulan data, perancangan kode dan model, pengujian dan evaluasi model, dokumentasi penelitian.

### 3.1.1 Studi Literatur

Pada tahap ini akan mengumpulkan materi atau literatur dari berbagai sumber seperti jurnal, artikel, buku, *conference*, dan sumber lainnya yang berkaitan dengan topik penelitian ini dengan tujuan menjadi pendukung dan sumber informasi dalam mendukung penelitian ini.

### 3.1.2 Pengumpulan Data

Pada tahap ini akan melakukan pencarian *dateset* yang sesuai topik penelitian dan berbagai referensi yang dibutuhkan dalam penelitian ini. Data dalam penelitian ini diambil dari *website* Kaggle yang berasal dari 3 *dataset* yaitu SARTAJ dataset, BR35H, Jun Cheng. Data yang dipakai memiliki 4 *class* yaitu *glioma, meningioma, pituitary, no tumor*.

Tabel 3.1. Data per kelas

| Nama Penyakit | Banyak Data |
|---|---|
| *Glioma* | 1621 |
| *Meningioma* | 1645 |
| *Pituitary* | 2000 |
| *NoTumor* | 1757 |

### 3.1.3 Perancangan Kode dan Model

Pada tahap ini perancangan kode dan model dimulai dengan melakukan *train test split* terhadap data agar terbagi menjadi dua bagian yaitu *training*

*data* dan *testing data*, dengan rasio 80:20, yaitu 80% *training* dan 20% *testing* .Menurut penelitian yang dilakukan oleh David Bajuzy, pembagian data dengan rasio 80:20 adalah yang paling optimal karena memberikan jumlah data saat proses *train* yang cukup dan data untuk *test* juga cukup. Kemudian proses *training* atau pembangunan model dimulai, langkah pertama adalah gambar *MRI* yang berisi jenis dari tumor otak dimasukkan ke dalam model setelah melalui tahap *preprocessing*, yang mencakup pengubahan ukuran gambar dan normalisasi nilai pixel. Pada tahap awal, model menerapkan layer *fused-MBConv* dengan *kernel* 3x3 untuk mengekstraksi fitur dasar dari gambar. Selanjutnya, pada setiap *stage*, jumlah *channel* meningkat secara bertahap, dan blok *Squeeze-and-Excitation (SE)* diterapkan untuk memperkuat fokus pada fitur-fitur yang dianggap penting bagi klasifikasi. Model kemudian melanjutkan proses ini melalui beberapa layer *MBConv*, yang memanfaatkan *depthwise separable convolution* untuk mengurangi jumlah parameter dan kompleksitas model, sembari tetap mempertahankan kemampuan dalam ekstraksi fitur. Setelah serangkaian konvolusi, layer *Global Average Pooling* digunakan untuk menggabungkan seluruh informasi spasial menjadi representasi vektor yang komprehensif. Akhirnya, layer *fully connected* menghasilkan output akhir yang menunjukkan kelas tumor, *yakni glioma, meningioma,* atau *pituitary.*

### 3.1.4 Pengujian dan Evaluasi Model

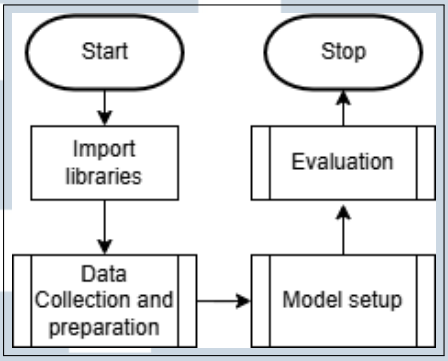
Pengujian akan dilakukan dengan memanfaatkan confussion matrix untuk pendapatkan hasil perhitungan dari akurasi, presisi, *recall*, *f1-Score*. Apabila hasil dari pengujian model dirasa kurang baik, dilakukan evaluasi terhadap model sehingga mendapatkan hasil yang lebih baik.

### 3.1.5 Dokumentasi Penelitian

Pada tahap ini dilakukan penulisan dokumentasi dari hasil penelitian yang sudah dilakukan dan evaluasi sistem yang sudah berhasil dibuat.

## 3.2 Perancangan Sistem

Pada penelitian ini dilakukan beberapa proses dalam melakukan perancangan sistem yang dapat dilihat pada Gambar 3.1.




Gambar 3.1. *Flowchart* perancangan sistem

### 3.2.1 Import Libraries

Pada tahap ini akan dilakukan *import* berbagai *library* yang digunakan dalam melakukan perancangan kode seperti *numpy, pandas, tensorflow, sklearn, matplotlib,* dan *seaborn*

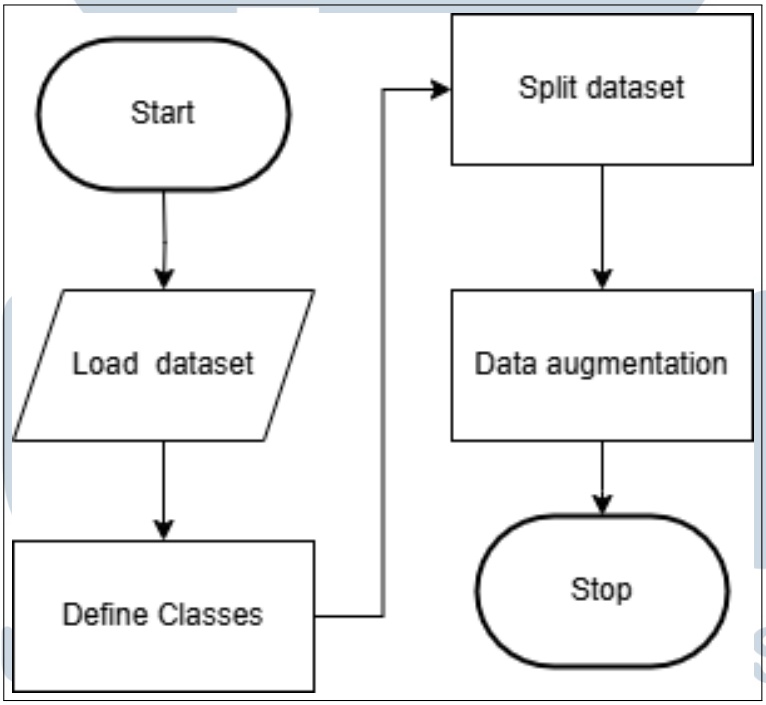
### 3.2.2 Data Collection and Preparation

Tahap pertama dalam penelitian ini adalah mempersiapkan dataset MRI (*Magnetic Resonance Imaging*) yang akan digunakan sebagai data pelatihan dan pengujian model. *Dataset* ini diambil dari sumber yang sudah tersedia, dan terdiri dari gambar MRI otak yang dikategorikan ke dalam empat kelas, yaitu *glioma*, *meningioma*, *notumor*, dan *pituitary*. Setiap kategori mewakili jenis tumor tertentu atau kondisi tanpa tumor (notumor), sehingga memungkinkan model untuk mempelajari perbedaan visual di antara berbagai tipe tumor otak. Setelah dataset dimuat, data tersebut dipecah menjadi dua bagian, yaitu 80% untuk pelatihan dan 20% untuk validasi. Pembagian ini bertujuan agar model dapat dilatih menggunakan sebagian besar data, sementara sisanya digunakan untuk mengevaluasi performa model pada data yang tidak terlihat selama pelatihan, sehingga membantu mencegah *overfitting*.

Tahap selanjutnya adalah menerapkan data *augmentation* pada dataset

pelatihan. Teknik ini menambahkan variasi pada data tanpa mengumpulkan data baru, sehingga meningkatkan kemampuan model untuk mengenali pola dalam gambar yang bervariasi. Beberapa teknik *augmentation* yang diterapkan meliputi *rescaling*, *shearing*, dan *zooming*. *Rescaling* dilakukan dengan mengubah nilai piksel gambar agar berada dalam rentang 0 hingga 1 (dari skala 255), yang membantu model mempercepat proses *konvergensi*. Selain itu, diterapkan *shearing* atau distorsi pada gambar dengan cara memiringkan bentuk objek di dalamnya, untuk melatih model dalam mengenali objek meski dalam bentuk yang berbeda.

Terakhir, teknik *zooming* memperbesar atau memperkecil gambar untuk memperkenalkan variasi dalam ukuran objek, sehingga model dapat menjadi lebih adaptif terhadap perubahan skala dalam gambar. Tahap persiapan ini memastikan bahwa data dalam kondisi optimal dan dapat membantu model untuk lebih generalizable pada data baru. *Flowchart* pada tahap ini bisa dilihat pada Gambar 3.2 berikut.




Gambar 3.2. *Flowchart* data *collection*

### 3.2.3 Model Setup

Pada tahap Model *Setup*, penelitian ini menggunakan arsitektur model EfficientNetV2B1 yang telah dilatih sebelumnya (*pre-trained*) dengan menggunakan *weights* dari dataset ImageNet. Tahap awal dimulai dengan memuat model ini beserta bobot yang telah dilatih menggunakan ImageNet, sebuah dataset umum yang terdiri dari lebih dari satu juta gambar yang mencakup lebih dari seribu kategori. ImageNet sering digunakan dalam pelatihan model *deep learning* karena mencakup berbagai kelas kategori visual, memberikan model pemahaman dasar tentang karakteristik umum, seperti pola, warna, bentuk, dan tepi objek.

Setelah memuat model, dilakukan proses *freezing* pada lapisan awal dan *fine-tuning* pada lapisan akhir. *Freezing* merupakan teknik ”mengunci” atau menghentikan pembaruan bobot pada lapisan awal model, sehingga fitur-fitur dasar yang telah dipelajari dari dataset ImageNet, seperti pola bentuk dan tepi, tetap terjaga dan relevan untuk mengenali elemen dasar dalam gambar MRI. *Fine-tuning*, di sisi lain, diterapkan pada lapisan-lapisan akhir yang lebih dalam, di mana bobot pada lapisan ini disesuaikan dengan dataset MRI yang baru. Lapisan-lapisan ini dilatih lebih lanjut untuk mempelajari pola spesifik dalam gambar MRI, memungkinkan model mengenali fitur tumor otak secara akurat. Pada penelitian ini *fine-tuning* yang digunakan adalah melakukan penyesuaian dalam nilai *learning rate*. *Learning rate* yang lebih rendah digunakan pada lapisan yang dilatih ulang. Hal ini memungkinkan model untuk melakukan penyesuaian pada bobot tanpa perubahan drastis, sehingga model tetap stabil dan meminimalkan risiko overfitting. Selain itu juga menggunakan regularisasi l2, diterapkan pada lapisan-lapisan yang dilatih ulang untuk menghindari bobot yang terlalu besar pada neuron. Ini membantu menjaga stabilitas model selama pelatihan dan mencegah model terlalu cocok dengan data pelatihan, atau overfitting.

Untuk meningkatkan performa model, arsitektur ini ditambahkan dengan beberapa lapisan tambahan, sebagai berikut:

**A Global Average Pooling Layer**

Lapisan ini mereduksi dimensi fitur yang dihasilkan oleh lapisan akhir model dengan menghitung nilai rata-rata dari setiap fitur. Hal ini membantu mengekstraksi informasi penting secara efisien dan mengurangi risiko *overfitting*.

### B Dropout Layer

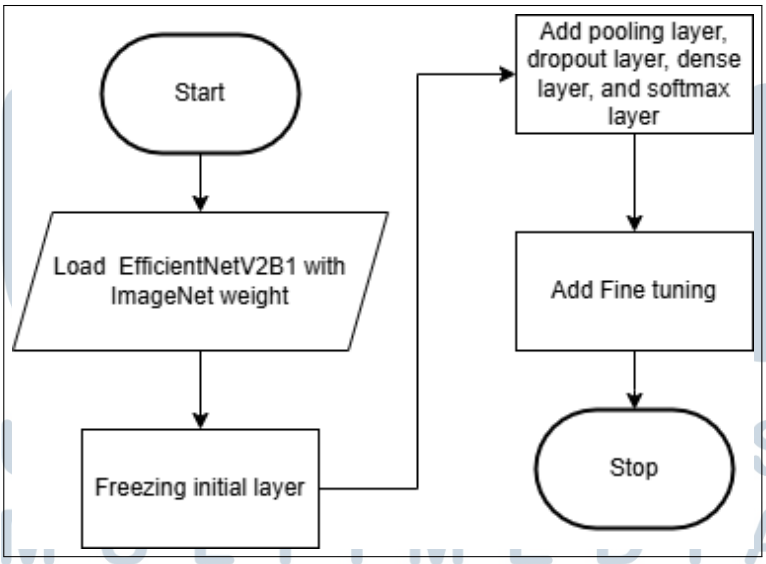
Berfungsi mencegah *overfitting* dengan cara mengabaikan secara acak sejumlah neuron selama pelatihan, membuat model tidak bergantung sepenuhnya pada neuron tertentu sehingga dapat menghasilkan model yang lebih *generalizable*.

### C Dense Layer

Lapisan ini menghubungkan seluruh neuron secara penuh dan berfungsi menangkap pola kompleks dari data yang dipelajari oleh model. Fungsi aktivasi *ReLU* pada lapisan ini meningkatkan kemampuan model dalam mengenali pola non-linear pada data MRI.

### D Softmax Layer

Pada lapisan akhir, softmax digunakan untuk mengubah keluaran model menjadi probabilitas yang menunjukkan kemungkinan gambar MRI termasuk dalam salah satu dari empat kelas tumor (*glioma*, *meningioma, notumor, dan pituitary*). *Flowchart* pada tahap ini bisa dilihat pada Gambar 3.3 berikut.




Gambar 3.3. *Flowchart* model *setup*

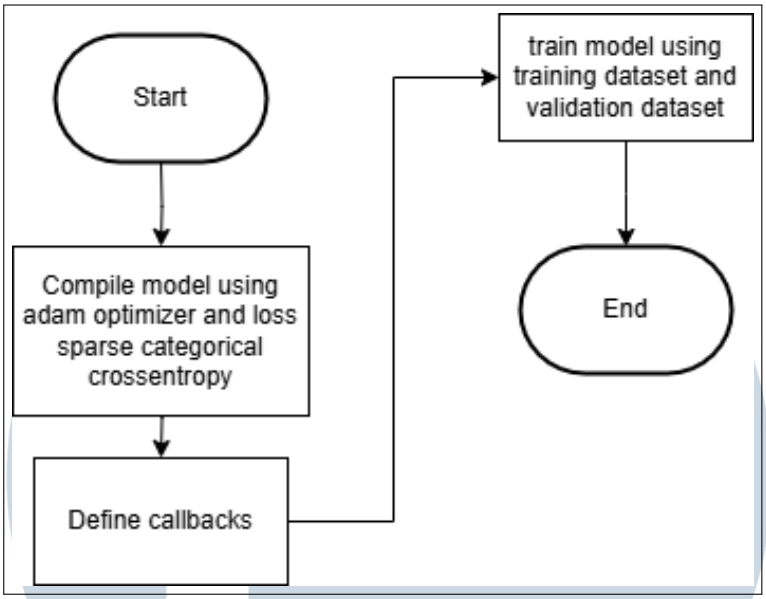
### 3.2.4 Training Process

Pada tahap Training Process, model yang telah dibangun akan dilatih menggunakan metode *fit* untuk mempelajari pola-pola yang terdapat dalam dataset. Dalam proses ini, model memanfaatkan *optimizer Adam*, yang merupakan salah satu algoritma optimisasi yang paling banyak digunakan dalam deep learning. Adam menggabungkan keunggulan dari dua metode optimisasi sebelumnya, yaitu *AdaGrad* dan *RMSProp*, dengan menghitung *learning rate* yang adaptif untuk setiap parameter. Pendekatan ini memungkinkan model untuk mencapai konvergensi yang lebih cepat dan stabil, sehingga menjadi pilihan yang tepat untuk menangani dataset yang besar dan kompleks.

Fungsi *loss* yang digunakan selama pelatihan adalah *sparse categorical crossentropy*, yang dirancang khusus untuk masalah klasifikasi multi-kelas. Fungsi ini mengukur selisih antara distribusi probabilitas keluaran model dan label kelas yang sesungguhnya, dengan memberikan penalti yang lebih besar pada prediksi yang salah. Pemilihan fungsi *loss* ini mendukung model dalam menilai performanya secara efektif dalam mengidentifikasi kelas yang benar dari gambar MRI.

Model akan menjalani beberapa siklus pelatihan yang disebut *epoch*. *Epoch* merupakan satu siklus penuh di mana model memproses seluruh dataset pelatihan. Dalam setiap *epoch*, model melakukan iterasi melalui semua *batch* data, memperbarui bobotnya berdasarkan perhitungan *loss* dan algoritma optimisasi yang digunakan. Jumlah *epoch* yang ditentukan dalam proses pelatihan mencerminkan seberapa sering model akan belajar dari data. Penting untuk memilih jumlah *epoch* yang sesuai, karena terlalu sedikit *epoch* dapat mengakibatkan *underfitting*, sementara terlalu banyak *epoch* dapat menyebabkan *overfitting*, di mana model belajar terlalu mendalam dari noise dalam data pelatihan.

Selama pelatihan, beberapa *callback* juga didefinisikan, antara lain *ModelCheckpoint* untuk menyimpan bobot model terbaik berdasarkan performa pada data validasi, *EarlyStopping* untuk menghentikan proses pelatihan ketika tidak terdapat perbaikan pada *loss* validasi setelah sejumlah *epoch* tertentu, dan *ReduceLROnPlateau* untuk mengurangi *learning rate* ketika *loss* validasi mengalami stagnasi. *Callback* ini berperan penting dalam meningkatkan efisiensi pelatihan dan mencegah *overfitting*, sehingga model dapat belajar dengan optimal dari data yang tersedia.*Flowchart* pada tahap ini bisa dilihat pada Gambar 3.4 berikut.



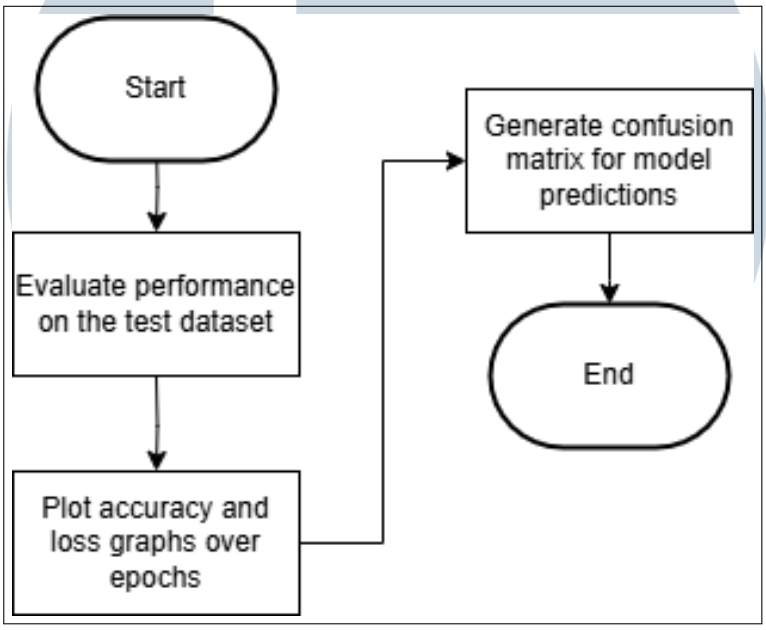

Gambar 3.4. *Flowchart training process*

### 3.2.5 Evaluation

Pada tahap *Evaluation*, model dievaluasi menggunakan metode *evaluate* pada dataset pengujian, yang bertujuan untuk mendapatkan metrik kinerja seperti akurasi dan loss. Hasil evaluasi ini disimpan dalam format dictionary untuk analisis lebih lanjut. Selanjutnya, model melakukan prediksi kelas pada dataset pengujian dengan menggunakan metode *predict*. Proses ini menghasilkan probabilitas untuk setiap kelas yang ada, yang kemudian diubah menjadi label kelas dengan menggunakan fungsi *argmax*.

Untuk mengukur kinerja model secara lebih mendetail, label kebenaran (*true labels*) dari dataset pengujian dikumpulkan. Ini dilakukan dengan mengiterasi melalui dataset dan mengekstrak label dari setiap batch. Dengan label kebenaran dan prediksi kelas yang dihasilkan, *confusion metrics* dihitung. *Confusion metrics* ini memberikan gambaran yang jelas tentang jumlah prediksi yang benar dan salah untuk setiap kelas, sehingga memungkinkan penilaian kinerja model dalam mengklasifikasikan data.

Akhirnya, *confusion metrics* divisualisasikan menggunakan *heatmap*, yang membantu dalam menganalisis performa model secara visual. Visualisasi ini menunjukkan di mana model melakukan kesalahan klasifikasi, sehingga memberikan wawasan yang berharga untuk perbaikan lebih lanjut dalam model yang telah dikembangkan.*Flowchart* pada tahap ini bisa dilihat pada Gambar 3.5 berikut.




Gambar 3.5. *Flowchart evaluation*