BAB 2
LANDASAN TEORI

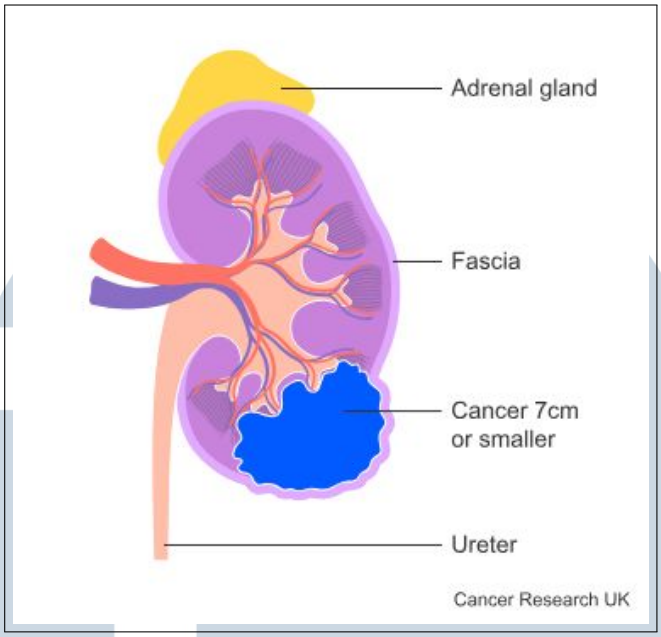
2.1 Aspek Klinis dan Patologis Kanker Ginjal *Clear Cell Renal Cell Carcinoma* (ccRCC)

Clear Cell Renal Cell Carcinoma (ccRCC) adalah subtipe kanker ginjal paling umum pada orang dewasa, yang dinamai berdasarkan penampilan sel tumornya yang "jernih" atau pucat saat diamati di bawah mikroskop [1,17]. Kanker ini berasal dari sel-sel yang melapisi tubulus kecil di ginjal, yang berfungsi menyaring limbah dari darah [17]. Diagnosis awal sering kali bersifat insidental, ditemukan melalui pemeriksaan pencitraan seperti *Computed Tomography* (CT) atau *Magnetic Resonance Imaging* (MRI) yang dilakukan untuk kondisi lain [1]. Setelah tumor terdeteksi, diagnosis definitif umumnya dikonfirmasi melalui biopsi, di mana seorang ahli patologi memeriksa sampel jaringan di bawah mikroskop untuk menentukan subtipe dan *grade* tumor [1].

Sistem *grading* tumor memiliki peran penting dalam menentukan prognosis dan perencanaan terapi [18]. Sistem *grading* yang paling banyak digunakan saat ini adalah klasifikasi *World Health Organization / International Society of Urological Pathology* (WHO/ISUP), yang telah menggantikan sistem Fuhrman sebelumnya [19]. Sistem WHO/ISUP ini tidak hanya mengandalkan ukuran nukleus, tetapi secara khusus menekankan pada *prominensi nukleolus* serta keberadaan *morfologi rhabdoid* atau sarkomatoid, yang dikategorikan sebagai *grade* 4 [19]. Ketergantungan pada fitur-fitur mikroskopis ini sangat relevan karena mencerminkan agresivitas dan heterogenitas tumor pada tingkat seluler [20]. Oleh karena itu, model diagnostik berbasis *artificial intelligence* (AI) yang efektif perlu mampu mengidentifikasi fitur kuantitatif dari pencitraan medis yang berkorelasi dengan karakteristik mikroskopis tersebut. Kemampuan ini berpotensi menjadikan model AI sebagai alat prediksi *grade* histologis secara non-invasif, yang merupakan kemajuan signifikan dalam diagnosis kanker ginjal.

Secara umum, tahapan kanker ginjal berdasarkan sistem *Tumor–Node–Metastasis* (TNM) diklasifikasikan sebagai berikut:

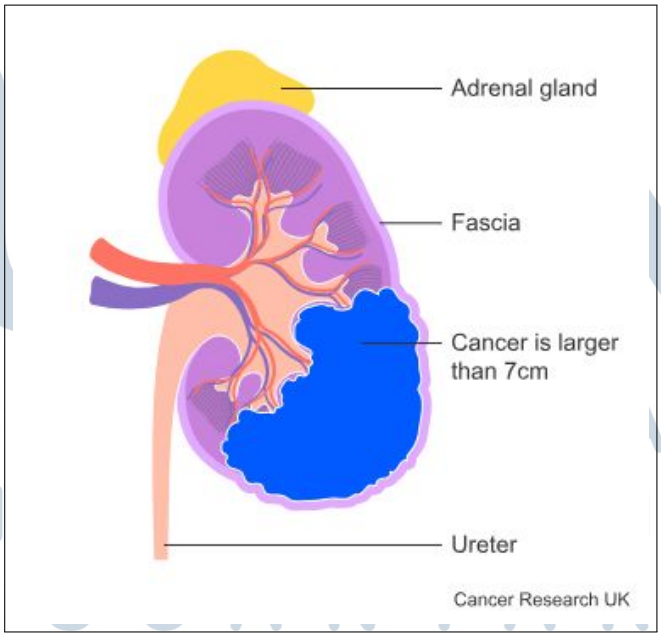
1. Stadium I: Tumor berukuran kecil, yaitu 7 cm atau kurang, dan sepenuhnya terbatas di dalam ginjal. Tumor belum menyebar ke kelenjar getah bening maupun organ lain (*T1, N0, M0*) [21,22].



Gambar 2.1. Ilustrasi Kanker Ginjal Stadium I

Sumber: Cancer Research UK [22]

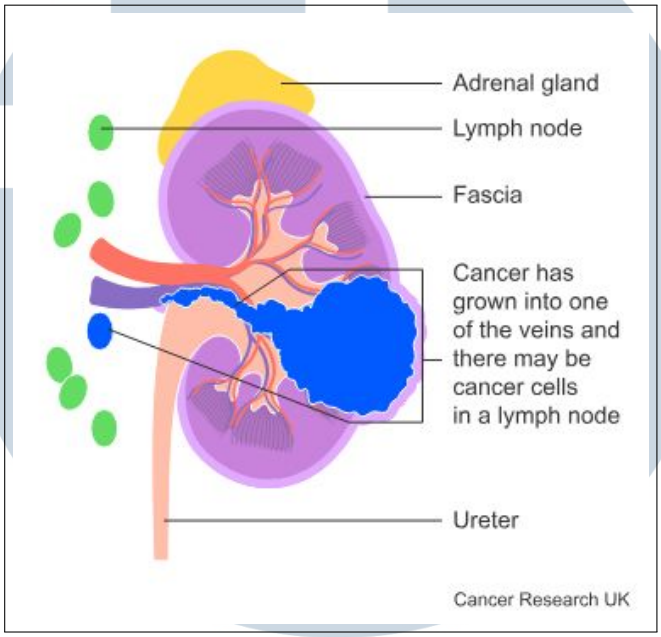
2. Stadium II: Tumor berukuran lebih besar dari 7 cm, namun masih sepenuhnya berada di dalam ginjal dan belum menyebar (*T2, N0, M0*) [21,22].



Gambar 2.2. Ilustrasi Kanker Ginjal Stadium II

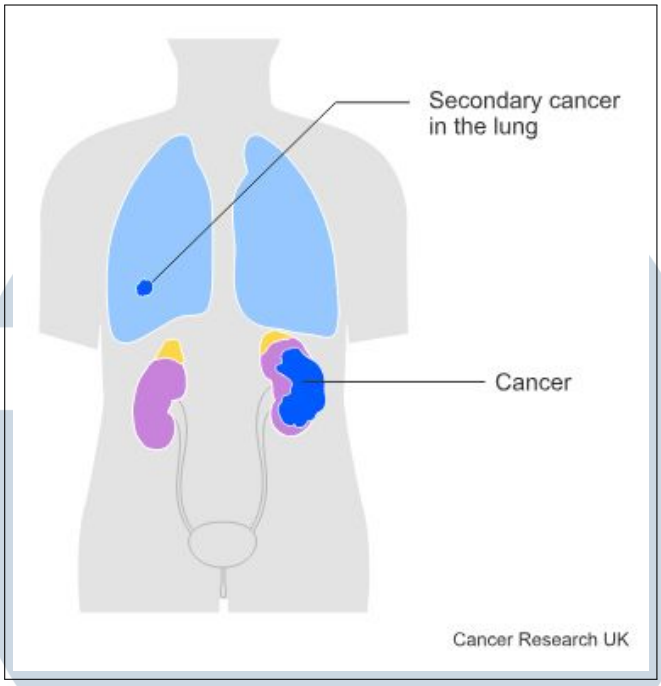
Sumber: Cancer Research UK [22]

3. Stadium III: Tumor dengan ukuran berapa pun yang telah menyebar ke pembuluh darah utama ginjal (seperti vena ginjal atau vena kava) dan/atau ke setidaknya satu kelenjar getah bening regional, namun belum menyebar ke organ jauh (*T3* atau *N1, M0*) [21,22].



Gambar 2.3. Ilustrasi Kanker Ginjal Stadium III
Sumber: Cancer Research UK [22]

4. Stadium IV: Stadium paling lanjut, ditandai dengan penyebaran tumor ke luar fasia Gerota (*Gerota's fascia*), keterlibatan kelenjar adrenal, atau metastasis jauh ke organ lain seperti paru-paru atau tulang (*T4* atau *M1*) [21,22].



Gambar 2.4. Ilustrasi Kanker Ginjal Stadium IV
Sumber: Cancer Research UK [22]

2.2 Peran Kecerdasan Buatan (*Artificial Intelligence*) dalam Pencitraan Medis dan Radiomik

Penerapan *artificial intelligence* (AI) telah merevolusi analisis pencitraan medis. Algoritma *machine learning* dan *deep learning* mampu menganalisis citra medis seperti MRI, CT scan, maupun *X-ray* secara otomatis untuk mendeteksi tumor, lesi, atau kelainan jaringan [23]. Kemampuan ini memungkinkan deteksi dini dan perencanaan terapi yang lebih cepat, serta berpotensi mengurangi kesalahan manusia dalam proses diagnosis.

Dalam konteks ini, *radiomics* muncul sebagai bidang penelitian yang berfokus pada ekstraksi sejumlah besar fitur kuantitatif dari citra medis konvensional [24]. Radiomik mengubah citra diagnostik menjadi data terukur yang dapat dianalisis lebih lanjut menggunakan metode statistik atau pendekatan AI [25]. Fitur-fitur kuantitatif tersebut, yang dikenal sebagai fitur radiomik, berpotensi mengungkap pola dan karakteristik tumor yang tidak dapat diidentifikasi melalui observasi visual semata [24]. Dengan demikian, radiomik memungkinkan konversi citra medis menjadi *biomarker* non-invasif yang mampu merepresentasikan biologi dan perilaku tumor secara lebih mendalam [26].

2.3 Keunggulan Pendekatan Multimodal

Dalam pengembangan model kecerdasan buatan (AI) untuk diagnosis medis, pendekatan multimodal telah terbukti memiliki keunggulan yang signifikan dibandingkan pendekatan unimodal [27]. Perbedaan mendasar antara kedua pendekatan ini terletak pada cakupan dan keragaman data yang digunakan dalam proses pemodelan. Model unimodal hanya memanfaatkan satu jenis data atau satu modalitas pencitraan, seperti CT atau MRI saja. Sebaliknya, model multimodal mengintegrasikan beberapa sumber data secara bersamaan, sehingga mampu menangkap karakteristik informasi yang lebih beragam dan saling melengkapi [25].

Integrasi berbagai modalitas memungkinkan model multimodal memperoleh pemahaman kontekstual yang lebih kaya dan komprehensif. Informasi yang tidak tertangkap oleh satu modalitas dapat dikompensasi oleh modalitas lain, sehingga menghasilkan representasi fitur yang lebih informatif. Hal ini berdampak langsung pada peningkatan performa prediksi, terutama pada tugas klasifikasi dan prognosis klinis yang memerlukan pemahaman kompleks terhadap kondisi patologis pasien [27].

Secara teknis, pengembangan model multimodal memiliki tingkat kompleksitas yang lebih tinggi karena memerlukan arsitektur khusus untuk memproses dan menggabungkan data dari berbagai sumber secara simultan, seperti melalui teknik *feature fusion* atau *decision fusion*. Meskipun demikian, peningkatan kompleksitas ini sebanding dengan peningkatan kinerja model. Studi sebelumnya melaporkan bahwa model radiomik multimodal yang menggabungkan fitur dari CT dan MRI mampu mencapai nilai *Area Under the Curve* (AUC) hingga 0,925, yang lebih tinggi dibandingkan model yang hanya menggunakan satu modalitas [27].

Keunggulan pendekatan multimodal menjadi sangat relevan dalam penelitian yang menggunakan dataset heterogen, seperti *The Cancer Imaging Archive* (TCIA). Data pada TCIA berasal dari pencitraan standar pelayanan klinis (*standard of care imaging*) di berbagai institusi, sehingga mencerminkan variasi yang tinggi dalam jenis pemindai, produsen perangkat, serta protokol akuisisi citra. Variasi ini dapat menjadi tantangan serius bagi model unimodal karena berpotensi menimbulkan bias dan menurunkan performa generalisasi. Dengan mengombinasikan informasi dari beberapa modalitas, pendekatan multimodal mampu mengurangi dampak variasi tersebut dan menghasilkan representasi data yang lebih *robust*, sehingga model yang dihasilkan memiliki kemampuan generalisasi yang lebih baik untuk penerapan di lingkungan klinis yang beragam

di masa depan [27].

Tabel 2.1 menyajikan ringkasan perbandingan antara model AI unimodal dan multimodal, serta menegaskan alasan pemilihan pendekatan multimodal sebagai metode yang optimal dalam penelitian ini.

Tabel 2.1. Perbandingan Aspek Model Unimodal dan Model Multimodal

Aspek	**Model Unimodal**	**Model Multimodal**
Lingkup Data	Mengolah satu jenis data atau satu modalitas pencitraan (misalnya CT atau MRI)	Mengintegrasikan beberapa jenis data atau modalitas pencitraan (misalnya CT dan MRI)
Pemahaman Konteks	Pemahaman konteks terbatas dan rentan terhadap ambiguitas informasi	Pemahaman kontekstual lebih kaya dan komprehensif
Performa Model	Performa baik pada tugas spesifik, namun menurun pada kasus kompleks	Memberikan analisis yang lebih akurat dan stabil pada berbagai kondisi
Kompleksitas Teknis	Relatif sederhana karena hanya memproses satu sumber data	Lebih kompleks karena membutuhkan mekanisme fusi data
Ketahanan terhadap Variasi Data	Rentan terhadap variasi perangkat dan protokol akuisisi	Lebih robust terhadap heterogenitas data

2.4 Ekstraksi Fitur dengan Radiomik (*Radiomics*)

Radiomics merupakan pendekatan komputasional yang bertujuan untuk mengubah citra medis menjadi representasi data numerik melalui proses ekstraksi fitur dari wilayah tumor yang telah disegmentasi, yang dikenal sebagai *Region of Interest* (ROI). Pendekatan ini memungkinkan penggalian informasi kuantitatif tingkat lanjut yang tidak dapat diamati secara visual oleh radiolog, sehingga hubungan antara pola pencitraan dan karakteristik biologis tumor dapat dianalisis secara objektif dan non-invasif.

Dalam penelitian ini, proses ekstraksi fitur radiomik dilakukan menggunakan pustaka *PyRadiomics*, yaitu sebuah framework berbasis Python yang dirancang untuk melakukan ekstraksi fitur secara terstandarisasi, transparan, dan dapat direproduksi [7]. PyRadiomics mengikuti definisi matematis yang direkomendasikan oleh *Image Biomarker Standardization Initiative* (IBSI), sehingga hasil fitur yang diperoleh memiliki konsistensi dan validitas yang tinggi

serta dapat dibandingkan dengan penelitian radiomik lainnya. Framework ini mendukung berbagai modalitas citra medis, termasuk CT, MRI, dan PET.

2.4.1 PyRadiomics sebagai Kerangka Kerja Ekstraksi Fitur (Framework)

PyRadiomics memerlukan dua input utama, yaitu citra medis dan *mask* ROI yang merepresentasikan area tumor hasil segmentasi. Sebelum proses ekstraksi fitur dilakukan, citra terlebih dahulu melalui tahapan pra-pemrosesan, seperti *resampling* voxel ke resolusi isotropik dan normalisasi intensitas. Tahapan ini bertujuan untuk mengurangi variasi spasial dan intensitas antar citra yang berasal dari perangkat dan protokol akuisisi yang berbeda, sehingga konsistensi perhitungan fitur dapat terjaga.

Ekstraksi fitur dilakukan tidak hanya pada citra asli (*original image*), tetapi juga pada citra hasil transformasi, seperti *wavelet*, *Laplacian of Gaussian* (LoG), dan *square root filters*. Transformasi citra ini digunakan untuk menyoroti karakteristik tekstur dan pola spasial pada berbagai skala, sehingga mampu menangkap heterogenitas internal tumor secara lebih komprehensif [7].

Seluruh parameter ekstraksi fitur, termasuk *bin width*, metode interpolasi, serta jenis filter pra-pemrosesan, didefinisikan melalui file konfigurasi berbasis YAML. Pendekatan ini memungkinkan proses ekstraksi dilakukan secara konsisten dan dapat direplikasi dengan mudah pada dataset lain, sehingga meningkatkan reliabilitas dan reprodusibilitas penelitian [7].

2.4.2 Fitur-Fitur Radiomik

PyRadiomics mengelompokkan fitur radiomik ke dalam beberapa kategori utama, yang masing-masing merepresentasikan aspek statistik, morfologis, dan tekstural dari jaringan tumor. Fitur-fitur ini berperan penting dalam mendeskripsikan heterogenitas tumor yang berkaitan dengan agresivitas, stadium, dan karakteristik biologis kanker [6,7].

1. First-Order Statistics: Fitur ini menggambarkan distribusi nilai intensitas voxel di dalam ROI tanpa mempertimbangkan hubungan spasial antar voxel. Fitur-fitur ini dihitung berdasarkan histogram intensitas dan mencakup ukuran statistik seperti *Mean*, *Variance*, *Skewness*, *Kurtosis*, *Entropy*, dan *Energy*. Parameter-parameter tersebut mencerminkan karakteristik distribusi intensitas, seperti tingkat variasi, homogenitas, serta kompleksitas jaringan tumor.

2. Shape Features: Fitur bentuk merepresentasikan karakteristik geometris dari ROI tumor, baik dalam dua dimensi (2D) maupun tiga dimensi (3D). Fitur 3D menggambarkan properti volumetrik seperti *Volume*, *Surface Area*, *Compactness*, dan *Sphericity*, sedangkan fitur 2D menangkap karakteristik bentuk pada irisan citra seperti *Perimeter*, *Area*, dan *Circularity*. Selain itu, fitur seperti *Elongation* dan *Axis Length* digunakan untuk menggambarkan bentuk dan orientasi tumor. Informasi ini berkaitan dengan ukuran dan morfologi tumor yang sering diasosiasikan dengan tingkat invasi dan stadium penyakit.

3. Gray Level Co-occurrence Matrix (GLCM): GLCM mengukur hubungan spasial antar pasangan voxel dengan nilai intensitas tertentu pada jarak dan arah tertentu. Fitur seperti *Contrast*, *Correlation*, *Homogeneity*, dan *Dissimilarity* digunakan untuk menilai keteraturan pola tekstur dan kompleksitas struktur jaringan tumor [6,7].

4. Gray Level Run Length Matrix (GLRLM): GLRLM mengkuantifikasi panjang deretan (*run*) voxel yang memiliki tingkat keabuan yang sama dalam arah tertentu. Setiap *run* merepresentasikan sekumpulan voxel berurutan dengan intensitas identik. Fitur yang diekstraksi, seperti *Short Run Emphasis* (SRE) dan *Long Run Emphasis* (LRE), digunakan untuk mengkarakterisasi tekstur citra, khususnya dalam mengidentifikasi pola halus (run pendek) dan kasar (run panjang) pada jaringan tumor [6,7].

5. Gray Level Size Zone Matrix (GLSZM): GLSZM mengkuantifikasi ukuran zona voxel yang saling terhubung dan memiliki tingkat keabuan yang sama, tanpa mempertimbangkan arah. Setiap zona merepresentasikan sekumpulan voxel dengan intensitas identik yang membentuk suatu area homogen. Fitur yang diekstraksi, seperti *Small Area Emphasis* (SAE) dan *Large Area Emphasis* (LAE), digunakan untuk mengkarakterisasi distribusi ukuran zona, yang berkaitan dengan tingkat heterogenitas tekstur pada area tumor [6,7].

6. Gray Level Dependence Matrix (GLDM): GLDM mengkuantifikasi tingkat ketergantungan voxel terhadap voxel pusat berdasarkan kesamaan intensitas dalam suatu lingkungan lokal. Sebuah voxel dianggap dependen jika perbedaan intensitasnya terhadap voxel pusat berada di bawah ambang batas tertentu. Fitur yang dihasilkan, seperti *Dependence Non-Uniformity* (DN) dan

Dependence Entropy (DE), digunakan untuk mengkarakterisasi kompleksitas tekstur serta variasi struktur internal pada tumor [6,7].

7. Neighboring Gray Tone Difference Matrix (NGTDM): NGTDM mengukur perbedaan antara intensitas suatu *voxel* dengan rata-rata intensitas *voxel* di sekitarnya (tetangga lokal). Fitur ini digunakan untuk menggambarkan karakteristik tekstur berdasarkan perubahan intensitas lokal. Beberapa fitur yang dihasilkan, seperti *Coarseness*, *Contrast*, *Busyness*, *Complexity*, dan *Strength*, memberikan informasi mengenai tingkat kekasaran, variasi, serta kompleksitas pola tekstur pada ROI [6,7].

Seluruh fitur radiomik dihitung berdasarkan formulasi matematis yang terdokumentasi dalam pustaka PyRadiomics [7]. Hasil ekstraksi berupa vektor fitur numerik yang merepresentasikan karakteristik statistik dan spasial tumor. Vektor fitur ini selanjutnya digunakan sebagai masukan pada tahap seleksi fitur dan pembangunan model klasifikasi untuk memprediksi stadium atau subtipe kanker ginjal secara otomatis.

2.4.3 Transformasi Citra dan Fitur Orde Tinggi (*High-Order Features*)

Selain fitur yang dihitung dari citra asli, PyRadiomics juga memungkinkan ekstraksi fitur dari citra hasil transformasi yang dikenal sebagai *high-order features*. Transformasi ini dilakukan sebelum proses ekstraksi fitur dengan tujuan untuk menyoroti variasi tekstur dan heterogenitas tumor yang mungkin tidak terlihat secara langsung pada citra asli. Beberapa transformasi citra yang digunakan dalam penelitian ini dijelaskan sebagai berikut.

1. *Square Filter*

   Transformasi ini menghitung kuadrat dari setiap intensitas voxel pada citra:

$$I_{\text{square}}(x,y,z) = I(x,y,z)^2 \tag{2.1}$$

   di mana $I(x,y,z)$ menyatakan intensitas voxel asli pada koordinat spasial (x,y,z). Operasi ini memperkuat nilai intensitas yang lebih tinggi sehingga dapat meningkatkan kontras pada area dengan intensitas terang dalam ROI tumor.

2. *Square Root Filter*

   Transformasi akar kuadrat dinyatakan sebagai:

$$I_{\text{sqrt}}(x,y,z) = \sqrt{I(x,y,z) + \varepsilon} \tag{2.2}$$

   dengan ε merupakan konstanta kecil yang ditambahkan untuk menghindari ketidakstabilan numerik ketika nilai intensitas mendekati nol atau bernilai negatif. Transformasi ini mengompresi rentang dinamis citra sehingga pengaruh nilai intensitas yang sangat tinggi dapat dikurangi.

3. *Logarithm Filter*

   Transformasi logaritmik diberikan oleh:

$$I_{\log}(x,y,z) = \log\left(I(x,y,z) + \varepsilon\right) \tag{2.3}$$

   di mana $\log(\cdot)$ merupakan fungsi logaritma natural dan ε digunakan untuk mencegah nilai tak terdefinisi ketika intensitas mendekati nol. Transformasi ini membantu mengurangi variasi intensitas yang besar serta menonjolkan perbedaan kecil pada area dengan intensitas rendah.

4. *Exponential Filter*

   Transformasi eksponensial dinyatakan sebagai:

$$I_{\exp}(x,y,z) = \exp\left(I(x,y,z)\right) \tag{2.4}$$

   di mana $\exp(\cdot)$ merupakan fungsi eksponensial. Transformasi ini memperbesar perbedaan intensitas pada nilai yang lebih tinggi sehingga dapat menonjolkan variasi halus pada area tumor dengan intensitas tinggi.

5. *Gradient Filter*

   Filter gradien menghitung besarnya perubahan intensitas voxel pada tiga arah spasial:

$$I_{\text{grad}} = \sqrt{\left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2 + \left(\frac{\partial I}{\partial z}\right)^2} \tag{2.5}$$

di mana $\frac{\partial I}{\partial x}$, $\frac{\partial I}{\partial y}$, dan $\frac{\partial I}{\partial z}$ merupakan turunan parsial fungsi intensitas terhadap masing-masing arah spasial. Nilai gradien ini menggambarkan kekuatan perubahan intensitas pada citra yang sering berkaitan dengan batas struktur tumor atau variasi tekstur internal.

6. *Laplacian of Gaussian* (LoG)

   Filter LoG menggabungkan proses perataan Gaussian dengan operator Laplacian:

$$I_{\text{LoG}} = \nabla^2 (G_\sigma * I) \tag{2.6}$$

   dengan operator Laplacian didefinisikan sebagai:

$$\nabla^2 I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2} + \frac{\partial^2 I}{\partial z^2} \tag{2.7}$$

   Simbol $*$ menunjukkan operasi konvolusi antara citra $I(x,y,z)$ dan kernel Gaussian G_σ yang didefinisikan sebagai:

$$G_\sigma(x,y,z) = \frac{1}{(2\pi\sigma^2)^{\frac{3}{2}}} \exp\left(-\frac{x^2+y^2+z^2}{2\sigma^2}\right) \tag{2.8}$$

   Parameter σ mengontrol tingkat perataan citra. Nilai σ yang lebih besar menghasilkan perataan yang lebih kuat sehingga memungkinkan deteksi pola struktural pada skala yang lebih besar.

7. *Wavelet Decomposition*

   Transformasi wavelet tiga dimensi dinyatakan sebagai:

$$I_{\text{wavelet}} = W_{\text{3D}}\{I(x,y,z)\} \tag{2.9}$$

di mana $W_{3D}\{\cdot\}$ merupakan transformasi wavelet tiga dimensi yang memecah citra menjadi beberapa komponen frekuensi. Dekomposisi ini menghasilkan sub-band frekuensi rendah dan tinggi yang memungkinkan analisis tekstur tumor pada berbagai skala spasial.

Dengan menerapkan berbagai transformasi tersebut, fitur radiomik dapat menangkap variasi tekstur dan struktur tumor pada berbagai skala spasial. Fitur-fitur yang dihasilkan kemudian digunakan sebagai masukan pada tahap seleksi fitur dan pembangunan model klasifikasi.

2.5 Seleksi Fitur

Seleksi fitur (*feature selection*) merupakan tahapan fundamental dalam pemrosesan data pra-analisis yang bertujuan untuk mengidentifikasi dan memilih subset fitur optimal S dari himpunan fitur asli F. Kualitas subset $S \subset F$ diukur dari kemampuannya untuk mempertahankan, atau bahkan meningkatkan, kinerja prediktif model klasifikasi sambil mengurangi dimensi input data. Proses ini menjadi sangat penting di tengah ledakan data berdimensi tinggi (*high-dimensional data*), terutama dalam bidang Genomika, di mana jumlah gen dapat mencapai ribuan, dan Radiomika yang menghasilkan ratusan hingga ribuan fitur tekstur dan statistik dari citra medis [28].

Tantangan utama dalam analisis data berdimensi tinggi adalah fenomena *curse of dimensionality*, di mana volume ruang pencarian meningkat secara eksponensial seiring bertambahnya dimensi, sehingga data menjadi semakin jarang dan proses analisis komputasional menjadi tidak praktis. Dengan melakukan seleksi fitur, manfaat utama yang diperoleh meliputi pencegahan *overfitting* (terutama ketika jumlah fitur jauh melebihi jumlah sampel), peningkatan interpretasi model melalui identifikasi fitur-fitur yang paling diskriminatif (misalnya, identifikasi biomarker spesifik), serta percepatan waktu komputasi dalam proses pelatihan model [29].

2.5.1 Taksonomi Metode Seleksi Fitur: Filter, Wrapper, dan Embedded

Metode seleksi fitur diklasifikasikan menjadi tiga kategori utama, yang dibedakan berdasarkan cara evaluasi subset fitur dilakukan, khususnya

hubungannya dengan algoritma pembelajaran mesin target [28]. Pemilihan kategori ini sangat menentukan biaya komputasi dan kualitas hasil akhir seleksi.

Metode Filter (Metode Penyaring)

Metode *filter* mengevaluasi relevansi fitur berdasarkan properti intrinsik data, independen dari model pembelajaran mesin yang akan digunakan [28]. Fitur-fitur diberi peringkat menggunakan metrik statistik seperti korelasi, uji χ^2, atau *mutual information*. Keunggulan utama metode Filter adalah efisiensi komputasi yang tinggi, menjadikannya pilihan ideal untuk data berdimensi sangat tinggi. Fitur yang dipilih cenderung lebih *generalizable* karena tidak terikat pada karakteristik spesifik model klasifikasi tertentu [28]. Minimum Redundancy Maximum Relevance (mRMR) adalah contoh inti dari metode Filter.

Metode Wrapper (Metode Pembungkus)

Sebaliknya, metode *wrapper* menggunakan kinerja model klasifikasi yang diuji (misalnya akurasi *cross-validation* atau AUC) sebagai fungsi objektif untuk memandu strategi pencarian. Metode ini secara eksplisit mempertimbangkan interaksi antar fitur karena subset dievaluasi secara kolektif berdasarkan dampaknya pada kinerja prediksi [28]. Meskipun Wrapper telah terbukti menghasilkan kinerja prediktif yang lebih tinggi, kelemahan kritikalnya adalah biaya komputasi yang substansial, sebab memerlukan pelatihan dan pengujian model berulang kali pada setiap iterasi pencarian [30]. *Genetic algorithm* (GA) merupakan salah satu strategi pencarian heuristik paling efektif dalam kategori Wrapper.

Metode Embedded (Metode Tertanam)

Metode *embedded* menawarkan kompromi seimbang dengan mengintegrasikan proses seleksi fitur langsung ke dalam pelatihan model, di mana fitur yang paling relevan dipelajari selama proses optimasi, seperti yang terlihat pada teknik regularisasi L_1 (*lasso*) [30].

Perbedaan mendasar antara Filter dan Wrapper—efisiensi komputasi versus akurasi optimasi—menentukan kebutuhan metodologis untuk studi yang berurusan dengan data berdimensi tinggi dan kompleksitas interaksi fitur. Metode Filter (seperti mRMR) menawarkan efisiensi tinggi namun dapat mengabaikan sinergi fitur, sementara metode Wrapper (seperti GA) menangkap sinergi namun

menghadapi kendala skalabilitas komputasi. Kontras ini membentuk justifikasi teknis untuk pengembangan pendekatan hibrida.[30]

2.5.2 Minimum Redundancy Maximum Relevance (mRMRe)

Minimum Redundancy Maximum Relevance (mRMRe) adalah salah satu metode seleksi fitur berbasis Filter yang paling efektif, terutama dalam aplikasi yang ditandai oleh banyaknya fitur yang saling berkorelasi atau berlebihan (*redundant*), seperti genetika dan radiomik [28].

A Prinsip Dasar dan Filsafat Seleksi Fitur Berbasis Informasi

mRMRe didasarkan pada kriteria ketergantungan statistik maksimal (*maximal statistical dependency criterion*). Secara teoritis, seleksi fitur yang optimal seharusnya memaksimalkan ketergantungan antara subset fitur yang dipilih dan variabel target. Karena kriteria ini sulit diimplementasikan secara langsung, Peng, Long, dan Ding (2005) menurunkan kriteria yang setara, yaitu mRMR, untuk seleksi fitur inkremental tingkat pertama [31]. Karya fundamental ini awalnya diterapkan pada data ekspresi gen mikroarray, dengan tujuan memilih subset kecil dari ribuan gen untuk klasifikasi fenotipe yang akurat [28]. Kriteria mRMR memastikan bahwa gen yang dipilih memberikan cakupan ruang fitur yang lebih seimbang dan mampu menangkap karakteristik fenotipe yang lebih luas, sehingga meningkatkan performa prediksi [28].

B Informasi Mutual (*Mutual Information*, MI) sebagai Metrik Ketergantungan

mRMRe memanfaatkan Informasi Mutual (MI), $I(X;Y)$, sebagai metrik utama untuk mengukur ketergantungan antar variabel [32]. MI mampu menangkap hubungan non-linear yang kompleks antara fitur dan target. Dalam kerangka mRMRe terdapat dua komponen krusial:

1. Relevansi Maksimum (*MaxRelevance*): $I(x_j;y)$, yaitu ketergantungan antara fitur ke-j dan target.

2. Redundansi Minimum (*MinRedundancy*): $I(x_j;x_k)$, yaitu MI antar fitur dalam subset yang dipilih.

C Formulasi Matematis Kriteria mRMRe

Tujuan mRMRe adalah memilih subset fitur S yang memaksimalkan relevansi rata-rata $D(S)$ dan meminimalkan redundansi rata-rata $R(S)$ [32].

$$D(S) = \frac{1}{|S|} \sum_{x_j \in S} I(x_j; y) \tag{2.10}$$

di mana $I(x_j; y)$ menyatakan *mutual information* antara fitur x_j dan label target y, sedangkan $|S|$ menyatakan jumlah fitur dalam subset S.

$$R(S) = \frac{1}{|S|^2} \sum_{x_j, x_k \in S} I(x_j; x_k) \tag{2.11}$$

di mana $I(x_j; x_k)$ merupakan *mutual information* antara dua fitur yang mengukur tingkat redundansi antar fitur.

Fungsi objektif kombinasi dengan skema *Mutual Information Difference (MID)* diformulasikan sebagai:

$$\max_S \quad \frac{1}{|S|} \sum_{x_j \in S} I(x_j; y) - \frac{1}{|S|^2} \sum_{x_j, x_k \in S} I(x_j; x_k) \tag{2.12}$$

Fungsi objektif ini memaksimalkan relevansi fitur terhadap label target sekaligus meminimalkan redundansi antar fitur dalam subset yang dipilih.

D Strategi Pencarian Sekuensial Inkremental

Karena jumlah total subset fitur adalah 2^M, pencarian secara *exhaustive* tidak praktis untuk data berdimensi tinggi. Oleh karena itu, mRMRe menggunakan pendekatan *greedy incremental search* [31, yang menurunkan kompleksitas komputasi dari $O(2^M)$ menjadi sekitar $O(M \cdot |S|)$.

Pada setiap iterasi, fitur baru dipilih berdasarkan kriteria berikut:

$$\max_{x_j \in F \setminus S} \left(I(x_j; y) - \frac{1}{|S|} \sum_{x_k \in S} I(x_j; x_k) \right) \tag{2.13}$$

Kriteria ini memilih fitur yang memiliki relevansi tinggi terhadap target sekaligus redundansi yang rendah terhadap fitur yang telah dipilih sebelumnya.

Meskipun efisien dan skalabel, pendekatan *greedy* ini merupakan aproksimasi sehingga tidak menjamin tercapainya solusi optimum global. Oleh karena itu, metode optimasi berbasis *wrapper* seperti *Genetic Algorithm* (GA) sering digunakan untuk mengeksplorasi kombinasi fitur yang lebih optimal [33,34].

2.5.3 Algoritma Genetika (*Genetic Algorithm*/GA)

Proses *Genetic Algorithm* (GA) dalam seleksi fitur melibatkan siklus berulang yang terdiri dari inisialisasi, evaluasi, seleksi, dan reproduksi [34].

A Inisialisasi Populasi dan Pengkodean (*Encoding*)

Langkah awal melibatkan inisialisasi populasi awal yang terdiri dari solusi kandidat. Dalam seleksi fitur, setiap solusi, yang disebut individu atau *kromosom*, mewakili satu subset fitur spesifik. Skema pengkodean paling umum adalah vektor biner [35]. Vektor biner ini memiliki panjang N (jumlah total fitur dalam himpunan data). Jika sebuah fitur disertakan dalam subset, nilai bit adalah 1; jika dikecualikan, nilai bit adalah 0 [35].

B Fungsi Kebugaran (*Fitness Function*): Pusat Biaya Komputasi

Fungsi kebugaran (*fitness function*) merupakan komponen paling kritis sekaligus paling mahal secara komputasi dalam algoritma *Genetic Algorithm* (GA) [35]. Evaluasi kebugaran $F(S_i)$ melibatkan pelatihan model klasifikasi (misalnya SVM atau *Naive Bayes*) pada subset fitur S_i dan pengujiannya menggunakan *cross-validation*. Skor performa model—misalnya akurasi, AUC, atau F1-score—digunakan sebagai ukuran kebugaran [35].

Sifat repetitif dari proses pelatihan dan evaluasi model ini menyebabkan terbentuknya *computational bottleneck*, sehingga penerapan GA murni pada dataset berdimensi tinggi menjadi sangat mahal secara komputasi [28].

Desain fungsi kebugaran yang optimal menuntut keseimbangan antara kinerja model, jumlah fitur yang dipilih, serta efisiensi komputasi [34]. Secara umum, fungsi kebugaran dalam konteks seleksi fitur berbasis GA dapat diformulasikan sebagai berikut:

$$F(S) = \alpha \cdot P(S) - \beta \cdot \frac{|S|}{M} + \gamma \cdot C(S) \tag{2.14}$$

dengan:

1. $F(S)$ adalah nilai kebugaran dari subset fitur S,
2. $P(S)$ menyatakan performa model klasifikasi yang dilatih menggunakan subset fitur S (misalnya akurasi, AUC, atau F1-score),
3. $|S|$ merupakan jumlah fitur yang dipilih dalam subset,
4. M adalah jumlah total fitur awal,
5. $C(S)$ merepresentasikan efisiensi komputasi, misalnya waktu pelatihan atau penggunaan sumber daya,
6. α, β, dan γ merupakan parameter pembobot yang mengatur kontribusi masing-masing komponen dalam fungsi kebugaran.

Penentuan nilai bobot yang tepat menjadi salah satu tantangan utama dalam perancangan fungsi kebugaran. Pengaturan parameter yang sesuai dapat memengaruhi dinamika evolusi populasi serta kualitas solusi akhir yang dihasilkan. Oleh karena itu, fungsi kebugaran berperan sebagai mekanisme pengendali utama yang mengarahkan proses evolusi GA menuju subset fitur yang kompak namun tetap memberikan performa klasifikasi yang optimal [34].

C Operator Seleksi (*Selection Operator*)

Operator Seleksi bertugas memilih individu terbaik untuk menghasilkan generasi berikutnya. Mekanisme umum meliputi *Roulette Wheel Selection* dan *Tournament Selection* [35]. Pendekatan probabilistik ini memberikan keseimbangan antara eksploitasi solusi unggul dan eksplorasi solusi alternatif [35].

D Operator Rekombinasi dan Mutasi (*Crossover and Mutation*)

Individu baru dihasilkan melalui operasi genetika:

1. *Crossover*: menggabungkan materi genetik dua orang tua untuk menghasilkan keturunan baru.
2. *Mutation*: mempertahankan keragaman dengan membalikkan nilai bit ($0 \leftrightarrow 1$) pada probabilitas kecil.

Mutasi mencegah konvergensi prematur dan membantu menghindari jebakan *local optimum* [35].

Tabel 2.2 merangkum peran dan mekanisme implementasi setiap komponen utama GA dalam konteks seleksi fitur.

Tabel 2.2. Komponen Utama dan Implementasi dalam *Genetic Algorithm* untuk Seleksi Fitur

Tahap GA	**Tujuan dalam Seleksi Fitur**	**Detail Implementasi (*Encoding* & Mekanisme)**	**Implikasi Kebugaran / Pencarian**
Encoding	Representasi subset fitur	Vektor biner (0/1) sepanjang N fitur [34]	Memungkinkan operasi genetika diskrit yang efisien.
Fitness Function	Mengukur kualitas subset fitur	Akurasi klasifikasi, AUC, atau F1-score menggunakan *cross-validation* [34]	Menggerakkan pencarian menuju kinerja model yang optimal (*eksploitasi*).
Selection	Memilih individu terbaik sebagai induk	*Roulette Wheel* atau *Tournament Selection* [34]	Memastikan individu paling fit memiliki peluang reproduksi lebih tinggi.
Mutation	Mempertahankan keragaman populasi	*Flipping bit* ($0 \leftrightarrow 1$) pada probabilitas rendah [34]	Mencegah konvergensi prematur dan membantu menghindari optimum lokal (*eksplorasi*).

2.5.4 Regularisasi L1-LASSO (*Least Absolute Shrinkage and Selection Operator*)

Regularisasi L1-LASSO merupakan metode seleksi fitur berbasis *Embedded* yang menggabungkan proses seleksi fitur dan pelatihan model regresi secara simultan [10]. Metode ini mengatasi keterbatasan regresi linear standar dengan menambahkan penalti norma L_1 pada fungsi objektif sehingga koefisien dari fitur yang kurang relevan terdorong menjadi nol [10]. Dengan demikian, LASSO tidak hanya melakukan estimasi parameter tetapi juga secara otomatis melakukan seleksi fitur.

Formulasi asli LASSO yang diperkenalkan oleh Tibshirani (1996) dinyatakan sebagai berikut:

$$(\hat{\alpha}, \hat{\beta}) = \arg\min_{\alpha,\beta} \left\{ \sum_{i=1}^{N} \left(y_i - \alpha - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 \right\} \quad \text{subject to} \quad \sum_{j=1}^{p} |\beta_j| \leq t \tag{2.15}$$

di mana N menyatakan jumlah sampel, p adalah jumlah fitur, dan t merupakan parameter batas (*constraint*) yang mengontrol jumlah total nilai absolut koefisien regresi. Semakin kecil nilai t, semakin kuat efek regularisasi yang diterapkan sehingga lebih banyak koefisien β_j didorong menuju nol.

Dalam implementasi modern, formulasi tersebut sering dinyatakan dalam bentuk penalti eksplisit menggunakan parameter regularisasi λ sebagai berikut:

$$\min_{(\beta_0,\beta)} \left[\frac{1}{2N} \sum_{i=1}^{N} \left(y_i - \beta_0 - x_i^T \beta \right)^2 + \lambda \|\beta\|_1 \right] \tag{2.16}$$

dengan $\|\beta\|_1 = \sum_{j=1}^{p} |\beta_j|$ merupakan norma L_1 dari vektor koefisien regresi. Parameter λ mengontrol tingkat penalti terhadap kompleksitas model: semakin besar nilai λ, semakin kuat penyusutan koefisien yang terjadi.

Formulasi ini juga dapat dipandang sebagai kasus khusus dari kerangka *Elastic Net* yang diperkenalkan oleh Friedman, Hastie, dan Tibshirani (2010), dengan fungsi penalti sebagai berikut:

$$P_\alpha(\beta) = (1-\alpha)\frac{1}{2}\|\beta\|_2^2 + \alpha\|\beta\|_1 \tag{2.17}$$

Pada kasus LASSO murni, parameter $\alpha = 1$, sehingga penalti hanya bergantung pada norma L_1 dari koefisien β. Sebaliknya, ketika $\alpha = 0$, formulasi tersebut ekuivalen dengan regularisasi Ridge.

Dalam implementasi modern seperti `LassoCV` pada pustaka *scikit-learn*, parameter λ ditentukan secara otomatis melalui prosedur *k-fold cross-validation*. Proses optimasi dilakukan menggunakan algoritme *coordinate descent*, yang memungkinkan perhitungan jalur regularisasi (*regularization paths*) secara efisien untuk berbagai nilai λ [36,37].

Pendekatan LASSO sangat efektif untuk data radiomik berdimensi tinggi yang sering memiliki korelasi kuat antar fitur. Dengan mendorong sebagian koefisien menuju nol, metode ini menghasilkan model yang lebih sederhana, stabil, dan memiliki kemampuan generalisasi yang lebih baik.

2.6 Klasifikasi Stadium

Klasifikasi stadium pada kanker ginjal *clear cell renal cell carcinoma* (ccRCC) merupakan proses penting untuk menentukan strategi penanganan klinis dan prognosis pasien. Dengan kemajuan dalam analisis radiomik, fitur kuantitatif yang diekstraksi dari citra medis seperti CT dan MRI dapat digunakan untuk mendukung proses klasifikasi ini. Berbagai algoritma komputasi, mulai dari *machine learning* tradisional hingga *deep learning*, telah diterapkan untuk mengolah fitur radiomik dan menghasilkan prediksi yang akurat mengenai karakteristik tumor.

2.6.1 Algoritma Support Vector Machine (SVM)

Support Vector Machine (SVM) merupakan algoritma pembelajaran mesin yang kuat, awalnya diperkenalkan oleh Cortes dan Vapnik [38]. Algoritma ini dirancang terutama untuk masalah klasifikasi dan regresi, dengan prinsip utama membangun batas keputusan (*hyperplane*) yang optimal untuk memisahkan kelas-kelas data dengan margin terbesar. Dalam konteks aplikasi medis, termasuk klasifikasi *staging*, SVM dikenal karena kemampuan generalisasinya yang tinggi, bahkan pada dataset dengan jumlah fitur tinggi dan sampel terbatas [39].

A Dasar Teori Support Vector Machine

Prinsip dasar SVM adalah memetakan vektor input $x \in X$ ke ruang fitur berdimensi tinggi, di mana di ruang fitur tersebut dapat dibangun permukaan keputusan linier (*hyperplane*). Untuk data yang terpisah secara linier, tujuannya adalah menemukan vektor bobot w dan bias b yang mendefinisikan *hyperplane* optimal:

$$w \cdot x + b = 0 \tag{2.18}$$

Hyperplane ini harus memenuhi kendala pemisahan kelas:

$$y_i(w \cdot x_i + b) \geq 1, \quad i = 1, \ldots, l \tag{2.19}$$

di mana $y_i \in \{-1, 1\}$ adalah label kelas dan x_i adalah vektor fitur pelatihan. Tujuan SVM adalah memaksimalkan margin, yaitu jarak terpendek antara *hyperplane* dan titik data pelatihan terdekat. Titik data terdekat ini disebut *support vectors* [38]. Memaksimalkan margin $\frac{2}{||w||}$ ekuivalen dengan meminimalkan norma kuadrat bobot w:

$$\min_{w,b} \frac{1}{2}\|w\|^2 \tag{2.20}$$

di bawah kendala $y_i(w \cdot x_i + b) \geq 1$.

B *Soft Margin* dan Penalti Kesalahan

Dalam kasus di mana data pelatihan tidak terpisah secara linier, atau untuk mencapai generalisasi yang lebih baik dengan mengorbankan beberapa kesalahan pelatihan [38], konsep *margin lunak (soft margin)* diperkenalkan. Hal ini dicapai dengan memperkenalkan variabel *slack* (ξ_i) non-negatif yang mengukur tingkat pelanggaran kendala pemisahan untuk setiap titik data.

Masalah optimasi *soft margin* dalam formulasi primal adalah [39]:

$$\min_{w,b,\xi} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l} \xi_i \tag{2.21}$$

dengan kendala:

$$\begin{aligned} y_i(w^T \phi(x_i) + b) &\geq 1 - \xi_i \\ \xi_i &\geq 0 \end{aligned} \tag{2.22}$$

Konstanta penalti C adalah parameter yang ditentukan pengguna yang merepresentasikan pertukaran antara memaksimalkan ukuran margin dan meminimalkan jumlah variabel *slack* [39].

C Formulasi Dual

Masalah optimasi SVM diselesaikan secara efisien melalui formulasi dual, yang merupakan masalah pemrograman kuadratik cembung.

Untuk kasus *soft margin* (dengan penalti kesalahan linier), fungsi dual yang dimaksimalkan terhadap koefisien *Lagrange multiplier* α_i adalah:

$$\max_{\alpha} W(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j (x_i \cdot x_j) \tag{2.23}$$

Fungsional yang sama juga dapat diekspresikan dalam bentuk matriks/vektor yang lebih ringkas [38]:

$$W(\Lambda) = \Lambda^T 1 - \frac{1}{2} \Lambda^T D \Lambda \tag{2.24}$$

di mana $\Lambda^T = (\alpha_1, \ldots, \alpha_l)$, 1 adalah vektor unit, dan D adalah matriks $l \times l$ dengan elemen $D_{ij} = y_i y_j x_i \cdot x_j$.

Masalah maksimisasi ini tunduk pada kendala (untuk kasus *soft margin*):

$$\begin{aligned} 0 \leq \alpha_i &\leq C \\ \sum_{i=1}^{l} \alpha_i y_i &= 0 \end{aligned} \tag{2.25}$$

Hanya $\alpha_i > 0$ yang berkontribusi pada solusi, dan titik-titik data yang bersesuaian disebut *support vectors* [38].

D Ekstensi Non-Linear dan Fungsi Kernel

Untuk data yang tidak terpisah secara linier, SVM menggunakan pemetaan non-linier $\phi(x)$ untuk memindahkan data ke ruang fitur berdimensi tinggi. Teknik *Kernel Trick* memungkinkan produk titik di ruang fitur $\langle \phi(x_i), \phi(x_j) \rangle$ dihitung tanpa secara eksplisit mendefinisikan $\phi(x)$, dengan menggunakan fungsi kernel $K(x_i, x_j)$ [38].

Dengan menggunakan kernel, fungsi dual menjadi:

$$\max_{\alpha} W(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \tag{2.26}$$

Fungsi keputusan untuk titik data tak dikenal x diberikan oleh:

$$f(x) = \text{sign}\left(\sum_{i=1}^{l} \alpha_i y_i K(x_i, x) + b \right) \tag{2.27}$$

Berbagai fungsi kernel dapat digunakan untuk menangani pola data yang berbeda, antara lain:

1. *Linear Kernel*:

$$K(x_i, x_j) = x_i^T x_j \tag{2.28}$$

Kernel linear merupakan bentuk kernel paling sederhana karena hanya menghitung hasil perkalian titik (*dot product*) antara dua vektor data x_i dan x_j. Kernel ini tidak melakukan transformasi ke ruang fitur berdimensi lebih tinggi, sehingga batas keputusan yang dihasilkan tetap berupa *hyperplane* linier pada ruang asli data. Dengan demikian, kernel linear sesuai digunakan ketika data antar kelas sudah relatif dapat dipisahkan secara linier.

Kernel ini banyak digunakan pada data berdimensi tinggi dengan jumlah fitur besar, misalnya pada klasifikasi teks, genomik, maupun radiomik, karena proses komputasinya lebih efisien dibanding kernel non-linier. Selain itu, risiko *overfitting* cenderung lebih rendah karena kompleksitas model lebih sederhana. Namun, apabila pola hubungan antar fitur bersifat kompleks dan

non-linier, kernel linear sering kali kurang mampu memberikan performa optimal.

2. *Polynomial Kernel*:

$$K(x_i, x_j) = (x_i^T x_j + c)^d \tag{2.29}$$

Kernel polynomial memperluas kernel linear dengan menambahkan konstanta c dan pangkat derajat d. Melalui transformasi ini, model mampu membentuk batas keputusan non-linier dengan mempertimbangkan interaksi antar fitur hingga orde tertentu. Sebagai contoh, jika $d = 2$, maka model dapat menangkap hubungan kuadratik antar variabel.

Parameter c berfungsi mengatur kontribusi komponen linear terhadap kernel, sedangkan parameter d menentukan tingkat kompleksitas model. Nilai d yang kecil menghasilkan model lebih sederhana, sedangkan nilai d yang terlalu besar dapat meningkatkan risiko *overfitting*. Kernel polynomial cocok digunakan ketika hubungan antar variabel tidak sepenuhnya linier, tetapi masih dapat direpresentasikan melalui interaksi polinomial.

Meskipun fleksibel, kernel ini umumnya membutuhkan waktu komputasi lebih besar dibanding kernel linear, terutama pada data berukuran besar.

3. *Radial Basis Function (RBF) Kernel* [39]:

$$K(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2) \tag{2.30}$$

Kernel RBF merupakan salah satu kernel yang paling umum digunakan dalam SVM karena memiliki kemampuan tinggi dalam menangani data non-linier. Kernel ini mengukur kemiripan dua data berdasarkan jarak Euclidean kuadrat. Nilai kernel akan mendekati 1 apabila dua data saling berdekatan, dan mendekati 0 apabila jaraknya jauh.

Melalui pendekatan ini, data secara implisit dipetakan ke ruang berdimensi sangat tinggi tanpa perlu menghitung transformasi secara eksplisit, sehingga model dapat membentuk batas keputusan yang kompleks dan fleksibel.

Parameter utama pada kernel ini adalah γ, yang mengontrol jangkauan pengaruh suatu data terhadap lingkungan sekitarnya.

Nilai γ kecil menghasilkan batas keputusan yang lebih halus dan general, sedangkan nilai γ besar membuat model sangat sensitif terhadap data pelatihan sehingga berpotensi menyebabkan *overfitting*. Karena keseimbangan antara fleksibilitas dan performanya, kernel RBF sering menjadi pilihan utama pada berbagai kasus klasifikasi medis, termasuk analisis citra radiologi dan radiomik.

4. *Sigmoid Kernel*:

$$K(x_i, x_j) = \tanh(\kappa x_i^T x_j + \theta) \tag{2.31}$$

Kernel sigmoid menggunakan fungsi hiperbolik tangen (*hyperbolic tangent*) yang memiliki kemiripan dengan fungsi aktivasi pada jaringan saraf tiruan. Karena itu, kernel ini sering dipandang sebagai penghubung konseptual antara SVM dan *neural network*.

Parameter κ mengontrol skala kemiringan fungsi, sedangkan θ merupakan parameter bias atau pergeseran. Dengan kombinasi parameter yang sesuai, kernel sigmoid mampu memodelkan hubungan non-linier antar fitur. Namun, performa kernel ini sangat sensitif terhadap pemilihan parameter, sehingga proses tuning menjadi penting.

Dalam praktiknya, kernel sigmoid lebih jarang digunakan dibanding kernel RBF atau linear karena kestabilan performanya cenderung lebih rendah pada beberapa jenis data. Meski demikian, kernel ini tetap relevan untuk eksperimen tertentu yang memiliki pola menyerupai aktivasi jaringan saraf.

5. *Laplacian Kernel*:

$$K(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|_1) \tag{2.32}$$

Kernel Laplacian memiliki bentuk serupa dengan kernel RBF, tetapi menggunakan jarak Manhattan (L_1) sebagai ukuran kedekatan antar data. Jarak Manhattan menghitung total selisih absolut pada setiap dimensi

fitur, sehingga lebih toleran terhadap adanya nilai ekstrem dibanding jarak Euclidean.

Karena karakteristik tersebut, kernel Laplacian sering dianggap lebih robust terhadap *outlier* atau distribusi data yang tidak terlalu halus. Parameter γ tetap berfungsi mengatur jangkauan pengaruh antar titik data. Nilai kecil menghasilkan model yang lebih umum, sedangkan nilai besar membuat batas keputusan lebih kompleks.

Kernel ini sesuai digunakan pada data yang memiliki banyak fitur dengan variasi besar atau ketika terdapat noise pada sebagian variabel.

6. *Chi-Square Kernel*:

$$K(x_i, x_j) = \exp\left(-\gamma \sum_k \frac{(x_{ik} - x_{jk})^2}{x_{ik} + x_{jk}}\right) \tag{2.33}$$

Kernel Chi-Square dirancang khusus untuk membandingkan dua vektor yang merepresentasikan distribusi frekuensi atau histogram. Pada kernel ini, perbedaan tiap komponen fitur dihitung relatif terhadap jumlah nilainya, sehingga sensitif terhadap perubahan distribusi antar bin histogram.

Karena sifat tersebut, kernel Chi-Square banyak digunakan pada bidang pengolahan citra, pengenalan objek, analisis tekstur, dan fitur radiomik berbasis histogram intensitas. Kernel ini efektif ketika data merepresentasikan proporsi atau frekuensi kemunculan suatu karakteristik, bukan sekadar nilai numerik biasa.

Parameter γ berfungsi mengatur sensitivitas kernel terhadap perbedaan distribusi. Nilai yang terlalu besar dapat menyebabkan model terlalu spesifik, sedangkan nilai terlalu kecil dapat membuat model kurang responsif terhadap variasi data.

E Cara Kerja Algoritma SVM

Secara umum, alur kerja algoritma *Support Vector Machine* (SVM) untuk klasifikasi dapat direpresentasikan dalam bentuk *pseudocode* sebagai berikut [38]:

```
Input  : Data latih (X, y), parameter C, kernel K
Output : Model SVM (alpha, b)
```

```
1. Lakukan pemetaan data ke ruang fitur:
   - Jika linier → gunakan X langsung
   - Jika non-linier → gunakan kernel K(x_i, x_j)

2. Formulasikan masalah optimasi:
   Maksimalkan fungsi dual:
   L(alpha) =  alpha_i - 1/2  alpha_i alpha_j y_i y_j K(x_i, x_j)

   Dengan kendala:
   0  alpha_i  C
    alpha_i y_i = 0

3. Selesaikan optimasi untuk mendapatkan alpha_i

4. Tentukan support vectors:
   - Ambil data dengan alpha_i > 0

5. Hitung bias (b):
   b = y_k -  alpha_i y_i K(x_i, x_k)

6. Bentuk fungsi keputusan:
   f(x) = sign(  alpha_i y_i K(x, x_i) + b )

7. Gunakan f(x) untuk klasifikasi data baru
```

Berdasarkan *pseudocode* tersebut, proses SVM dimulai dengan memetakan data ke ruang fitur, baik secara linier maupun menggunakan fungsi kernel untuk menangani data non-linier. Pada kasus linier, data dapat langsung diproses tanpa transformasi tambahan. Namun, untuk data non-linier, digunakan fungsi kernel $K(x_i, x_j)$ yang memungkinkan data diproyeksikan ke ruang berdimensi lebih tinggi tanpa menghitung transformasi secara eksplisit.

Selanjutnya, algoritma membentuk masalah optimasi dalam bentuk *dual problem* untuk mencari nilai koefisien α_i yang optimal. Parameter α_i merupakan *Lagrange multiplier* yang menunjukkan tingkat kontribusi masing-masing data terhadap model. Proses optimasi ini dikontrol oleh parameter regularisasi C, yang

mengatur keseimbangan antara margin maksimum dan kesalahan klasifikasi pada data latih.

Nilai α_i yang tidak nol akan mengidentifikasi *support vectors*, yaitu titik data yang berada paling dekat dengan batas keputusan (*hyperplane*). Titik-titik ini memiliki peran penting karena secara langsung menentukan posisi dan arah *hyperplane*. Sementara itu, data dengan $\alpha_i = 0$ tidak berkontribusi dalam pembentukan model sehingga dapat diabaikan dalam fungsi keputusan.

Setelah nilai α_i diperoleh, langkah berikutnya adalah menghitung bias b menggunakan salah satu *support vector*. Nilai bias ini berfungsi untuk mengatur posisi *hyperplane* agar dapat memisahkan kelas secara optimal. Perhitungan bias dilakukan dengan memasukkan nilai α_i, label y_i, serta fungsi kernel ke dalam persamaan yang telah ditentukan.

Model SVM yang terbentuk kemudian digunakan untuk melakukan klasifikasi data baru melalui fungsi keputusan berbasis kernel. Fungsi ini menghitung kombinasi dari kernel antara data uji dan *support vectors*, kemudian ditambahkan dengan bias b. Tanda dari hasil fungsi tersebut menentukan kelas dari data yang diprediksi.

Dengan pendekatan ini, SVM mampu menghasilkan batas keputusan dengan margin maksimum sehingga meningkatkan kemampuan generalisasi model. Selain itu, penggunaan kernel memungkinkan SVM menangani data berdimensi tinggi dan pola non-linier secara efektif. Hal ini menjadikan SVM sebagai salah satu metode yang kuat untuk klasifikasi pada berbagai kasus, termasuk data radiomik dalam citra medis.

F Aplikasi dalam Klasifikasi Medis

Support Vector Machine (SVM) merupakan alat pembelajaran mesin yang banyak digunakan dalam aplikasi perawatan kesehatan, meliputi diagnosis, prognosis, dan prediksi hasil penyakit [39]. SVM efektif dalam menangani data medis yang kompleks karena kemampuannya dalam mengatasi hubungan non-linear antara fitur dan kelas [39].

Interpretasi dan Ekstensi

1. Aplikasi Citra Medis dan Radiomik: SVM telah diterapkan secara luas untuk deteksi dan diagnosis penyakit pada pencitraan medis seperti MRI, X-ray, dan

CT-PET [39]. Algoritma ini sangat relevan dalam klasifikasi data *radiomik*, yang melibatkan ekstraksi dan analisis fitur kuantitatif dari citra medis untuk memprediksi prognosis atau *staging* penyakit [39]. Untuk interpretasi visual, pendekatan seperti *Quadtree* dapat digunakan untuk melokalisasi *Region of Interest* (ROI) atau wilayah diskriminatif yang mendasari prediksi SVM pada citra medis [40].

2. *Twin Support Vector Machine* (TWSVM): Varian ini menghasilkan dua *hyperplane* non-paralel dan umumnya lebih cepat dibandingkan dengan SVM standar, dengan aplikasi yang semakin berkembang di bidang medis [39].

3. *Cost-Sensitive SVM*: Model ini menangani masalah ketidakseimbangan kelas (*imbalanced datasets*) yang umum pada data kesehatan dengan memberikan bobot penalti yang berbeda (C_+ dan C_-) untuk misklasifikasi kelas minoritas, sehingga meningkatkan sensitivitas terhadap kelas target [39].

2.6.2 Algoritma Extreme Gradient Boosting (XGBoost)

Extreme Gradient Boosting (XGBoost) merupakan algoritma Ensemble Learning berbasis teknik *gradient boosting* yang dikembangkan oleh Chen dan Guestrin [41]. Berbeda dengan metode *ensemble* seperti *Random Forest* yang membangun pohon secara independen, XGBoost membangun pohon keputusan secara sekuensial, di mana setiap pohon baru bertujuan memperbaiki kesalahan prediksi dari model sebelumnya. Pendekatan ini memungkinkan model untuk belajar dari residual kesalahan sehingga meningkatkan akurasi prediksi secara bertahap.

XGBoost dikenal karena efisiensinya dalam komputasi, kemampuan regularisasi untuk menghindari *overfitting*, serta optimasi sistem yang memungkinkan pemrosesan *dataset* besar secara cepat [41,42].

A Dasar Teori XGBoost

Secara umum, model XGBoost merepresentasikan prediksi sebagai penjumlahan dari beberapa pohon keputusan. Jika terdapat K pohon, maka prediksi untuk suatu sampel $\mathbf{x}_i$ diberikan oleh:

$$\hat{y}_i = \sum_{k=1}^{K} f_k(\mathbf{x}_i), \quad f_k \in \mathscr{F} \tag{2.34}$$

di mana f_k merupakan fungsi pohon keputusan ke-k dan $\mathscr{F}$ adalah ruang fungsi yang berisi semua kemungkinan pohon regresi.

B Fungsi Objektif

Proses pelatihan XGBoost dilakukan dengan meminimalkan fungsi objektif yang terdiri dari dua komponen utama: fungsi kerugian (*loss function*) dan regularisasi kompleksitas model [41].

$$\mathscr{L} = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k) \tag{2.35}$$

di mana:

1. $l(y_i, \hat{y}_i)$ adalah fungsi kerugian yang mengukur perbedaan antara nilai aktual y_i dan prediksi $\hat{y}_i$,
2. $\Omega(f_k)$ adalah fungsi regularisasi yang mengontrol kompleksitas setiap pohon keputusan.

Fungsi regularisasi pada XGBoost didefinisikan sebagai:

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2 \tag{2.36}$$

di mana:

1. T adalah jumlah daun (*leaf nodes*) pada pohon,
2. w_j adalah bobot pada daun ke-j,
3. γ adalah penalti untuk menambah jumlah daun,
4. λ adalah parameter regularisasi untuk bobot daun.

Regularisasi ini membantu mengontrol kompleksitas model dan mengurangi risiko *overfitting*.

C Optimisasi dengan Gradient Boosting

XGBoost menggunakan pendekatan *gradient boosting*, di mana model diperbarui secara iteratif dengan menambahkan pohon baru yang meminimalkan residual kesalahan dari model sebelumnya. Fungsi objektif diaproksimasi menggunakan ekspansi deret Taylor orde dua untuk mempercepat proses optimasi [41].

$$\mathscr{L}^{(t)} \approx \sum_{i=1}^{n} \left[g_i f_t(\mathbf{x}_i) + \frac{1}{2} h_i f_t^2(\mathbf{x}_i) \right] + \Omega(f_t) \tag{2.37}$$

dengan:

1. $g_i = \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$ merupakan gradien pertama,
2. $h_i = \partial^2_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$ merupakan turunan kedua (*Hessian*).

Pendekatan ini memungkinkan XGBoost untuk mempercepat proses pembelajaran dan meningkatkan stabilitas optimisasi.

D Keunggulan XGBoost

XGBoost memiliki sejumlah keunggulan yang menjadikannya salah satu algoritma pembelajaran mesin paling populer [41,41]:

1. Regularisasi yang Kuat: Penambahan komponen regularisasi membantu mencegah *overfitting*.
2. Efisiensi Komputasi: Implementasi paralel dan optimasi memori memungkinkan pelatihan model pada *dataset* besar secara cepat.
3. Kemampuan Menangani Missing Value: XGBoost dapat secara otomatis menentukan arah terbaik dalam pohon keputusan ketika terdapat nilai yang hilang.
4. Feature Importance: Model dapat memberikan estimasi kontribusi masing-masing fitur terhadap prediksi.

E Cara Kerja Algoritma XGBoost

Secara umum, alur kerja algoritma *Extreme Gradient Boosting* (XGBoost) dalam proses pelatihan model dapat direpresentasikan dalam bentuk *pseudocode* sebagai berikut [41]:

```
Input  : Data latih (X, y), jumlah iterasi T, learning rate eta
Output : Model XGBoost (ensemble pohon)

1. Inisialisasi prediksi awal:
   y_hat_i = 0 untuk semua data

2. Untuk t = 1 hingga T:
   a. Hitung gradien (g_i) dan hessian (h_i):
      g_i = turunan pertama loss terhadap y_hat_i
      h_i = turunan kedua loss terhadap y_hat_i

   b. Bangun pohon keputusan f_t(x):
      - Tentukan split terbaik berdasarkan gain
      - Gunakan g_i dan h_i untuk menghitung kualitas split

   c. Hitung bobot pada setiap leaf:
      w_j = -  g_i / ( h_i + lambda)

   d. Update prediksi:
      y_hat_i = y_hat_i + eta * f_t(x_i)

3. Output model akhir:
   y_hat = f_t(x)
```

Berdasarkan *pseudocode* tersebut, proses XGBoost dimulai dengan inisialisasi prediksi awal $\hat{y}_i$ yang biasanya bernilai nol atau rata-rata dari target. Nilai awal ini digunakan sebagai *baseline* sebelum model mulai melakukan pembelajaran secara iteratif. Pendekatan ini memungkinkan model untuk secara bertahap memperbaiki kesalahan prediksi melalui penambahan model baru pada setiap iterasi.

Selanjutnya, pada setiap iterasi ke-t, algoritma menghitung gradien (g_i) dan *hessian* (h_i) dari fungsi kerugian terhadap prediksi sebelumnya. Gradien merupakan turunan pertama yang menunjukkan arah perubahan error, sedangkan *hessian* adalah turunan kedua yang memberikan informasi mengenai kelengkungan fungsi *loss*. Dengan memanfaatkan kedua informasi ini, XGBoost dapat melakukan optimisasi yang lebih akurat dan stabil dibandingkan metode *gradient boosting* tradisional.

Berdasarkan nilai gradien dan *hessian* tersebut, XGBoost membangun pohon keputusan baru $f_t(x)$. Proses ini dilakukan dengan mencari *split* terbaik pada setiap node menggunakan metrik *gain*, yang mengukur peningkatan kualitas prediksi setelah pemisahan data. Pemilihan *split* yang optimal memastikan bahwa setiap pohon yang dibangun berkontribusi secara maksimal dalam mengurangi error model.

Setelah struktur pohon terbentuk, bobot pada setiap *leaf node* dihitung menggunakan akumulasi gradien dan *hessian*. Bobot ini merepresentasikan nilai prediksi yang akan diberikan oleh setiap *leaf* terhadap data yang masuk ke dalamnya. Selain itu, parameter regularisasi λ digunakan untuk mengontrol kompleksitas model agar tidak terjadi *overfitting*.

Prediksi model kemudian diperbarui dengan menambahkan kontribusi dari pohon baru yang telah dibangun, yang dikalikan dengan *learning rate* (η). Parameter η berfungsi untuk mengatur seberapa besar pengaruh setiap pohon terhadap model secara keseluruhan. Nilai η yang kecil akan membuat proses pembelajaran lebih lambat tetapi cenderung menghasilkan model yang lebih stabil.

Proses ini diulang hingga jumlah iterasi T tercapai atau hingga model mencapai kondisi konvergensi. Model akhir merupakan hasil penjumlahan dari seluruh pohon keputusan yang telah dibangun secara bertahap. Dengan pendekatan ini, XGBoost mampu menghasilkan model yang kuat dengan memperbaiki kesalahan residual pada setiap iterasi.

Selain itu, keunggulan utama XGBoost terletak pada penggunaan informasi turunan kedua (*hessian*) serta mekanisme regularisasi yang baik. Hal ini membuat proses pelatihan menjadi lebih efisien, stabil, dan tahan terhadap *overfitting*. Oleh karena itu, XGBoost banyak digunakan dalam berbagai permasalahan klasifikasi dan regresi, termasuk pada analisis data berdimensi tinggi seperti fitur radiomik.

F Penerapan dalam Klasifikasi Medis

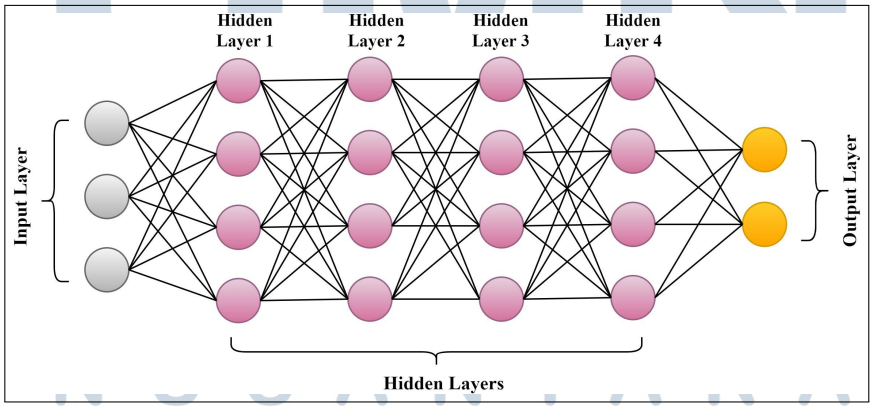
XGBoost telah digunakan secara luas dalam berbagai aplikasi medis, termasuk diagnosis penyakit, prediksi prognosis, dan analisis data biomedis berdimensi tinggi [42]. Dalam konteks analisis radiomik, algoritma ini mampu menangani jumlah fitur yang besar serta hubungan non-linear antar fitur, sehingga efektif digunakan untuk klasifikasi stadium penyakit atau prediksi hasil klinis.

2.6.3 Deep Neural Network (DNN)

Deep Neural Network (DNN) merupakan fondasi dari bidang *Deep Learning* (DL), yang merupakan subset dari *Machine Learning*. DL dicirikan oleh penggunaan jaringan saraf buatan dengan banyak lapisan tersembunyi (*hidden layers*) antara lapisan input dan output [42]. Kedalaman inilah yang memungkinkan DNN mempelajari representasi fitur yang kompleks dan hirarkis secara otomatis langsung dari data mentah, tanpa perlu rekayasa fitur manual.

Secara umum, arsitektur DNN terdiri dari tiga komponen utama, yaitu *input layer*, satu atau lebih *hidden layers*, dan *output layer*. Setiap neuron pada suatu lapisan terhubung dengan neuron pada lapisan berikutnya melalui bobot (*weights*) dan bias. Informasi diproses secara berlapis melalui operasi linier dan fungsi aktivasi non-linear.

Ilustrasi arsitektur *Deep Neural Network* ditunjukkan pada Gambar 2.5, yang menggambarkan alur propagasi data dari input hingga menghasilkan output prediksi.



Gambar 2.5. Ilustrasi arsitektur *Deep Neural Network* (DNN).

A Dasar Teori Deep Neural Network

Jaringan Saraf Tiruan (ANN) merupakan inti dari *Deep Learning*, yang dicirikan oleh kemampuannya untuk memodelkan kapabilitas otak manusia dan potensinya dalam prediksi dan klasifikasi, bahkan ketika data tidak tepat dan bising (*imprecise and noisy*) [43].

A.1 Definisi Neuron Buatan

Dasar dari jaringan saraf tiruan adalah neuron buatan. Sebuah neuron buatan dengan bobot $\omega_1, \ldots, \omega_n \in \mathbb{R}$, bias $b \in \mathbb{R}$, dan fungsi aktivasi $\rho : \mathbb{R} \to \mathbb{R}$ didefinisikan sebagai fungsi $f : \mathbb{R} \to \mathbb{R}$ yang diberikan oleh [43]:

$$f(x_1, \ldots, x_n) = \rho\left(\sum_{i=1}^{n} x_i \omega_i - b\right) = \rho(\langle x, \omega \rangle - b) \tag{2.38}$$

Fungsi aktivasi non-linier yang umum digunakan (selain fungsi Heaviside) meliputi:

1. Sigmoid: $\rho(x) = \frac{1}{1+e^{-x}}$ [43]

   Fungsi sigmoid memetakan nilai input ke dalam rentang antara 0 dan 1, sehingga sering digunakan pada lapisan output untuk kasus klasifikasi biner. Karakteristik kurva berbentuk S (*sigmoid curve*) memungkinkan model untuk merepresentasikan probabilitas. Namun, fungsi ini memiliki kelemahan berupa masalah *vanishing gradient*, di mana gradien menjadi sangat kecil pada nilai input yang ekstrem, sehingga dapat memperlambat proses pembelajaran pada jaringan yang dalam.

2. Rectified Linear Unit (ReLU): $\rho(x) = \max\{0, x\}$ [43]

   Fungsi ReLU merupakan fungsi aktivasi yang sangat populer dalam arsitektur DNN modern karena kesederhanaan dan efisiensinya. Fungsi ini menghasilkan nilai nol untuk input negatif dan mempertahankan nilai input untuk input positif, sehingga mampu mengatasi sebagian masalah *vanishing gradient*. Selain itu, ReLU mempercepat proses konvergensi selama pelatihan. Meskipun demikian, ReLU memiliki kelemahan yang dikenal sebagai *dying ReLU*, yaitu kondisi di mana neuron menjadi tidak aktif dan selalu menghasilkan nol.

A.2 Definisi Matematis Deep Neural Network (DNN)

Deep Neural Network (DNN) didefinisikan sebagai fungsi komposit dari banyak lapisan. Misalkan $d \in \mathbb{N}$ adalah dimensi lapisan input, L adalah jumlah lapisan, dan N_l adalah dimensi lapisan ke-l. Transformasi affine-linear pada lapisan ke-l (T_l) didefinisikan sebagai [43]:

$$\begin{aligned} T_l &: \mathbb{R}^{N_{l-1}} \to \mathbb{R}^{N_l} \\ T_l x &= W^{(l)} x + b^{(l)} \end{aligned} \tag{2.39}$$

di mana $W^{(l)} \in \mathbb{R}^{N_l \times N_{l-1}}$ adalah matriks bobot dan $b^{(l)} \in \mathbb{R}^{N_l}$ adalah vektor bias dari lapisan ke-l [43].

Maka, DNN $\Phi : \mathbb{R}^d \to \mathbb{R}^{N_L}$ dengan kedalaman L didefinisikan oleh komposisi fungsi-fungsi ini [43]:

$$\Phi(x) = T_L \rho(T_{L-1} \rho(\dots \rho(T_1(x)) \dots)) \tag{2.40}$$

A.3 Pelatihan Jaringan dan Optimasi

Pelatihan DNN bertujuan untuk menemukan bobot dan bias optimal dengan meminimalkan risiko empiris melalui optimasi. Masalah optimasi umum dalam pelatihan DNN, yang melibatkan Fungsi Kerugian ($\mathscr{L}$) dan suku regularisasi ($\mathscr{P}$), adalah [43]:

$$\min_{W^{(l)}, b^{(l)})_l} \sum_{l=1}^{m} \mathscr{L}(\Phi(W^{(l)}, b^{(l)})_l(x_l) - y^{(l)}) + \lambda \mathscr{P}((W^{(l)} + b^{(l)})_l) \tag{2.41}$$

Salah satu pendekatan algoritmik yang umum untuk menyelesaikan masalah ini adalah *Stochastic Gradient Descent* (SGD), yang digunakan karena efisiensinya dalam mengatasi skala data yang besar (m yang besar), dengan cara mengambil hanya sebagian gradien secara acak di setiap iterasi [43].

B Cara Kerja Deep Neural Network

Secara umum, proses pelatihan *Deep Neural Network* (DNN) dapat direpresentasikan dalam bentuk pseudocode sebagai berikut [43]:

```
Input  : Data latih (X, y), bobot W, bias b, learning rate eta
Output : Model DNN terlatih

1. Inisialisasi bobot W dan bias b secara acak

2. Untuk setiap epoch:
   a. Forward propagation:
      - Hitung output tiap layer:
        z = W.x + b
        a = fungsi aktivasi(z)

   b. Hitung loss:
      L = loss(y, y_hat)

   c. Backpropagation:
      - Hitung gradien terhadap W dan b
      - Gunakan chain rule untuk setiap layer

   d. Update parameter:
      W = W - eta * dW
      b = b - eta * db

3. Ulangi hingga konvergen atau epoch selesai

4. Gunakan model untuk prediksi:
   y_hat = forward propagation(X baru)
```

Berdasarkan pseudocode tersebut, proses DNN dimulai dengan inisialisasi bobot (W) dan bias (b) secara acak. Inisialisasi ini penting untuk memastikan bahwa setiap neuron dapat belajar fitur yang berbeda selama proses pelatihan. Jika semua bobot diinisialisasi dengan nilai yang sama, maka jaringan tidak akan mampu mempelajari representasi yang kompleks.

Selanjutnya, dilakukan *forward propagation*, yaitu proses mengalirkan data dari *input layer* menuju *output layer* melalui beberapa *hidden layers*. Pada setiap neuron, dilakukan operasi linier berupa kombinasi bobot dan input, yaitu

$$z = Wx + b \tag{2.42}$$

yang kemudian diterapkan fungsi aktivasi non-linear

$$a = f(z) \tag{2.43}$$

Fungsi aktivasi seperti ReLU, sigmoid, atau tanh digunakan untuk memungkinkan model menangkap hubungan non-linear dalam data.

Setelah diperoleh prediksi ($\hat{y}$), dilakukan perhitungan fungsi kerugian (*loss*) untuk mengukur selisih antara prediksi dan nilai aktual (y). Fungsi loss yang umum digunakan antara lain *Mean Squared Error* (MSE) untuk regresi dan *Cross-Entropy* untuk klasifikasi. Nilai loss ini menjadi indikator seberapa baik model dalam melakukan prediksi.

Tahap berikutnya adalah *backpropagation*, yaitu proses menghitung gradien error terhadap setiap parameter dalam jaringan. Proses ini dilakukan dengan menyebarkan error dari *output layer* ke *input layer* menggunakan aturan rantai (*chain rule*). Tujuan utama dari tahap ini adalah untuk mengetahui seberapa besar perubahan bobot dan bias yang diperlukan untuk mengurangi nilai loss.

Secara matematis, untuk suatu neuron berlaku hubungan:

$$z = Wx + b, \quad a = f(z) \tag{2.44}$$

Gradien terhadap bobot dihitung menggunakan aturan rantai sebagai berikut:

$$\frac{\partial L}{\partial W} = \frac{\partial L}{\partial a} \cdot \frac{\partial a}{\partial z} \cdot \frac{\partial z}{\partial W} \tag{2.45}$$

Sedangkan gradien terhadap bias diberikan oleh:

$$\frac{\partial L}{\partial b} = \frac{\partial L}{\partial a} \cdot \frac{\partial a}{\partial z} \tag{2.46}$$

Perhitungan ini dilakukan secara berurutan dari layer terakhir hingga layer pertama (*backward pass*), sehingga setiap parameter memperoleh informasi kontribusinya terhadap error.

Setelah gradien diperoleh, parameter model diperbarui menggunakan metode optimasi seperti *gradient descent*. Proses pembaruan dilakukan dengan rumus:

$$W = W - \eta \frac{\partial L}{\partial W} \tag{2.47}$$

$$b = b - \eta \frac{\partial L}{\partial b} \tag{2.48}$$

di mana η merupakan *learning rate* yang mengatur besar langkah pembaruan parameter. Nilai *learning rate* yang terlalu besar dapat menyebabkan proses tidak stabil, sedangkan nilai yang terlalu kecil akan memperlambat konvergensi.

Proses *forward propagation* dan *backpropagation* ini diulang untuk setiap epoch hingga model mencapai konvergensi atau jumlah iterasi tertentu. Dengan adanya banyak *hidden layers* dan fungsi aktivasi non-linear, DNN mampu mempelajari representasi fitur yang kompleks dan hirarkis. Hal ini menjadikan DNN sangat efektif dalam menangani data berdimensi tinggi dan pola non-linear, seperti pada fitur radiomik dalam citra medis.

C Penerapan dalam Klasifikasi Staging Medis

Dalam klasifikasi *staging* medis, DNN seringkali diterapkan ketika data berupa citra medis (menggunakan *Convolutional Neural Networks*/CNN) atau data sekuensial (menggunakan *Recurrent Neural Networks*/RNN). Untuk data tabular (seperti data klinis atau genetik), DNN dapat berfungsi sebagai alat klasifikasi yang ampuh, mampu menangkap interaksi non-linier yang kompleks antar fitur yang mungkin terlewatkan oleh metode ML tradisional [44].

Keunggulan utama *Deep Learning* dalam *staging* medis:

1. Representasi Fitur Otomatis: DNN dapat secara otomatis mempelajari fitur yang paling diskriminatif untuk klasifikasi *staging* tanpa memerlukan rekayasa fitur domain-spesifik yang intensif [44].
2. Kinerja pada Data Skala Besar: DNN umumnya unggul ketika tersedia data dalam jumlah besar, menjadikannya ideal untuk repositori data kesehatan modern yang luas.

3. Hubungan Non-Linear Kompleks: Kedalaman arsitektur memungkinkan pemodelan hubungan yang sangat non-linier antara input fitur dan label *staging*.

2.7 Metrik Evaluasi

Evaluasi model klasifikasi digunakan untuk mengukur seberapa baik algoritma dari model klasifikasi, seperti *Support Vector Machine* (SVM), *Random Forest* (RF), dan *Deep Neural Network* (DNN), dapat memprediksi kategori atau kelas yang benar dari data masukan yang diberikan. Dalam konteks penelitian ini, klasifikasi dilakukan berdasarkan fitur radiomik yang diekstraksi dari citra medis untuk menentukan stadium atau subtipe tumor. Evaluasi performa model sangat penting karena hasil prediksi klasifikasi dapat berdampak langsung terhadap diagnosis klinis dan pengambilan keputusan medis. Untuk itu, diperlukan metrik yang dapat memberikan gambaran menyeluruh mengenai kemampuan model dalam mendeteksi kasus positif maupun negatif dengan tingkat akurasi dan keandalan tinggi [45].

Metrik evaluasi yang digunakan mencakup *Accuracy*, *Precision*, *Recall* (*Sensitivity*), *F1-Score*, dan *ROC-AUC*. Kelima metrik ini dihitung berdasarkan nilai-nilai dari *confusion matrix*, yang terdiri dari empat komponen utama: True Positive (TP), True Negative (TN), False Positive (FP), dan False Negative (FN). Masing-masing komponen memiliki makna klinis tersendiri; misalnya, dalam klasifikasi tumor, TP menunjukkan jumlah pasien yang benar teridentifikasi memiliki tumor, sedangkan FN menunjukkan jumlah pasien dengan tumor yang gagal terdeteksi oleh model.

A Accuracy

Accuracy atau akurasi mengukur proporsi total prediksi yang benar terhadap seluruh jumlah sampel data. Metrik ini memberikan pandangan umum tentang seberapa sering model membuat prediksi yang benar, baik untuk kelas positif maupun negatif:

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}. \tag{2.49}$$

Akurasi adalah metrik paling sederhana namun sering digunakan sebagai

indikator awal performa model. Namun, dalam data medis yang sering kali tidak seimbang (misalnya, jumlah pasien sehat jauh lebih banyak daripada pasien dengan tumor), nilai akurasi saja tidak cukup untuk mengevaluasi kualitas model. Model dapat mencapai akurasi tinggi hanya dengan memprediksi mayoritas kelas, sementara gagal mendeteksi kasus minoritas yang justru lebih penting secara klinis [39,45].

B Precision

Precision atau presisi menunjukkan seberapa banyak dari semua prediksi positif yang benar-benar positif. Dengan kata lain, *precision* menggambarkan keandalan model dalam memberikan hasil positif yang benar:

$$\text{Precision} = \frac{TP}{TP+FP}. \tag{2.50}$$

Presisi yang tinggi menunjukkan bahwa sebagian besar sampel yang diklasifikasikan sebagai positif memang benar-benar positif. Dalam konteks medis, nilai *precision* yang tinggi sangat penting untuk menghindari kesalahan diagnosis berupa *false positive*, yang dapat menyebabkan pasien sehat menerima perlakuan atau pengobatan yang tidak perlu [45]. Sebaliknya, presisi rendah menunjukkan bahwa banyak prediksi positif yang salah, sehingga menurunkan keandalan sistem klasifikasi.

C Recall (*Sensitivity*)

Recall atau sensitivitas mengukur kemampuan model dalam mengidentifikasi seluruh kasus positif secara benar. Metrik ini menunjukkan proporsi pasien yang benar-benar memiliki penyakit yang berhasil dideteksi oleh model.

$$\text{Recall} = \frac{TP}{TP+FN} \tag{2.51}$$

di mana TP (*True Positive*) adalah jumlah data positif yang berhasil diprediksi dengan benar, sedangkan FN (*False Negative*) adalah jumlah data positif yang salah diprediksi sebagai negatif.

Nilai *recall* yang tinggi menunjukkan bahwa model mampu menangkap sebagian besar kasus positif. Dalam konteks medis, metrik ini sangat penting karena kesalahan dalam mendeteksi pasien positif (*false negative*) dapat berdampak serius terhadap keputusan klinis. Sebagai contoh, dalam diagnosis kanker, nilai *recall* yang rendah menunjukkan bahwa model gagal mendeteksi sejumlah pasien yang sebenarnya mengidap tumor, yang dapat berakibat fatal [39,45].

D F1-Score

F1-Score merupakan rata-rata harmonis antara *precision* dan *recall*, yang memberikan keseimbangan antara kemampuan mendeteksi kasus positif dan kemampuan menghindari hasil positif palsu:

$$\text{F1-Score} = \frac{2TP}{2TP + FP + FN} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{2.52}$$

F1-Score menjadi metrik penting dalam klasifikasi medis dengan data tidak seimbang karena menilai performa berdasarkan keseimbangan antara dua aspek utama: sensitivitas dan presisi. Nilai F1 yang tinggi menunjukkan bahwa model tidak hanya mendeteksi sebagian besar kasus positif, tetapi juga akurat dalam prediksi tersebut. Sebaliknya, nilai F1 yang rendah menandakan ketidakseimbangan antara banyaknya hasil positif palsu dan negatif palsu [45]. Pada penelitian berbasis radiomik dan klasifikasi *staging* tumor, F1-Score sering digunakan sebagai indikator utama karena mampu merepresentasikan performa model secara lebih stabil dibandingkan hanya menggunakan akurasi [39].

E ROC-AUC

Receiver Operating Characteristic (ROC) merupakan kurva yang digunakan untuk mengevaluasi kemampuan model klasifikasi dalam membedakan antara kelas positif dan kelas negatif pada berbagai nilai ambang keputusan (*threshold*). Kurva ROC dibangun dengan memplot *True Positive Rate* (TPR) terhadap *False Positive Rate* (FPR) pada berbagai nilai ambang probabilitas [46].

True Positive Rate (TPR) atau sensitivitas menunjukkan proporsi kasus positif yang berhasil dideteksi oleh model, sedangkan *False Positive Rate* (FPR) menunjukkan proporsi kasus negatif yang secara keliru diprediksi sebagai positif.

$$TPR = \frac{TP}{TP+FN}, \qquad FPR = \frac{FP}{FP+TN}. \tag{2.53}$$

Kurva ROC memberikan gambaran visual mengenai kemampuan model dalam membedakan kedua kelas pada berbagai nilai ambang. Semakin mendekati sudut kiri atas grafik ROC, semakin baik kemampuan model dalam melakukan klasifikasi.

Untuk mengukur performa model secara kuantitatif, digunakan nilai *Area Under the Curve* (AUC), yaitu luas area di bawah kurva ROC. Nilai AUC berkisar antara 0 hingga 1, di mana nilai yang lebih tinggi menunjukkan kemampuan diskriminasi model yang lebih baik.

$$AUC = \int_0^1 TPR(FPR)\, d(FPR). \tag{2.54}$$

Nilai AUC yang mendekati 1 menunjukkan bahwa model memiliki kemampuan yang sangat baik dalam membedakan antara kelas positif dan negatif. Sebaliknya, nilai AUC yang mendekati 0.5 menunjukkan bahwa performa model tidak lebih baik dibandingkan klasifikasi acak. Dalam penelitian berbasis radiomik dan machine learning medis, ROC-AUC sering digunakan sebagai metrik utama karena mampu mengevaluasi performa model secara lebih komprehensif pada berbagai nilai ambang keputusan [45,46].