BAB 2
LANDASAN TEORI

Terdapat beberapa literatur yang dapat menunjang penelitian ini adalah sebagai berikut.

2.1 Tumor Otak

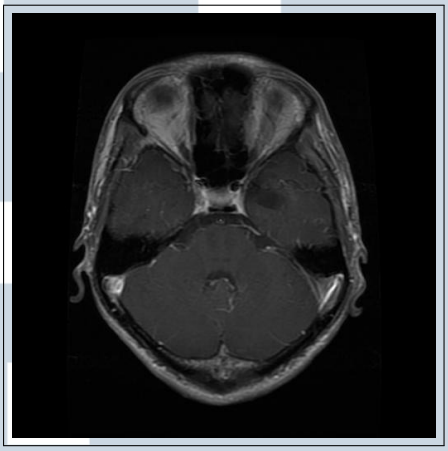
Tumor otak merupakan suatu kondisi di mana terjadi pertumbuhan abnormal sel-sel di dalam otak. Tumor ini dapat dibagi menjadi dua kategori utama, yaitu tumor otak primer yang berasal langsung dari sel-sel otak, dan tumor otak sekunder yang disebabkan oleh penyebaran (metastasis) kanker dari bagian tubuh lain ke otak. Tumor otak juga diklasifikasikan menjadi tumor jinak dan ganas, tergantung pada tingkat keganasannya serta potensinya untuk tumbuh dan menyebar ke jaringan otak lainnya[16].

Secara umum, tumor otak dapat menyebabkan berbagai gangguan fungsi otak, tergantung pada lokasi dan ukuran tumor tersebut. Beberapa gejala yang sering muncul termasuk sakit kepala, kejang, gangguan penglihatan, masalah keseimbangan, serta gangguan kognitif dan motorik. Perkembangan dan penyebaran tumor otak sangat bervariasi, dengan beberapa jenis tumbuh dengan lambat (jinak) sementara yang lain bisa berkembang dengan cepat dan agresif (ganas)[16].

Penyebab tumor otak dapat dipengaruhi oleh beberapa faktor. Paparan radiasi, terutama radiasi pengion yang digunakan dalam perawatan kanker, dianggap sebagai salah satu faktor risiko signifikan karena radiasi ini dapat merusak DNA sel otak dan menyebabkan perkembangan tumor.[17]. Selain itu, faktor genetik juga berperan, di mana mutasi pada gen tertentu atau sindrom bawaan seperti *neurofibromatosis* dan sindrom *Li-Fraumeni* dapat meningkatkan risiko seseorang terkena tumor otak[18]. Penelitian lain juga menunjukkan bahwa cedera kepala berulang dapat meningkatkan risiko tumor otak, khususnya jenis *glioma* dan *meningioma*, meskipun hubungan ini masih memerlukan penelitian lebih lanjut untuk memastikan keterkaitannya[19].

Tumor otak memiliki beberapa jenis, diantaranya adalah :

A Glioma

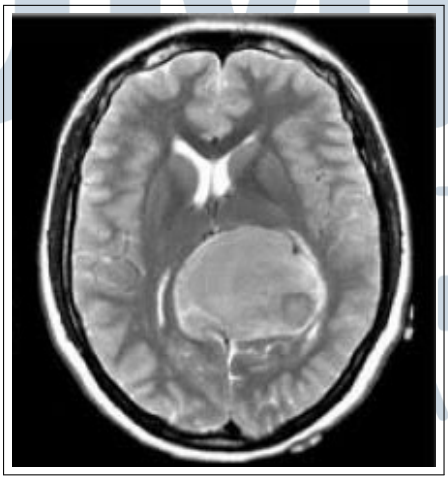


Gambar 2.1. MRI *Glioma*

Sumber: [20]

Glioma adalah tumor yang berasal dari sel *glia* (berguna untuk melindungi sel otak dan sumsum tulung belakang). Sel *glia* yang mengalami mutasi akan tumbuk secara tidak terkontrol dan lama kelamaan akan membentuk tumor [21].

B Meningioma

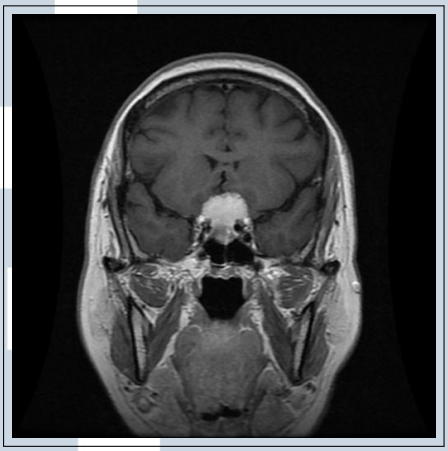


Gambar 2.2. MRI *Meningioma*

Sumber: [22]

Meningioma merupakan tumor primer sistem saraf pusat yang paling umum. *Meningioma* biasanya merupakan *neoplasma* jinak yang tumbuh lambat dan diduga berasal dari sel *meningotel* [23].

C Pituitary



Gambar 2.3. MRI *Pituitary*

Sumber: [24]

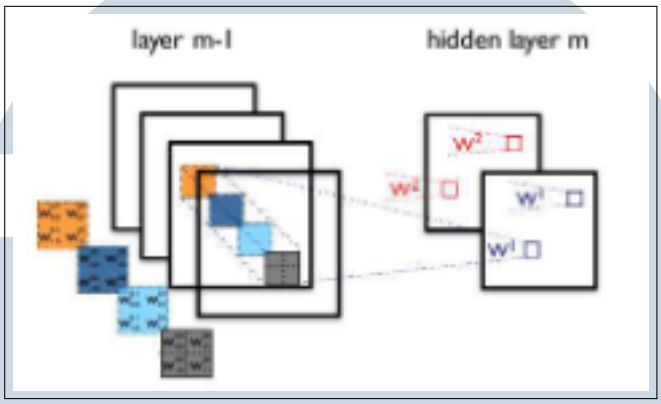
Tumor *pituitary* adalah pertumbuhan abnormal yang terjadi pada kelenjar *pituitary*, sebuah kelenjar kecil yang terletak di dasar otak. Kelenjar ini berperan penting dalam mengatur berbagai fungsi tubuh melalui produksi hormon.

2.2 Convolutional Neural Network

Convolutional Neural Network (CNN) merupakan evolusi dari *Multilayer Perceptron* (MLP) yang dirancang khusus untuk mengolah data dua dimensi. CNN termasuk dalam kategori *Deep Neural Network* karena memiliki kedalaman jaringan yang signifikan dan sering digunakan untuk menghandle data citra. Saat berhadapan dengan tugas klasifikasi citra, MLP kurang cocok karena tidak mampu menyimpan informasi spasial dari citra, dan menganggap setiap piksel sebagai fitur yang independen, sehingga menghasilkan performa yang kurang optimal.

Convolutional Neural Network (CNN) memproses data dua dimensi, yang menyebabkan perbedaan dalam operasi linear dan parameter bobot dibandingkan dengan jaringan lain. Dalam CNN, operasi linear dilakukan melalui operasi konvolusi, di mana bobotnya tidak lagi bersifat satu dimensi, melainkan memiliki

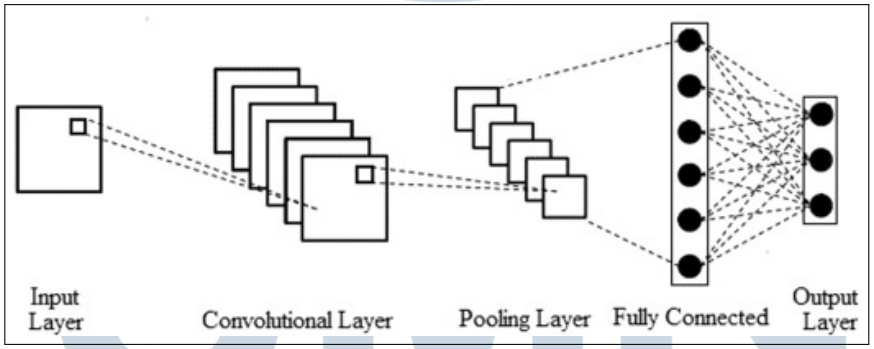
bentuk empat dimensi yang merupakan kumpulan kernel konvolusi. Oleh karena itu CNN hanya bisa digunakan pada input yang dimensi nya tidak hanya satu seperti gambar dan suara.



Gambar 2.4. Proses Konvolusi pada CNN

Sumber: [25]

Arsitektur CNN terbentuk dari beberapa layer, diantaranya adalah *convolutional layer*, *pooling layer*, dan *fully connected layer*.



Gambar 2.5. CNN *layer*

Sumber: [26]

Selain layer layer tersebut ada juga dua parameter yang menunjang ketiga layer tersebut, yaitu *dropout layer* dan *activation function* yang akan dijelaskan dibawah ini :

2.2.1 Convolutional Layer

Convolution layer pada *Convolutional Neural Network* (CNN) beroperasi dengan menghubungkan filter atau kernel ke data input. Filter ini memiliki dimensi

kecil dan terdapat bobot yang dapat di-*adjust* selama proses pelatihan model. Proses konvolusi dimulai dengan pemasangan filter di sudut kiri atas data input. Setelah itu, filter digeser ke arah kanan dan ke bawah dengan ukuran yang konsisten. Pada setiap posisi, filter melakukan operasi perkalian antara bobotnya dan nilai-nilai dalam data input. Hasil perkalian tersebut diakumulasikan dan disimpan sebagai *output* dari *convolution layer*. Proses konvolusi ini berulang hingga seluruh wilayah dari data input telah diolah. *Output* dari lapisan konvolusi ini selanjutnya diteruskan ke *layer* berikutnya dalam arsitektur CNN. Dengan demikian, setiap filter berperan dalam mengekstraksi fitur-fitur dari data input dan memainkan peran penting dalam proses pembelajaran jaringan untuk tugas tertentu[27].

2.2.2 Pooling layer

Dalam kebanyakan kasus, *convolutional layer* diikuti oleh *pooling layer*. Tujuan utama dari layer ini adalah untuk memperkecil ukuran peta fitur yang dikonvolusi untuk mengurangi berat komputasi. Hal ini dilakukan dengan mengurangi koneksi antar *layer* dan beroperasi secara independen pada setiap peta fitur. Terdapat beberapa jenis *pooling* yang sering digunakan. Pada *max pooling*, elemen terbesar diambil dari peta fitur. *Average pooling* menghitung rata-rata elemen dalam bagian gambar berukuran yang telah ditentukan. Jumlah total elemen di bagian yang telah ditentukan dihitung dalam *sum pooling*. *Pooling layer* biasanya berfungsi sebagai jembatan antara *convolution layer* dan *fully connected layer* [27].

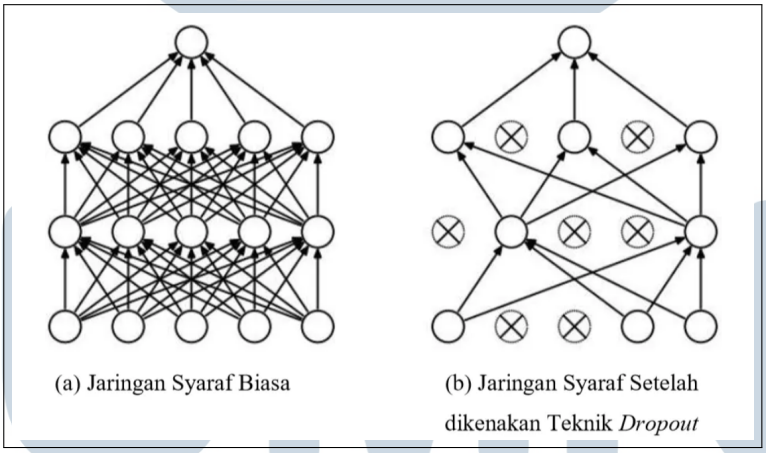
2.2.3 Fully Connected Layer

Fully-connected layer merupakan suatu lapisan dalam arsitektur jaringan saraf yang memiliki semua neuron aktivitas dari lapisan sebelumnya terhubung dengan semua neuron di lapisan berikutnya, mirip dengan jaringan saraf tiruan pada umumnya. Sebelum dihubungkan ke semua neuron di *fully-connected layer*, aktivitas dari lapisan sebelumnya perlu diubah menjadi data satu dimensi. *fully-connected layer* biasanya digunakan dalam metode *multi-layer perceptron* dengan tujuan utama untuk mengolah data sehingga dapat diklasifikasikan. Perbedaan mendasar antara *fully-connected layer* dan *convolutional layer* terletak pada koneksi neuron. Neuron pada *convolutional layer* terhubung hanya ke daerah tertentu pada input, sedangkan *fully-connected layer* memiliki neuron yang terhubung secara keseluruhan. Meskipun demikian, keduanya masih menggunakan

operasi produk dot, sehingga fungsinya tidak memiliki perbedaan yang signifikan. [28]

2.2.4 Dropout

Dropout merupakan suatu teknik regularisasi pada *neural nertwork* yang melibatkan pemilihan acak beberapa neuron untuk tidak aktif selama proses pelatihan. Neuron-neuron ini dapat dianggap sebagai sementara ”dibuang”. Artinya, kontribusi dari neuron yang dinonaktifkan dihentikan sementara, dan bobot baru tidak diterapkan pada neuron tersebut selama proses *backpropagation*. Penggunaan *dropout* bertujuan untuk mencegah *overfitting* dan mempercepat proses pembelajaran. *Dropout* melibatkan penghapusan neuron, baik yang terletak pada *hidden layer* maupun *visible layer* dalam jaringan. Dengan melakukan penghapusan ini, suatu neuron dihilangkan sementara dari jaringan. Proses pemilihan neuron untuk dihilangkan dilakukan secara acak, dan setiap neuron diberikan probabilitas dengan nilai antara 0 dan 1.



Gambar 2.6. *Dropout layer*

Sumber: [29]

2.2.5 Activation Function

Fungsi aktivasi memainkan peran kunci pada tahap sebelum dan sesudah melakukan *pooling layer*, yang berlangsung setelah proses konvolusi. Pada tahap ini, hasil konvolusi dikenai fungsi aktivasi atau *activation function*. Terdapat beberapa pilihan fungsi aktivasi yang umum digunakan dalam *convolutional network*, seperti tanh() atau reLU. Beberapa peneliti cenderung memilih aktivasi

reLU karena kinerjanya yang dianggap lebih efektif. Dalam konteks penggunaan reLU sebagai fungsi aktivasi, nilai *output* dari neuron akan menjadi 0 jika inputnya bersifat negatif. Namun, jika nilai input dari fungsi aktivasi adalah positif, maka *output* dari neuron akan menjadi nilai input aktivasi tersebut [28].

2.3 EfficientNetV2B1

EfficientNetV2B1 adalah model yang menggunakan arsitektur dari *Convolutional Neural Network* (CNN). *Convolutional Neural Network* (CNN) merupakan evolusi dari *Multilayer Perceptron* (MLP) yang dirancang khusus untuk mengolah data dua dimensi[30]. CNN termasuk dalam kategori *Deep Neural Network* karena memiliki kedalaman jaringan yang signifikan dan sering digunakan untuk menghandle data citra. Saat berhadapan dengan tugas klasifikasi citra, MLP kurang cocok karena tidak mampu menyimpan informasi spasial dari citra, dan menganggap setiap piksel sebagai fitur yang independen, sehingga menghasilkan performa yang kurang optimal.

Convolutional Neural Network (CNN) memproses data dua dimensi, yang menyebabkan perbedaan dalam operasi linear dan parameter bobot dibandingkan dengan jaringan lain. Dalam CNN, operasi linear dilakukan melalui operasi konvolusi, di mana bobotnya tidak lagi bersifat satu dimensi, melainkan memiliki bentuk empat dimensi yang merupakan kumpulan kernel konvolusi. Oleh karena itu CNN hanya bisa digunakan pada input yang dimensi nya tidak hanya satu seperti gambar dan suara.

EfficientNetV2B1 dirancang menggunakan pendekatan *neural architecture search* (NAS) yang mengoptimalkan efisiensi parameter dan kecepatan pelatihan. Model ini menggabungkan beberapa inovasi arsitektural yang menjadikannya efisien dan cepat, seperti penggunaan *MBConv* dan *Fused-MBConv* pada lapisan awal, dengan ukuran kernel kecil (3x3) dan rasio ekspansi yang lebih kecil. Ini memungkinkan pengurangan penggunaan memori dan *overhead* akses data [13].

Arsitektur *EfficientNetV2B1* dapat dilihat pada Tabel 2.1.

Pada *Stage 1*, arsitektur menggunakan *fused-MBConv* dengan *kernel* 3x3, tanpa blok *Squeeze-and-Excitation* (SE). *Output* layer pada tahap ini memiliki dimensi 112x112 dengan 16 *channel*. *Stride* 2 digunakan untuk melakukan *downsampling*, yang bertujuan mengurangi ukuran spasial *input* menjadi lebih kecil. Pada *Stage 2* hingga *Stage 4*, *fused-MBConv* tetap digunakan, namun jumlah *channel* bertambah secara bertahap. Pada stage-stage ini, *channel*-nya

Tabel 2.1. EfficientNetV2B1 *Architecture*

Stage	Operator	Repeats	Output Size	Expansion Ratio	Channels	SE	Stride
1	Fused-MBConv, 3x3	1	112x112	1	16	No	2
2	Fused-MBConv, 3x3	2	112x112	4	32	No	1
3	Fused-MBConv, 3x3	3	56x56	4	48	No	2
4	Fused-MBConv, 3x3	4	28x28	4	96	No	2
5	MBConv, 3x3	3	14x14	4	112	Yes	1
6	MBConv, 3x3	3	14x14	6	192	Yes	2
7	MBConv, 3x3	1	7x7	6	320	Yes	1
8	Conv2D, 1x1	1	7x7	-	1280	No	-
9	Global Average Pooling	1	1x1	-	1280	No	-
10	Fully Connected	1	-	-	(Menyesuaikan)	No	-

masing-masing sebesar 32, 48, dan 96. Selain itu, *expansion ratio* diterapkan untuk memperbesar dimensi *channel* sebelum melakukan *convolution pointwise* (*convolution* dengan *kernel* 1x1). Pada *Stage 4*, *stride* 2 kembali digunakan untuk *downsampling*, sehingga ukuran spasial menjadi 28x28. Pada *Stage 5* hingga *Stage 7*, arsitektur beralih dari *fused-MBConv* ke *MBConv* standar, yang merupakan *depthwise separable convolution*. Pada tahap ini, blok SE mulai diterapkan, memberikan perhitungan tambahan pada *channel* untuk memperkuat informasi fitur yang dianggap penting. Pada *Stage 7*, layer *MBConv* terakhir menghasilkan *output* dengan 320 *channel* dan ukuran spasial 7x7. Pada *Stage 8*, digunakan layer *convolution pointwise* dengan *kernel* 1x1. Layer ini menghasilkan *output* dengan ukuran 7x7x1280, sehingga meningkatkan jumlah *channel* tanpa mengubah ukuran spasial. Selanjutnya, pada *Stage 9*, layer *Global Average Pooling* digunakan untuk mereduksi semua informasi spasial menjadi satu piksel per *channel*. *Output* dari stage ini memiliki dimensi 1x1x1280, yang menyatukan seluruh informasi spasial menjadi satu vektor per *channel*. Lalu pada tahap terakhir layer *fully connected* digunakan untuk menghasilkan *output* akhir sesuai dengan kelas yang ada di *dataset*. *Output* ini disesuaikan dengan tugas klasifikasi yang umum digunakan pada *dataset* seperti *ImageNet*.

A Scaling EfficientNetV2B1

Model ini menerapkan compound scaling yang menyelaraskan tiga dimensi utama dari jaringan saraf: resolusi gambar(r), kedalaman model(d), dan lebar model(w). Formula skala gabungan diuraikan sebagai berikut:

$$d = \alpha^k \cdot d_0, \quad w = \beta^k \cdot w_0, \quad r = \gamma^k \cdot r_0 \tag{2.1}$$

Dimana α, β, γ adalah faktor skala untuk kedalaman, lebar, dan resolusi,

dan k merupakan koefisien yang ditentukan berdasarkan kompleksitas model awal. Dengan demikian, model dapat diskalakan lebih efisien tanpa meningkatkan jumlah parameter yang berlebihan [13].

B MBConv dan Fused-MBConv

MBConv *(Mobile Inverted Bottleneck Convolution)* adalah blok dasar yang digunakan di EfficientNet. MBConv mengombinasikan lapisan *convolution* dengan blok residual untuk meningkatkan efisiensi. Dalam rumus, jika X adalah input, dan W adalah bobot, maka MBConv dapat ditulis sebagai:

$$Output = ReLU6(W\,.\,X + b\,) \tag{2.2}$$

Dimana *b* adalah bias, dan ReLU6 adalah fungsi aktivasi yang digunakan untuk menjaga kestabilan numerik pada perangkat dengan sumber daya terbatas. Pada *EfficientNetV2B1*, *Fused-MBConv* digunakan di lapisan awal. Ini menggabungkan operasi *convolution* dengan lapisan *batch normalization* dan fungsi aktivasi *ReLU6* tanpa melibatkan *separable convolution*, yang meningkatkan kecepatan dan efisiensi memori [31].

C Progressive Learning

Untuk meningkatkan kecepatan pelatihan dan akurasi, model ini menerapkan metode progressive learning, di mana ukuran gambar dan teknik augmentasi gambar ditingkatkan secara bertahap selama pelatihan, cara ini memungkinkan model untuk mulai dengan ukuran gambar kecil, yang meningkatkan efisiensi pada tahap awal pelatihan, dan secara bertahap memperbesar ukuran gambar saat pelatihan berlanjut[31].

2.4 Evaluasi Model

Dalam mengukur peforma model klasifikasi, dapat menggunakan *confussion metrics* [32]. Dari *metrics* tersebut dapat dihitung *metrics* lain seperti akurasi, presisi, *recall*, dan *f1-score*. Berikut penjelasan dari *metrics-metrics* tersebut.

A Confussion Metrics

Confussion metrics adalah sebuah *metrics* yang digunakan untuk mencatat seberapa bagus sebuah model dari sebarapa banyak benar dan salah dari prediksi model tersebut [32]. Dalan *confussion metrics* ada 4 bagian utama yaitu *true positive, true negative, false positive, false negative* [33].

True positive adalah dimana tebakkan model adalah true dan ternyata memang *true*. *True negative* adalah dimana tebakkan model adalah *false* dan ternyata memang *false*. *False positive* adalah dimana tebakkan model adalah *true* dan ternyata *false*. *False negative* adalah dimana tebakkan model adalah *false* dan ternyata *true* [33].

Dengan menggunakan *true positive, true negative, false positive* dan *false negative*, *metrics-metrics* lain seperti akurasi, *recall*, presisi, dan *f1-score* bisa dihitung. Berikut adalah formula dari metrics berikut.

Akurasi adalah metrik evaluasi yang mengukur seberapa baik model membuat prediksi yang benar dari total prediksi yang dilakukan.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{2.3}$$

Precision adalah metrik evaluasi yang mengukur seberapa baik model membuat prediksi yang benar untuk kelas positif dari total prediksi positif yang dilakukan.

$$Precision = \frac{TP}{TP+FP} \tag{2.4}$$

Recall adalah metrik evaluasi yang menggambarkan seberapa baik suatu model dalam mengidentifikasi kelas positif dengan benar.

$$Recall = \frac{TP}{TP+FN} \tag{2.5}$$

Nilai *f1-score* akan memberikan informasi tentang seberapa baik sebuah model dalam menggabungkan kemampuan *precision* dan *recall*, sehingga kita bisa memahami seberapa efektif model kita dalam melakukan klasifikasi.

$$F1\ score\ = 2\frac{RecallxPrecision}{Recall+Precision} \tag{2.6}$$