BAB II

LANDASAN TEORI

2.1 Penelitian Terdahulu

Penelitian terdahulu memiliki peran penting sebagai dasar dan alasan penelitian. Pada subbab ini berisikan kajian literatur mengenai citra medis tulang menggunakan pendekatan *deep learning*, yaitu ResNet-50. Bagian ini memiliki tujuan untuk melihat bagaimana penggunaan model, serta tantangan yang terkait seperti keterbatasan data dan kebutuhan akan akurasi tinggi dalam diagnosis. Tabel 2.1 merangkum berbagai temuan penelitian terdahulu yang berkaitan dengan penggunaan *deep learning* dalam domain medis terkait tulang.

Penelitian komparasi model *deep learning* berbasis CNN seperti ResNet-50, VGGNet, dan AlexNet dilakukan pada klasifikasi fraktur citra *X-ray* [22]. Hasil penelitian menunjukkan bahwa model ResNet-50 memperoleh performa terbaik dengan nilai akurasi sebesar 78%, dibandingkan dengan VGGNet yang memperoleh akurasi 75% dan AlexNet sebesar 72%. Hasil tersebut menunjukkan bahwa arsitektur ResNet memiliki kemampuan yang lebih baik dalam mengekstraksi fitur citra medis dibandingkan arsitektur CNN konvensional lainnya. Penggunaan model *deep learning* diimplementasikan untuk melakukan klasifikasi fraktur tulang femur proksimal menggunakan *dataset* yang terdiri dari 1.347 gambar radiografi pasien. Model ResNet-50 menunjukkan hasil yang signifikan, mencapai F1-score 87% dan AUC 95% untuk klasifikasi tiga kelas, serta *Area Under* Curve (AUC) 98% untuk klasifikasi dua kelas, menunjukkan akurasi tinggi dalam identifikasi fraktur pada pasien, yang sangat penting untuk perawatan pasien lansia [28].

Penelitian lain memanfaatkan arsitektur ResNet-50 yang ditingkatkan dengan *Squeeze-and-Excitation Network* (SENet) untuk mendeteksi fraktur pergelangan kaki pada citra radiografi [29]. Model tersebut dilatih menggunakan dataset citra *X-ray* yang telah dikurasi dan dievaluasi menggunakan berbagai metrik seperti accuracy, precision, dan recall. Hasil penelitian menunjukkan bahwa model

adaptasi ResNet-50 dengan integrasi SENet mampu mencapai akurasi sebesar 93% dan nilai AUC sebesar 95%. Selain itu, visualisasi Grad-CAM digunakan untuk menginterpretasikan area citra yang berkontribusi terhadap keputusan model. Hasil ini menunjukkan bahwa integrasi *attention mechanism* pada arsitektur ResNet dapat meningkatkan kemampuan model dalam mengidentifikasi area fraktur secara lebih akurat pada citra medis.Pengujian algoritma *deep learning* juga diterapkan untuk mendeteksi fraktur leher femur pada *X-ray*, menggunakan *dataset* 4.189 gambar dari dua rumah sakit [30]. Dengan menggunakan ResNet yang dipadukan dengan CBAM dan *Gradient-weighted Class Activation Mapping,* atau disingkat dengan Grad-CAM, model ini memperoleh AUC 99% dan akurasi 98% pada *dataset* internal, serta AUC 97% dan akurasi 97% pada validasi eksternal, menunjukkan efektivitas *deep learning* dalam mendeteksi dan memvisualisasikan area fraktur dengan akurasi tinggi.

Sebuah penelitian mengusulkan pendekatan berbasis CNN dengan integrasi beberapa *attention blocks* untuk mendeteksi fraktur pada berbagai region tulang menggunakan citra *X-ray* [27]. Dataset yang digunakan mencakup beberapa area tulang seperti pinggul, lutut, serta beberapa daerah tulang lainnya untuk meningkatkan cakupan deteksi. Model CNN dikembangkan dengan memanfaatkan mekanisme *attention* pada CBAM untuk menyoroti fitur penting pada citra medis. Hasil penelitian menunjukkan bahwa model tersebut mampu mencapai akurasi hingga 96,7% dalam proses klasifikasi fraktur. Meskipun menunjukkan performa yang baik, pendekatan ini tidak menggunakan teknik *transfer learning* serta membutuhkan komputasi yang relatif besar dalam proses pelatihannya. Penelitian lain mengkaji penerapan model *deep learning* untuk menganalisis kerusakan pada jaringan tulang menggunakan *dataset* tulang kortikal dan trabekular dengan resolusi tinggi dari citra SR-microCT [19]. Model ini mengungkapkan tantangan dalam penggunaan data beresolusi tinggi pada proses pelatihan. Model terdiri dari CNN dan ResNet-50 dengan *fine-tuning*, yang menunjukkan hasil akurasi tinggi, dengan nilai akurasi 99% pada tulang trabekular dan 98,8% pada tulang kortikal, membuktikan potensi model *deep learning* dalam meningkatkan akurasi klasifikasi pada data medis yang kompleks.

Terdapat penelitian yang berfokus pada segmentasi fraktur pada gambar *X-ray* dengan menggunakan *dataset* FracAtlas yang berisi 4.497 gambar fraktur *X-ray* [20]. Penelitian tersebut mengemukakan bahwa segmentasi fraktur yang akurat sangat penting dalam proses pengobatan, dan tantangan utama yang dihadapi adalah perlunya komputasi yang lebih mendalam untuk mendapatkan informasi *X-ray* yang lebih jelas. Dalam penelitian ini, metode *deep learning* diterapkan dengan augmentasi data, menggunakan arsitektur *backbone* seperti ResNet, pada model yang dinamai DeepLabV3. Model yang dilatih selama 50 epoch berhasil mencapai nilai IoU (*Intersection over Union*) melebihi 93% dan nilai AUC mencapai 99%, menunjukkan efektivitas pendekatan *deep learning* dalam meningkatkan akurasi segmentasi fraktur. Selanjutnya, model ResNet diuji pada *dataset X-ray* pediatrik dari Kaggle dan GRAZPERDWRI-DX yang berjumlah 20.327 gambar untuk deteksi fraktur pergelangan tangan dengan pendekatan *self-supervised learning*. Model yang digunakan mencapai akurasi 94,10%, dengan AUROC 94,12% [31]. Berbeda dengan penggunaan model sebelumnya, terdapat perbandingan model *ensemble deep learning* untuk klasifikasi gambar *X-ray* tulang bahu, menggunakan *dataset* MURA [21]. Dengan menggunakan model seperti ResNet sebagai *backbone*, hasil evaluasi menunjukkan akurasi terbaik dengan nilai AUC 88,62% pada model ensemble.

Penelitian lain mengembangkan sistem pendukung keputusan berbasis ResNet50 untuk membedakan fraktur tulang belakang jinak dan ganas pada citra MRI, dengan tujuan membantu klinisi yang kurang berpengalaman [32]. Model dilatih pada 190 pasien (50 fraktur ganas dan 140 jinak), menggunakan satu segmen spinal abnormal per pasien dan memasukkan irisan T1W dan T2W ke dalam jaringan ResNet50. Kinerja dievaluasi dengan *tenfold cross-validation*, dan model mencapai akurasi 92% dalam membedakan fraktur jinak vs ganas, dengan sensitivitas meningkat dari 78% menjadi 94% dan spesifisitas dari 61% menjadi 91% bila dibandingkan dengan pembacaan residen tahun pertama. Selain itu, pengembangan model ResNet dan Faster R-CNN untuk deteksi fraktur pada tulang kaki bagian bawah, dengan menggunakan dataset 50 gambar X-ray dari Rumah Sakit Bahawal Victoria [33]. Dengan ini, model dapat mencapai *accuracy* 94% dan

precision 96%, menekankan efektivitas model dalam mendeteksi lokasi fraktur yang lebih tepat

Penelitian terdahulu menunjukkan keberhasilan dalam menerapkan berbagai pendekatan *deep learning* untuk deteksi dan klasifikasi fraktur tulang. Namun, masih terdapat beberapa keterbatasan atau *research gap* yang perlu diperhatikan, baik dari segi generalisasi model, fokus penelitian yang sempit pada jenis fraktur tertentu, serta *output* klasifikasi yang terbatas pada jenis *binary* atau 2 kelas prediksi. Terdapat penelitian yang melakukan penelitian komparasi antar model CNN dengan ResNet-50 sebagai model yang direkomendasikan untuk klasifikasi fraktur *X-ray*, namun penelitian hanya sebatas perbandingan model basis (*baseline*) [22]. Penelitian yang berfokus pada klasifikasi fraktur tulang femur proksimal menggunakan ResNet-50 dengan pendekatan *automatic region of interest* (ROI) ditemukan [28], namun ruang lingkupnya terbatas pada area tulang tertentu. Penelitian dengan objek citra tulang kortikal dan trabekular menggunakan pendekatan ResNet-50 dengan *fine-tuning* untuk klasifikasi kerusakan pada tulang tersebut [19], tetapi fokusnya terbatas pada data tulang yang dihasilkan menggunakan mesin SR-microCT beresolusi tinggi. Pada konteks validasi eksternal, penggunaan ResNet dengan *eXplainable* AI (XAI) Grad-CAM untuk mendeteksi fraktur leher femoral, mencapai akurasi tinggi (98,6%) pada *dataset* internal, meski menurun pada validasi eksternal (97,1%) [30], menunjukkan tantangan generalisasi model. Sebuah penelitian memilih *backbone* ResNet pada model usulannya, DeepLabV3, untuk segmentasi fraktur, menghasilkan IoU > 0,93 dan AUC 0,99, tetapi berbeda fokus dibandingkan klasifikasi fraktur [20]. Pendekatan *self-supervised learning* menggunakan arsitektur ResNet juga ditemukan dalam penelitian ini untuk mendeteksi fraktur pergelangan tangan, tetapi terbatas pada data pediatrik [31].

Implementasi arsitektur pada penelitian ini menggunakan ResNet-50 sebagai *baseline* model yang telah terbukti baik dibandingkan model lain dalam berbagai tugas klasifikasi citra medis untuk mendeteksi fraktur tulang yang melatarbelakangi penelitian ini [20][22]. Lalu, pendekatan *transfer learning* diterapkan pada model untuk meningkatkan efisiensi pembelajaran pada *dataset* baru, dengan

menggunakan metode *pretrain*, yakni memanfaatkan bobot yang telah dilatih sebelumnya pada domain serupa [19]. Selain optimalisasi performa melalui *transfer learning*, kebaruan penelitian ini juga terletak pada perluasan cakupan objek penelitian. Klasifikasi dilakukan secara *multi-class* menggunakan jenis fraktur yang lebih representatif pada ekstremitas atas dan bawah. Untuk mendukung cakupan yang luas tersebut, arsitektur model ditingkatkan dengan mengintegrasikan *attention mechanism*, yaitu CBAM. Integrasi CBAM membantu model lebih terfokus pada fitur-fitur penting dalam citra *X-ray* [26][27][30]. Setelah model berhasil dirancang, performa model akan diuji menggunakan berbagai metrik evaluasi, seperti *accuracy*, *precision*, *recall*, F1-*score* [22][28][30]. Model yang telah tervalidasi dengan performa terbaik kemudian akan diintegrasikan dalam perancangan sistem deteksi dini berbasis *web*. Penelitian ini diharapkan dapat memberikan kontribusi nyata dalam meningkatkan akurasi dan efisiensi diagnosis fraktur tulang, serta mengurangi risiko kesalahan interpretasi pada citra *X-ray* di sektor medis.

Tabel 2.1 Penelitian Terdahulu

Judul/Referensi	**Tahun**	**Masalah**	**Dataset**	**Arsitektur**	**Hasil**	**Kelebihan**	**Kekurangan**
CNN-based bone fracture Detection for medical imaging Using ResNet-50 [22]	2024	Perlunya pendekatan solutif untuk klasifikasi fraktur yang presisi dibandingkan diagnosis yang tidak selalu akurat	*Collected Clinical Radiographic Data*	ResNet-50, VGGNet, dan AlexNet	ResNet-50 (78%) menghasilkan akurasi yang lebih tinggi dibandingkan VGGNet (75%) dan AlexNet (72%)	Model ResNet-50 mengungguli model *deep learning lain* seperti VGGNet dan AlexNet dalam tugas	Masih ada ruang perkembangan bagi model dalam mendeteksi fraktur ekstrimitas atas dan bawah dengan optimisasi
Precise proximal femur fracture classification for interactive training and surgical planning [28]	2020	Mengklasifikasi fraktur tulang femur proksimal dengan akurat, yang sangat krusial dalam perawatan dan penanganan pasien terutama pada populasi lanjut usia dan dampak pada sosio-ekonomi yang terkait	*Collected Clinical Radiographic Data*	ResNet-50 dan AlexNet	F1-score 87% dan AUC 95% untuk klasifikasi tiga kelas (A, B, dan non-fraktur), serta AUC 98% untuk klasifikasi dua kelas (fraktur vs. non-fraktur).	Penggunaan ResNet-50 sebagai dasar model untuk klasifikasi fraktur menunjukkan performa yang sangat baik	Dataset terbatas pada jenis tipe fraktur tulang femur proksimal
Harnessing ResNet50 and SENet for enhanced ankle fracture identification [29]	2024	Deteksi fraktur pergelangan kaki dari citra radiografi untuk meningkatkan akurasi diagnosis	*Radiographic Ankle Fracture Dataset*	*Adapted* ResNet-50 + SENet *attention*	*Accuracy* 93%, AUC 95%, *Recall* 92%	Arsitektur ResNet-50 dengan mekanisme *attention* menghasilkan akurasi yang sangat baik	Fokus objek penelitian hanya terbatas pada pergelangan kaki
External Validation of Deep Learning Algorithm for Detecting	2021	Keterbatasan pencocokan umur serta jenis kelamin pasien dengan fraktur yang dialami, yang mempengaruhi hasil	*Collected from two tertiary hospitals*	ResNet-18 dengan CBAM (*Convolutional Block*	AUC 99%, *accuracy* 98%, *recall* 96%, *specificity* 99% pada *dataset* internal. Untuk validasi	Penggunaan CBAM pada arsitektur ResNet meningkatkan nilai akurasi	Data penelitian ini hanya mencakup tulang leher femoral

Judul/Referensi	Tahun	Masalah	Dataset	Arsitektur	Hasil	Kelebihan	Kekurangan
and Visualizing Femoral Neck Fracture Including Displaced and Non-displaced Fracture on Plain X-ray [30]		klinis, resolusi gambar yang buruk, serta hasil akurasi model validasi eksternal yang rendah ketika diterapkan pada rumah sakit lain.		*Attention Module*) dan Grad-CAM	eksternal, AUC 97%, *accuracy* 97%, *recall* 94%, *specificity* 98%.	hingga 98% dan penerapan xAI Grad-CAM sebagai visualisasi lanjutan	
Automatic Fracture Detection Convolutional Neural Network with Multiple Attention Blocks Using Multi-Region X-Ray Data [27]	2025	Keperluan akan deteksi fraktur *X-ray* yang akurat untuk menentukan penanganan yang sesuai	*Collected from various physiological areas*	Model CNN dengan CBAM	Menghasilkan akurasi 96,7% pada variasi anatomi tulang	Menunjukkan bahwa penambahan *attention* (CBAM) meningkatkan fokus fitur fraktur dan performa di banyak metrik.	Dataset terbatas, hanya satu sumber yang digeneralisasikan ke rumah sakit lain
Deep learning approach to assess damage mechanics of bone tissue [19]	2021	Keterbatasan data eksperimental yang sulit diperoleh, serta tantangan dalam menggunakan data citra beresolusi tinggi untuk pelatihan model *machine learning* yang optimal.	*SR-microCT Dataset*	CNN, ResNet-50 (dengan *fine-tuning* dan *transfer learning*)	ResNet-50 dengan fine-tuning mencapai akurasi 99% pada tulang trabekular dan 98,8% pada tulang kortikal. CNN dengan 4 *hidden layer* mencapai akurasi 98,3% pada tulang kortikal dan 90,1% pada tulang trabekular.	Membuktikan bahwa ResNet dengan *transfer learning* dapat diandalkan dalam konteks data medis yang terbatas	Fokus pada data microCT yang sedikit, serta bukan citra X-ray, sehingga langsungannya ke deteksi fraktur radiograf tidak otomatis
The impact of implementing backbone architectures on fracture segmentation in X-ray images [20]	2024	Keakuratan segmentasi fraktur yang mempengaruhi proses pengobatan. Perlunya komputasi yang	*FracAtlas Dataset*	DeepLabV3 (ResNet-50, ResNet-101, dan	Dengan pelatihan mencapai 50 *epochs*, model ini mendapatkan nilai IoU (*intersection*	Penggunaan *backbone* ResNet yang sangat kuat dalam segmentasi fraktur bercitra X-	Belum mengeksplorasi fokus lokal pada fitur fraktur lebih dalam, serta penggunaan

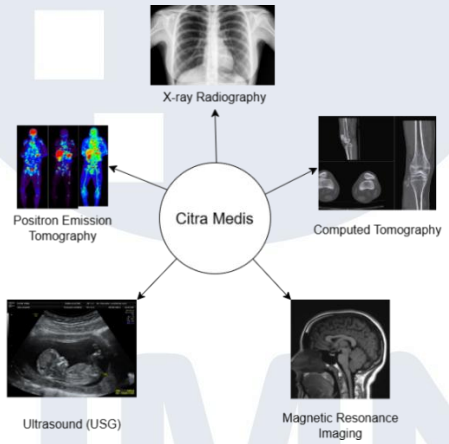
Judul/Referensi	Tahun	Masalah	Dataset	Arsitektur	Hasil	Kelebihan	Kekurangan
		memadai untuk mendapatkan informasi *x-ray* yang secara mendalam		MobileNetV3)	*over Union*) melebihi 93% dan nilai AUC mencapai 99%	ray. Penjelasan *trade-off* antara ResNet yang menggunguli MobileNet	dataset yang relatif kecil
Wrist fracture detection using self-supervised learning methodology [31]	2024	Diagnosis dan pengobatan patah tulang yang kurang tepat waktu, menyebabkan komplikasi lanjutan hingga kecacatan. Masalah ini diperburuk oleh meningkatnya populasi global dan meningkatnya kejadian patah tulang.	*GRAZPER DWRI-DX Dataset*	ResNet	Akurasi 94,1%, Spesifisitas 93,2%, AUROC 94,1%	ResNet sebagai arsitektur utama efektif dengan menggunakan *self-supervised learning* pada data pelabelan yang terbatas	Fokus penelitian hanya pada data *wrist* pediatrik, generalisasi ke tulang lain atau konteks tulang dewasa belum terbuktikan.
Classification of Shoulder X-ray Images with Deep Learning Ensemble Models [21]	2021	Kompleksitas dan variabilitas cedera pada diagnosis patah tulang bahu.	*MURA Dataset*	*Ensemble Learning Model* EL1 (*backbone*: ResNet), *Spinal Fully Connected* (Spinal FC, EL2)	Akurasi (EL1 dan EL2 berurutan): 0,8455, 0,8472, Cohen' Kappa: 0,6907, 0,6942, dan kelas dibawah *Receiver Operating Characteristic* (ROC) *Curve* (AUC): 0,8862, 0,8695.	Membandingkan keluarga ResNet dengan banyak *backbone* lain untuk klasifikasi X-ray bahu (DenseNet, VGGNet, MobileNet, Inception, dll) memberikan dasar kuat bahwa ResNet dapat menjadi *backbone* yang lebih unggul	Kompleksitas penelitian terlalu tinggi (*ensemble*), yang berpotensi memberikan performa yang sama dengan arsitektur optimasi yang lebih sederhana dan mudah dirancang seperti ResNet

Judul/Referensi	Tahun	Masalah	Dataset	Arsitektur	Hasil	Kelebihan	Kekurangan
A deep learning-based method for the diagnosis of vertebral fractures on spine MRI: retrospective training and validation of ResNet [32]	2022	Diagnosis fraktur tulang belakang serta membantu dokter dengan pengalaman terbatas	*Spine Fracture Dataset*	ResNet-50	Akurasi 92% untuk membedakan fraktur jinak vs ganas; AUC per-*slice* > 0,90 di semua run *cross-validation*	Menunjukkan bahwa ResNet-50 mampu mencapai akurasi diagnostik tinggi dan robust (AUC > 0,9) di data MRI fraktur vertebra, memperkuat pemilihan ResNet-50 sebagai backbone utama.	Model fokus pada satu segmen pada data MRI bukan X-ray, namun memenuhi konteks deteksi fraktur
Lower Leg Bone Fracture Detection and Classification Using Faster RCNN for X-rays Images [33]	2020	Tantangan klasifikasi fraktur tulang tungkai secara akurat yang masih dilakukan secara manual, dimana para ahli kesulitan mengidentifikasi lokasi fraktur secara efektif	Citra Radiologi dari Rumah Sakit Bahawal, Victoria	Faster R-CNN, VGG16, InceptionV2, ResNet	Akurasi model mencapai 94% dengan Precision sebesar 96%	Menggabungkan deteksi lokasi fraktur dan klasifikasi berkat ResNet *backbone*	Ukuran dataset tergolong sedikit untuk pelatihan model (*manual collecting*)

2.2 Teori yang Berkaitan

2.2.1 Citra Medis

Citra medis adalah proses menciptakan representasi visual dari struktur dan fungsi jaringan serta organ manusia, yang digunakan baik dalam pengaturan klinis maupun untuk mempelajari anatomi dan fisiologi secara mendalam [34]. Gambar 2.1 menunjukkan teknologi yang digunakan seperti *X-ray*, MRI, dan CT-scan, untuk diagnosis dan perencanaan pengobatan berdasarkan gambaran penyakit. Alat-alat ini menginterpretasikan anomali atau masalah pada pasien dalam bentuk gambar kepada dokter, memberikan wawasan yang dibutuhkan bagi pasien, terutama dalam situasi darurat. Kemampuan untuk memvisualisasikan masalah internal secara signifikan meningkatkan perawatan pasien, karena deteksi penyakit dan rencana perawatan yang lebih tepat dapat dilakukan lebih dini [35].

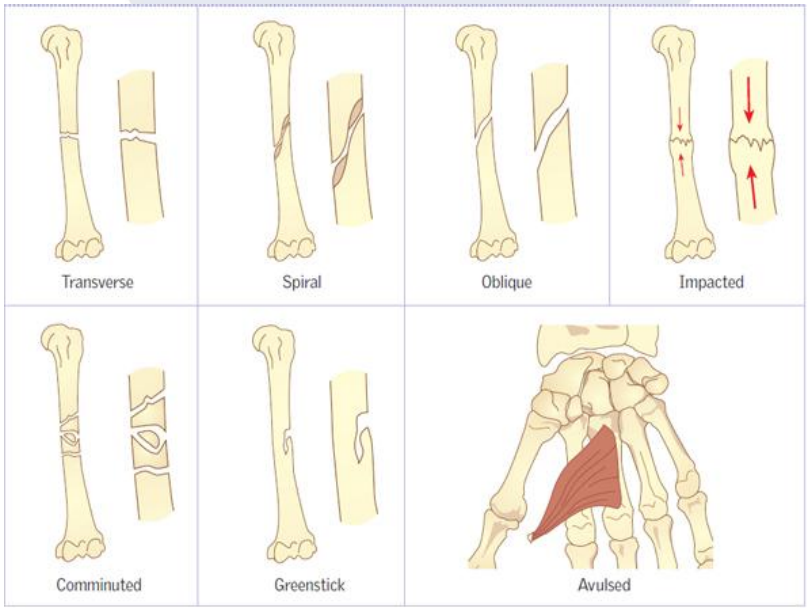


Gambar 2.1 Bidang Citra Medis
Sumber: [36]

2.2.2 Fraktur Tulang

Fraktur tulang adalah kondisi di mana tulang mengalami keretakan atau patah akibat cedera fisik [37], tekanan berlebih [38][39], atau penyakit seperti osteoporosis [40]. Menurut Whiteing dalam [39], terjadinya fraktur dapat menyebabkan pendarahan ke otot dan sendi, yang berujung pada hilangnya fungsi sementara area sekitar fraktur. Jenis fraktur dapat bervariasi, mulai dari fraktur ringan, sedang, hingga berat [41]. Fraktur halus (*hairline fracture*), merupakan kondisi tulang dengan retakan kecil tanpa memisahkan struktur utamanya. Selanjutnya, fraktur sederhana (*simple fracture*) merupakan kasus tipe menengah

di mana sebagian tulang retak tetapi belum sepenuhnya terpisah atau terbelah menjadi dua bagian. Tingkat fraktur berkeping (*comminuted fracture*) dikategorikan sebagai fraktur berat, di mana tulang dapat terbelah sepenuhnya menjadi beberapa bagian. Fraktur sering ditemukan pada ekstremitas atas dan bawah, khususnya pada tulang panjang seperti femur dan humerus, serta tulang pendek pada pergelangan tangan atau kaki. Fraktur dapat didiagnosis umumnya melalui teknologi citra seperti X-ray atau CT-scan, yang memberikan gambaran jelas mengenai lokasi dan tingkat kerusakan pada tulang [42]. Penanganan fraktur bervariasi tergantung pada jenis dan tingkat keparahannya, mulai dari penggunaan gips, terapi fisik, hingga tindakan pembedahan untuk pemasangan *fixator* atau kerangka logam dalam kasus fraktur yang lebih parah. Gambar 2.2 menunjukkan beberapa jenis fraktur pada tulang dengan contoh pada ekstremitas atas tulang lengan dan tangan.

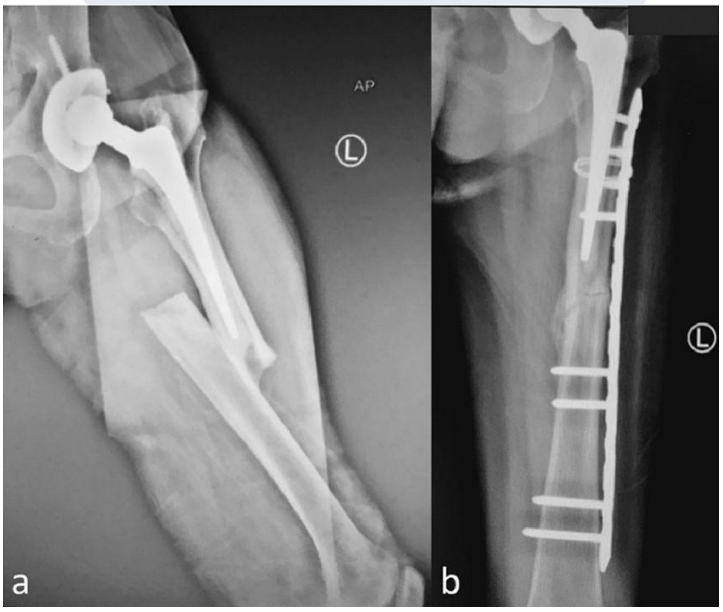


Gambar 2.2 Jenis Fraktur Tulang
Sumber: [43]

2.2.3 *Simple Fracture*

Fraktur simple, yang juga dikenal sebagai fraktur sederhana, merupakan kondisi patahan tulang di mana tulang retak atau patah namun kulit dan jaringan lunak di sekitarnya tetap utuh dan tidak mengalami perforasi [44]. Secara morfologis, fraktur sederhana ditandai oleh garis patahan yang relatif bersih dan

tulang hanya terbagi menjadi dua fragmen utama, sehingga secara radiologis dapat diamati sebagai diskontinuitas yang linear pada citra X-ray. Jika dikategorikan lebih lanjut dari subbab sebelumnya, fraktur ini mencakup beberapa subtipe berdasarkan arah patahan, yaitu fraktur transversal (*transverse fracture*) di mana tulang patah tegak lurus terhadap sumbu panjangnya, fraktur oblik (*oblique fracture*) dengan arah patahan yang miring, dan fraktur spiral (*spiral fracture*) yang memiliki pola melingkar mengikuti putaran tulang [45]. Identifikasi fraktur sederhana pada citra radiograf menjadi penting secara klinis karena penanganannya tergantung kondisi fraktur, dapat dilakukan imobilisasi, reduksi tertutup (*closed reduction)* atau memerlukan prosedur pembedahan terbuka yang disebut *open reduction internal fixation* (ORIF). Gambar 2.3 merepresentasikan tipe fraktur sederhana pada tulang femur atau paha dengan diagnosa fraktur subtipe *transversal fracture*, serta penanganannya menggunakan *plate and screw* pada operasi reduksi terbuka ORIF.



Gambar 2.3 Fraktur Sederhana pada a) Femur dan b) *Post-Operative* ORIF
Sumber: [46]

2.2.4 *Comminuted Fracture*

Fraktur kominutif atau berkeping merupakan jenis fraktur yang lebih kompleks dibandingkan fraktur sederhana [47], di mana tulang mengalami patahan menjadi tiga bagian atau lebih akibat benturan yang besar dan menyebabkan trauma

akut [48]. Fragmentasi yang terjadi pada fraktur kominutif menghasilkan pecahan-pecahan tulang yang dapat tersebar tidak beraturan di sekitar lokasi cedera, sehingga menjadikannya salah satu jenis fraktur yang paling sulit ditangani secara klinis. Secara radiologis, fraktur kominutif ditandai oleh adanya beberapa garis patahan yang saling berpotongan dan fragmen tulang yang dislokasi, berbeda dengan fraktur sederhana yang hanya menunjukkan diskontinuitas tunggal. Dalam proses identifikasi, fraktur kominutif sering menjadi tantangan tersendiri karena tekstur dan bentuk pola frakturnya yang tidak teratur dan sangat bervariasi antar pasien, sehingga diperlukan ketelitian ekstra bagi dokter atau tenaga medis dalam menangkap visual fraktur yang kompleks. Gambar 2.4 merupakan salah satu contoh fraktur kominutif tulang tibia atau tulang kering dari tampak depan dan samping.

Gambar 2.4 Fraktur Kominutif pada Tibia Secara a) Tampak Depan dan b) Tampak Samping
Sumber: [48]

2.2.5 *Data Validation*

Data validation adalah proses verifikasi yang bertujuan untuk memastikan bahwa data yang digunakan dalam pelatihan memiliki kualitas, konsistensi, dan representativitas yang memadai sebelum dimasukkan ke dalam *pipeline*

pembelajaran model [49]. Dalam konteks citra medis, kualitas data hasil validasi menjadi sangat krusial karena kesalahan dalam pelabelan atau kualitas gambar yang buruk dapat menghasilkan model yang bias dan tidak dapat diandalkan [50]. Proses validasi umumnya melibatkan dua tahap utama, yaitu validasi teknis dan validasi klinis. Validasi teknis mencakup pemeriksaan integritas berkas data, konsistensi format, dan distribusi kelas label untuk memastikan tidak ada ketidakseimbangan yang ekstrem. Sementara itu, validasi klinis dilakukan oleh pakar bertujuan untuk memverifikasi bahwa setiap label yang diberikan pada citra sesuai dengan diagnosis yang tepat. Keterlibatan validator ahli secara langsung memengaruhi keandalan (*reliability*) dataset, yang pada akhirnya menentukan seberapa jauh hasil pelatihan model dapat dipercaya dan digeneralisasi ke data baru.

2.2.6 *Data Splitting*

Data splitting merupakan proses membagi keseluruhan kumpulan data (*dataset*) menjadi beberapa subset dengan tujuan untuk menghindari *over-fitting* dalam siklus pengembangan model *machine learning* [51]. Secara umum, dataset dibagi menjadi tiga bagian: *training set* yang digunakan untuk melatih parameter model, *validation set* yang digunakan untuk pemilihan konfigurasi dan penyetelan hiperparameter selama eksperimen, serta *test set* sebagai evaluasi akhir yang tetap murni hingga model final ditetapkan, sehingga pengukuran kinerja lebih merefleksikan kemampuan generalisasi pada data baru yang belum pernah dilihat sebelumnya [52]. Pembagian dataset ke dalam data pelatihan dan data pengujian, misalnya dengan rasio sekitar 80 persen untuk *data training* dan 20 persen untuk *data test*, merupakan pendekatan yang umum diterapkan dalam pengembangan model karena mampu menyediakan *data training* yang cukup besar sekaligus data evaluasi yang representatif [53]. Namun, tidak ada komposisi pembagian yang bersifat mutlak untuk semua kasus. Penentuan rasio tersebut sangat bergantung pada beberapa aspek, seperti jumlah data yang tersedia, variasi dan distribusi data, serta tingkat kompleksitas model yang digunakan.

Pada masalah klasifikasi dengan distribusi kelas yang tidak seimbang, pendekatan pemisahan terstratifikasi (*stratified splitting*) dapat digunakan untuk

memastikan proporsi label tetap konsisten di setiap subset, sehingga evaluasi model tidak bias terhadap kelas mayoritas. [54]. Untuk estimasi yang lebih stabil dibandingkan satu kali pemisahan, teknik *k-fold cross-validation* dapat diterapkan, di mana data latih dibagi menjadi lipatan "k" dan proses pelatihan-validasi diulang sebanyak "k" kali secara bergantian, di mana pada setiap iterasi satu lipatan digunakan sebagai data validasi dan sisanya sebagai data pelatihan. Kinerja model selanjutnya dihitung dengan merata-ratakan hasil evaluasi dari seluruh iterasi tersebut [54]. Dalam skema ini, *test set* tetap dipertahankan sepenuhnya terpisah dari proses cross-validation untuk mencegah kebocoran informasi dari proses penyetelan *hyperparameter* [53].

2.2.7 *Data Augmentation*

Augmentasi data adalah teknik yang bertujuan untuk memperluas ukuran dan variasi data dalam dataset pelatihan dengan menerapkan serangkaian transformasi sintetis pada data yang sudah ada, tanpa mengubah label kelas yang sudah ada. Dalam domain citra medis, teknik ini sangat krusial dikarenakan keterbatasan data medis, seperti contohnya data kondisi penyakit yang jarang terjadi [55]. Hal ini dapat mempengaruhi keseimbangan komposisi kelas dan menjadi hambatan utama dalam pengembangan model *deep learning*. Dengan menerapkan *augmentation*, model tidak hanya mendapat lebih banyak sampel pelatihan secara kuantitas, tetapi juga memperkenalkan model dengan variasi kondisi citra yang lebih luas sehingga meningkatkan kemampuan generalisasinya terhadap data baru dan mengurangi resiko *overfitting* [55].

Secara umum, teknik augmentasi data pada citra terbagi ke dalam beberapa kategori transformasi. Transformasi geometri mencakup rotasi, (*flipping horizontal* maupun *vertical*), *cropping*, pergeseran (*translation*), dan perubahan skala (*zoom*). Transformasi fotometrik meliputi penyesuaian kecerahan, kontras, saturasi, dan penambahan *noise* Gaussian. Terdapat pula teknik yang lebih canggih seperti *mixup*, *cutout*, dan *cutmix* yang menggabungkan atau menutup sebagian citra untuk memaksa model memanfaatkan seluruh *region* citra secara lebih holistik [56]. Dalam konteks radiografi tulang untuk deteksi fraktur, augmentasi geometri seperti

rotasi dan *flipping* sangat relevan karena citra sinar-X dapat diambil dari berbagai sudut proyeksi dan orientasi pasien yang berbeda-beda. Data augmentation juga berperan penting dalam mengatasi ketidakseimbangan distribusi kelas, karena teknik ini dapat diterapkan secara selektif pada kelas minoritas untuk menyeimbangkan jumlah sampel pelatihan dan dengan demikian mencegah model menjadi bias terhadap kelas yang lebih dominan [55][56]. Dalam penelitian ini, augmentation diterapkan pada masing-masing kumpulan data latih hingga uji, sehingga setiap *epoch* model akan mendapatkan variasi citra yang berbeda-beda dan tidak terpaku pada satu versi data yang tetap.

2.2.8 *User Acceptance Testing*

User Acceptance Testing (UAT) adalah tahap pengujian yang dilakukan oleh pengguna akhir untuk memastikan bahwa perangkat lunak telah memenuhi kebutuhan dan persyaratan yang disepakati, baik dari sisi fungsional maupun operasional [57][58]. UAT berfungsi menghasilkan bukti bahwa sistem yang dikembangkan dapat diterima oleh pengguna karena telah berjalan sesuai kebutuhan bisnis dan spesifikasi yang telah disusun sebelumnya. Proses UAT umumnya meliputi beberapa tahapan utama, yaitu perancangan, pelaksanaan pengujian, dan analisis hasil [57][59]. Pada tahap perancangan, sebuah rencana UAT disusun dengan mencakup tujuan, prosedur pengujian, serta aspek-aspek yang akan diuji seperti fungsionalitas, kinerja, tampilan antarmuka, efisiensi, dan produktivitas pengguna. Lalu, tahap pelaksanaan dilakukan dengan cara pengguna menjalankan sistem sesuai prosedur yang telah dirancang sebelumnya, kemudian menilai pengalaman penggunaan sistem berdasarkan skenario atau daftar pertanyaan yang telah disiapkan. hasil pengujian dihitung dan diinterpretasikan, misalnya dalam bentuk persentase tingkat penerimaan atau kategori kualitas sistem (seperti dari kategori "*passed*" atau "*failed*") untuk menentukan apakah sistem sudah layak diterapkan atau masih memerlukan perbaikan. Salah satu metode yang banyak digunakan untuk mengukur tingkat penerimaan pengguna dalam UAT adalah kuesioner menggunakan skala Likert [59][60]. Pendekatan ini menggunakan penilaian yang terukur terhadap berbagai aspek sistem melalui pernyataan yang

dijawab dengan menggunakan skala Likert 1-5, mulai dari “Tidak Setuju” hingga “Sangat Setuju” dan kemudian dapat dihitung persentase dari keseluruhan penilaian pengguna terhadap sistem.

2.3 *Framework* dan Algoritma

2.3.1 *Cross-Industry Standard Process for Data Mining*

Cross-Industry Standard Process for Data Mining (CRISP-DM) adalah kerangka kerja proses yang dirancang secara sistematis untuk pelaksanaan proyek *data mining* dan *machine learning* dari tahap pemahaman masalah hingga penerapan solusi [61][62]. CRISP-DM mencakup seluruh siklus transformasi data mentah menjadi pengetahuan yang bermakna dalam pengambilan keputusan. Proses sistematis ini mengintegrasikan enam tahapan krusial, yang meliputi *business understanding*, *data understanding*, *data preparation*, *modeling*, *evaluation*, hingga tahap *deployment*. Struktur dari alur kerja yang menyeluruh diilustrasikan secara mendetail pada Gambar 2.5, yang menyajikan keterkaitan antar tahapan utama guna menjamin validitas hasil dalam setiap fase pengembangan:

1. ***Business Understanding***

   Business understanding adalah langkah awal dalam kerangka kerja CRISP-DM yang berfokus pada pemahaman tujuan dan kebutuhan proyek dari sudut pandang domain atau bidang yang diteliti. Proses ini bertujuan untuk memastikan bahwa permasalahan yang akan diselesaikan telah didefinisikan secara jelas, terukur, dan selaras dengan kebutuhan pengguna akhir. Pada penelitian medis, misalnya, tahap ini mencakup identifikasi kebutuhan klinis seperti peningkatan akurasi deteksi fraktur tulang yang selama ini masih bergantung pada interpretasi manual radiolog. Tahap ini sangat penting karena kejelasan tujuan akan menentukan arah seluruh proses analisis berikutnya.

2. ***Data Understanding***

   Setelah tujuan ditetapkan, langkah selanjutnya adalah *data understanding*. Tahapan ini mencakup pengumpulan data awal, eksplorasi karakteristik

dataset, serta identifikasi potensi masalah seperti ketidakseimbangan kelas (*class imbalance*), *value* yang hilang, atau inkonsistensi format. Selain itu, analisis distribusi data dilakukan untuk memastikan bahwa dataset yang tersedia cukup representatif terhadap permasalahan yang ingin diselesaikan. Pemahaman mendalam terhadap data pada tahap ini bertujuan untuk memberikan landasan yang kuat dalam pengambilan keputusan terkait persiapan dan pemodelan data pada tahapan selanjutnya.

3. Data Preparation

Tahapan ini bertujuan untuk mempersiapkan data mentah menjadi format yang bersih, terstruktur, dan siap digunakan oleh arsitektur *deep learning*. Proses ini mencakup pembersihan data untuk menghilangkan nilai-nilai yang tidak valid, duplikasi, atau inkonsistensi yang mungkin terdapat dalam dataset, serta penanganan *noise* agar informasi yang tidak relevan tidak memengaruhi hasil analisis. Selanjutnya, data ditransformasikan ke dalam representasi yang sesuai, misalnya citra medis seperti X-ray diubah ukurannya, dinormalisasi, dan diperkaya melalui augmentasi untuk memastikan data yang dihasilkan terstruktur, informatif , serta efisien untuk dianalisis oleh model.

4. Modeling

Modeling adalah inti dari tahapan CRISP-DM, di mana arsitektur dirancang untuk menemukan pola atau hubungan tersembunyi dalam data menjadi sebuah model yang matang. Pada tahap ini, algoritma *deep learning* digunakan untuk menganalisis data dan mengidentifikasi pola kompleks yang sulit ditemukan secara manual. Tahapan ini mencakup pemilihan arsitektur model, konfigurasi *hyperparameter*, dan proses pelatihan menggunakan data yang telah disiapkan. Dalam konteks deteksi fraktur tulang, misalnya, *modeling* membantu mengidentifikasi area pada citra X-ray yang menunjukkan kemungkinan adanya fraktur melalui pembelajaran representasi fitur visual secara hierarkis.

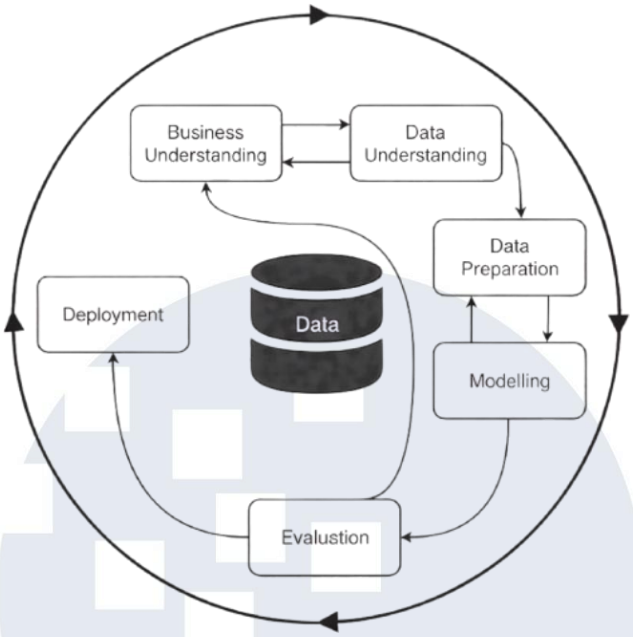
5. Evaluation

Setelah model dibangun dan dilatih, tahap berikutnya adalah mengevaluasi hasil analisis untuk memastikan bahwa model yang dihasilkan valid, relevan, dan memenuhi tujuan yang telah ditetapkan pada tahap *business understanding*. Hasil dievaluasi menggunakan metrik yang sesuai, seperti *accuracy*, *precision*, *recall*, dan F1-*score*, tergantung pada konteks analisis. Selain itu, interpretasi hasil dilakukan untuk menyajikan temuan dalam format yang mudah dipahami oleh pengguna akhir, seperti dokter atau peneliti.

6. ***Deployment***

   Tahap *deployment* merupakan tahapan akhir yang membedakan CRISP-DM dari kerangka kerja lainnya, di mana model yang telah dievaluasi dan dinyatakan layak diterapkan ke dalam lingkungan nyata sesuai kebutuhan pengguna akhir. Pada konteks penelitian ini, *deployment* mencakup penyajian model dalam *user interface* atau sistem yang dapat diakses oleh pengguna akhir untuk mendukung proses dari sistem yang telah dibuat. Tahap ini juga mencakup pengujian performa model secara *real-time* untuk memastikan bahwa model tetap memberikan hasil yang dapat diandalkan ketika dihadapkan pada data baru di luar kondisi pelatihan, serta perencanaan pembaruan model apabila diperlukan di masa mendatang.

Kerangka kerja CRISP-DM memiliki fleksibilitas dalam penggunaan berbagai metode yang dirancang untuk menangani data dalam skala besar dengan efisiensi tinggi. CRISP-DM menawarkan kemampuan generalisasi yang luas karena bersifat *domain-agnostic* dan dapat diterapkan pada berbagai bidang industri maupun penelitian tanpa kehilangan efektivitasnya. Kerangka kerja ini juga sekaligus berfungsi dalam menekankan pentingnya keterkaitan antara tujuan bisnis dan hasil teknis sebagai satu kesatuan proses yang utuh.



Gambar 2.5 *Framework* CRISP-DM
Sumber: [61]

2.3.2 *Deep Learning*

Deep learning merupakan evolusi dari arsitektur *artificial neural network* (ANN), yang melahirkan istilah "deep" dimana merujuk pada jumlah lapisan tersembunyi dalam ANN [13]. Evolusi ini terjadi karena ANN memiliki keterbatasan dalam menangani kompleksitas data, sehingga menyebabkan performa yang kurang optimal dalam mempelajari fitur-fitur yang lebih dalam atau non-linear. Hal ini mendorong pengembangan *deep learning*, yang memiliki struktur lebih mendalam yang memungkinkan model untuk mempelajari fitur-fitur yang lebih kompleks dan abstrak pada berbagai jenis data [21].

Selain itu, kemampuan *deep learning* untuk mengolah data dalam jumlah besar menjadi salah satu keunggulannya dibandingkan metode pembelajaran lainnya. Dengan memanfaatkan algoritma berbasis *backpropagation* [63][64], model *deep learning* dapat mengoptimalkan bobot pada setiap lapisan dengan efisiensi tinggi, menjadikannya sangat efektif dalam tugas seperti pengenalan gambar, pengolahan teks, dan analisis suara. Aplikasi ini semakin relevan seiring dengan kemajuan teknologi komputasi dan ketersediaan data yang melimpah,

membuka peluang model *deep learning* untuk digunakan dalam berbagai domain, termasuk citra medis hingga analisis prediktif.

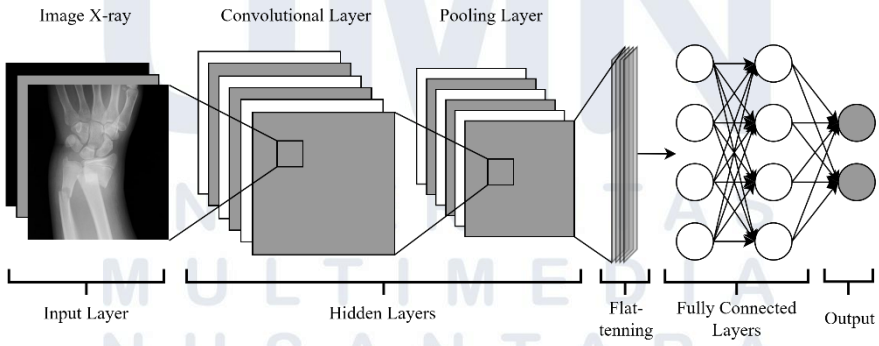
2.3.3 *Transfer Learning*

Transfer learning merupakan sebuah teknik dalam kecerdasan buatan yang memanfaatkan model yang telah dilatih pada *dataset* besar untuk diterapkan pada domain serupa dengan data yang lebih terbatas [65][66]. Strategi ini secara mendasar bertujuan untuk mentransfer basis pengetahuan, seperti ekstraksi fitur dan pola bobot yang telah diperoleh dari domain asal (*source domain*) ke dalam domain baru yang menjadi sasaran penelitian (*target domain*). Hal tersebut secara signifikan dapat mengurangi ketergantungan model terhadap kebutuhan jumlah data pelatihan yang sangat besar pada domain yang ditargetkan, sehingga proses pengembangan menjadi lebih efisien [67]. Dengan pendekatan ini, *transfer learning* mampu mendukung model dalam menyelesaikan tugas-tugas baru yang masih terkait dengan domain asal meskipun data pada domain target terbatas atau bahkan tidak tersedia [68].

Dalam implementasinya, *transfer learning* seringkali digunakan pada arsitektur *deep learning*, di mana model *pretrained* seperti ResNet, VGG, dan Inception telah dikembangkan menggunakan *dataset* besar seperti ImageNet. Model-model ini dapat diadaptasi untuk berbagai tugas spesifik dengan menyesuaikan lapisan-lapisan akhir pada model atau melatih ulang lapisan tertentu (*fine-tuning*). Keunggulan teknik ini terletak pada efisiensinya, karena mengurangi waktu pelatihan dan kebutuhan sumber daya komputasi, dibandingkan dengan membangun model dari awal [69]. Selain itu, teknik ini dikenal dapat memperkokoh model, sehingga sangat berguna dalam kasus nyata di mana pengumpulan data baru memerlukan waktu serta biaya yang tidak sedikit. *Transfer learning* juga telah terbukti lebih efisien dibandingkan metode pelatihan model dari awal, terutama pada domain yang membutuhkan waktu pelatihan yang cepat dan hasil yang akurat [69][70].

2.3.4 *Convolutional Neural Network*

Convolutional Neural Network (CNN) adalah salah satu arsitektur *deep learning* yang dirancang khusus untuk menganalisis data berbentuk *grid* seperti citra. CNN secara luas digunakan dalam klasifikasi gambar karena kemampuannya untuk secara otomatis mengekstraksi fitur penting seperti pola, tepi, hingga hierarki fitur dari data [15]. Selain itu, CNN memiliki keunggulan dalam mempertahankan hubungan spasial antar piksel melalui operasi konvolusi, sehingga informasi lokal pada citra dapat dipelajari secara lebih efektif dan efisien dibandingkan metode tradisional yang sering kali mengabaikan posisi relatif antar piksel. Proses ini memfasilitasi model untuk mengenali, mengekstraksi, sekaligus menggabungkan representasi fitur secara bertahap, mulai dari tingkat rendah hingga tingkat yang lebih tinggi. Tahapan tersebut dimulai dari identifikasi pola sederhana seperti tepi, sudut, dan tekstur, yang kemudian berkembang menjadi pemahaman terhadap struktur kompleks seperti bentuk objek spesifik dan karakteristik pola global secara menyeluruh yang terdapat di dalam dataset. Melalui mekanisme hierarkis yang terintegrasi ini, CNN mampu membangun pemahaman yang jauh lebih mendalam terhadap isi citra, sehingga secara signifikan meningkatkan akurasi dalam berbagai tugas visi komputer seperti klasifikasi, deteksi, maupun segmentasi objek. Struktur CNN secara fundamental terdiri dari beberapa lapisan utama seperti lapisan *convolution*, *pooling*, dan *fully connected* yang bekerja secara berurutan, sebagaimana diilustrasikan Gambar 2.6.



Gambar 2.6 Struktur CNN
Sumber: [71]

Lapisan pertama adalah *convolutional layer*, yang berfungsi untuk mengekstraksi fitur-fitur penting dari data *input*, seperti tepi, pola, atau tekstur. Layer ini menggunakan *filter* atau *kernel* yang bergerak melintasi *input*, melakukan operasi matriks untuk menghasilkan *feature map*. Setiap *feature map* mewakili intensitas pola tertentu yang terdeteksi oleh *filter*, sementara koneksi spasial lokal memungkinkan fokus pada detail kecil tanpa terlalu banyak parameter. *Zero padding* sering digunakan untuk menjaga dimensi output tetap sama dengan input, sehingga fitur di tepi gambar tidak terabaikan. *Layer* ini membantu jaringan mengenali detail kecil secara efisien tanpa perlu terlalu banyak parameter. Berikut adalah persamaan dari tugas konvolusi dengan *zero padding* [72]:

$$S_{i,j} = (I * K)_{i,j} = \sum m \sum n \, I_{i,j} \cdot K_{i-m,j-n} \quad (2.1)$$

Rumus 2.1 Persamaan *Convolutional Task*
Sumber: [72]

Berikut adalah notasi serta penjelasan persamaan (2.1):

1. *Feature Map* $S_{i,j}$ yang merupakan elemen pada posisi (i,j) setelah operasi konvolusi dilakukan. Matriks ini berisi fitur penting yang dihasilkan dari proses ekstraksi oleh kernel.
2. *Input Matrix* I, seperti gambar yang diolah. Matrik pada posisi (i,j) mewakili intensitas piksel dalam gambar *input*.
3. *Kernel* K, atau disebut juga dengan filter, yang berfungsi untuk mendeteksi fitur spesifik pada *input*. Elemen $K_{i-m,j-n}$ merujuk pada nilai *kernel* saat dilakukan *sliding* terhadap *input*.
4. *Convolution Operation* $(I * K)_{i,j}$, yaitu proses pengkalian elemen $I * K$ pada posisi (i,j).

Setelah melewati *convolutional layer*, data diteruskan ke lapisan *pooling* untuk mengurangi dimensi data sambil tetap mempertahankan informasi penting. Lapisan ini bertujuan untuk mereduksi jumlah *parameter* dan komputasi dalam jaringan, sehingga membuat proses pelatihan lebih efisien dan mencegah *overfitting*. *Pooling* dilakukan dengan cara memilih nilai maksimal (*max pooling*)

atau rata-rata (*average pooling*) dari area tertentu dalam *feature map*. Metode ini juga membantu jaringan menjadi lebih tahan terhadap pergeseran kecil pada *data input*, seperti perbedaan posisi atau orientasi fitur pada citra. Dengan begitu, *pooling* menjadi langkah penting dalam memperkuat kemampuan generalisasi model. Di bawah ini merupakan rumus kedua metode *pooling*:

Max Pooling:

$$pool = down\,(max(y_{i,j}))_{,i,j} \in p \tag{2.2}$$

Rumus 2.2 Persamaan *Max Pooling*

Sumber: [72]

Berikut adalah notasi serta penjelasan persamaan (2.2):

1. *Area Pooling* $y_{i,j}$ yang merupakan elemen matriks pada posisi (i,j) *region* persegi (2×2, 3×3) yang didefinisikan oleh parameter *pooling*.
2. $max(y_{i,j})$, yaitu nilai maksimum dari elemen-elemen dalam area pooling. Proses ini memilih elemen terbesar dalam *region* tersebut untuk representasi keluaran.
3. $down$, merupakan operator yang menggabungkan hasil pooling menjadi keluaran dengan dimensi lebih kecil (*downsampling*).
4. p, yang merupakan wilayah *pooling* atau *patch* yang didefinisikan pada *input*. Pooling dilakukan secara terpisah untuk setiap wilayah p dalam *input*.

Average Pooling:

$$M_j = tanh\,(\beta \sum_{N \times N} M_n\, x_n + b) \tag{2.3}$$

Rumus 2.3 Persamaan *Average Pooling*

Sumber: [72]

Berikut adalah notasi serta penjelasan persamaan (2.3):

1. M_j yang merupakan elemen keluaran dari *average pooling*.

2. $\sum_{N\times N} M_n\, x_n + b$, yang merupakan penjumlahan semua elemen dalam wilayah pooling berukuran $N \times N$. Setiap elemen diberi bobot M_n sebelum dijumlahkan dengan *input* x_n.
3. β, yaitu faktor skala yang mengontrol pengaruh dari rata-rata terhadap hasil *pooling*.
4. p, yang merupakan bias yang ditambahkan untuk meningkatkan fleksibilitas dalam penyesuaian hasil *pooling*.
5. $tanh$, merupakan fungsi aktivasi hiperbolik tangent yang digunakan untuk membatasi hasil *pooling* dalam rentang nilai tertentu (-1 dan 1).

Untuk memperkenalkan non-linearitas dan menangkap pola yang lebih kompleks, CNN dapat menggunakan fungsi aktivasi seperti ReLU (*Rectified Linear Unit*), Sigmoid, dan Tanh. Pada tahap akhir, *fully connected layers* menggabungkan semua fitur yang diekstraksi untuk menghasilkan prediksi akhir, seperti klasifikasi objek atau nilai regresi. Tabel 2.2 adalah *pseudocode* dari proses model CNN [73].

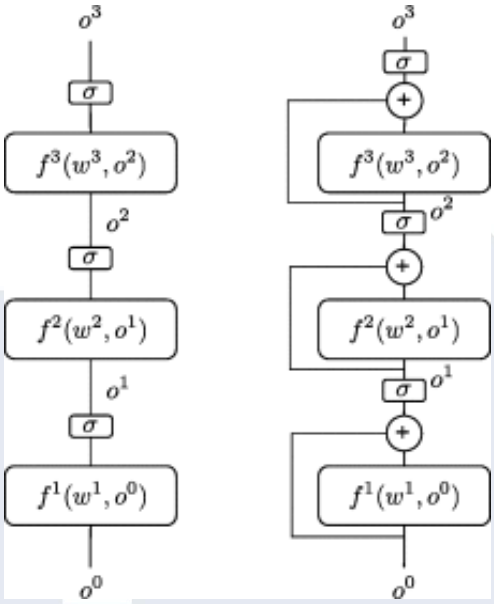
Tabel 2.2 *Pseudocode* Model CNN

(1) Menginisialisasi *filter* $\boldsymbol{K}$ dengan ukuran $\boldsymbol{k \times k}$ dan *bias* $\boldsymbol{I}$ dengan nilai nol
(2) Melakukan operasi *forward pass*, atau *convolution* dengan dimensi gambar $\boldsymbol{i_d \times j_d}$ dan menyimpan hasil konvolusi dengan $\boldsymbol{(i_d - k + 1) \times (j_d - k + 1)}$ (4)
(3) Mulai iterasi untuk posisi $\boldsymbol{i, j}$
(4) Melakukan *backpropagation* dengan *input* dari *array error* dengan dimensi $\boldsymbol{(i_d - k + 1) \times (j_d - k + 1)}$, dan diisi oleh nilai nol untuk masing-masing dimensi $\boldsymbol{i_d \times j_d}$ (5)
(5) Mulai iterasi posisi $\boldsymbol{i_d \times j_d}$ dalam *array error*
(6) Perbarui *filter* dan *bias* menggunakan optimasi

2.3.5 *Residual Network*

Residual Network, atau yang dikenal sebagai ResNet, merupakan salah satu arsitektur berbasis CNN yang secara spesifik dirancang untuk tugas kompleks yang berhubungan dengan *computer vision* [74]. Arsitektur ini juga dikembangkan untuk mengatasi masalah *vanishing gradient*, sebuah kendala teknis yang sering kali muncul pada struktur *deep neural network* ketika jumlah lapisan semakin bertambah dalam [75]. Masalah ini pada dasarnya menyebabkan nilai gradien menjadi sangat kecil saat proses *backpropagation*, sehingga pembaruan *parameter*

terhenti pada lapisan-lapisan awal dan secara drastis menurunkan performa serta akurasi model secara keseluruhan.



Gambar 2.7 Rangkaian Kerja *Residual Block*
Sumber: [76]

Gambar 2.7 menunjukkan cara kerja *residual blocks* pada model ResNet, dimana jaringan dapat melewati beberapa lapisan melalui mekanisme *skip connections*. Mekanisme ini memungkinkan informasi dari lapisan sebelumnya langsung diteruskan ke lapisan berikutnya tanpa dimodifikasi, sehingga jaringan lebih fokus mempelajari perbedaan antara *input* dan *target output*, daripada mempelajari *target output* secara langsung. Persamaan dari *residual block* adalah sebagai berikut [76]:

$$o^n = f^n(w^n, o^{n-1}) = \sigma(o^{n-1} + f^n(w^n, o^{n-1})) \quad (2.4)$$

Rumus 2.4 Persamaan *Residual Block*

Berikut adalah notasi serta penjelasan persamaan (2.4):

1. o^{n-1} yang merupakan keluaran dari lapisan ke $n-1$.
2. $f^n(w^n, o^{n-1})$, yang merupakan fungsi transformasi *non-linear* yang diterapkan pada o^{n-1}, tergantung pada parameter w^n.
3. σ, yaitu fungsi aktivasi *non-linear* seperti ReLU atau Tanh.

Berdasarkan persamaan (7), proses dalam *residual block* dimulai dengan inisialisasi *input*, yang merupakan output dari lapisan sebelumnya, dilambangkan sebagai o^{n-1}. *Input* ini kemudian diproses melalui fungsi transformasi non-linear $f^n(w^n, o^{n-1})$ di mana w^n adalah parameter yang mewakili bobot yang dipelajari selama pelatihan. Fungsi transformasi ini biasanya mencakup operasi seperti *convolution*, normalisasi *batch*, dan fungsi aktivasi seperti ReLU atau Tanh. Hasil dari transformasi non-linear ini kemudian dijumlahkan dengan *input* awal o^{n-1}, membentuk proses yang dikenal sebagai penjumlahan *residual*. Penjumlahan ini memungkinkan jaringan untuk menggabungkan informasi awal dengan informasi yang dihasilkan oleh transformasi, dengan tujuan agar jaringan bisa lebih efisien dalam mempelajari perbedaan (*residual*) antara *input* dan *output*. Tabel 2.3 merupakan bentuk *pseudo-code* dari algoritma *residual network* [77].

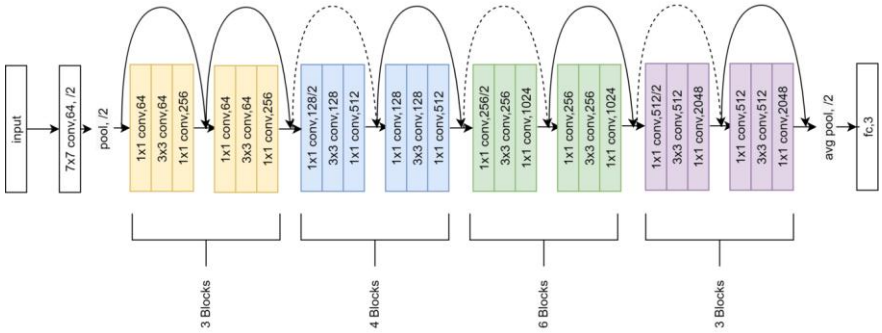
Tabel 2.3 *Pseudocode* Algoritma *Residual Network*

(1) Menginisialisasi *input weight* $\boldsymbol{w^n}$ dan jumlah lapisan jaringan $\boldsymbol{n}$
(2) Melakukan konvolusi pertama dengan $\boldsymbol{x}$ *filter*, *kernel*, dan *stride*.
(3) Menerapkan normalisasi *batch* dan fungsi aktivasi.
(4) Mendefinisikan lapisan *max pooling* dengan fungsi aktivasi
(5) Mulai iterasi (*epoch*) untuk setiap *stage*, dengan menggunakan *residual block* dari $\boldsymbol{o^1}$ hingga $\boldsymbol{o^n}$ (7)
(6) Menerapkan *convolutional block* pada *stage tertentu*
(7) Akhiri iterasi jika setiap *stage* memiliki *identity* dan *convolution block* dengan mengulangi langkah 6 hingga 8 atau jumlah *epoch* melebihi batas
(8) Mendefinisikan lapisan *average pooling* dengan fungsi aktivasi
(9) Mendefinisikan lapisan *fully-connected* dan *output* dengan fungsi aktivasi
(10) Bangun model dengan *input* dan *output* yang telah didefinisikan
(11) Kompilasi nilai *loss* model dengan fungsi $\boldsymbol{y - o^n}$, yang menghitung perbedaan *output* prediksi dengan target
(12) Melakukan *backpropagation* untuk pembaruan *weight* untuk setiap lapisan $\boldsymbol{n}$ dengan fungsi optimasi
(13) Kembalikan model yang telah dikompilasi

Pendekatan *residual* ini sangat berguna karena memungkinkan pembelajaran untuk melakukan identitas *mapping* secara langsung jika fungsi transformasi tidak memberikan kontribusi yang signifikan, yang mengurangi masalah *vanishing gradient* dan mempermudah optimasi. Dengan demikian, *residual block* dalam ResNet tidak hanya memungkinkan penggabungan informasi yang lebih efisien tetapi juga meningkatkan kemampuan jaringan untuk menangani representasi yang lebih kompleks

2.3.5.1 ResNet-50

ResNet-50 adalah salah satu varian dari model ResNet yang memiliki total 50 lapisan [78]. Arsitektur ini dirancang untuk mengatasi tantangan pada tugas-tugas *computer vision* yang lebih kompleks, seperti klasifikasi gambar dengan ratusan kategori atau segmentasi citra beresolusi tinggi. ResNet-50 mempertahankan prinsip dasar arsitektur *residual* dengan menggunakan *residual blocks* yang dilengkapi *skip connections* [78][79], tetapi memperluas jumlah lapisan untuk meningkatkan kapasitas jaringan dalam mempelajari representasi data yang lebih mendetail.



Gambar 2.8 Arsitektur ResNet-50
Sumber: [80]

ResNet-50 diciptakan untuk memenuhi kebutuhan model yang lebih dalam tanpa mengorbankan stabilitas pelatihan atau efisiensi komputasi. Berdasarkan Gambar 2.8, struktur dari arsitektur ResNet-50 terdiri dari 16 *residual blocks* yang dipecah menjadi empat tahap dengan konfigurasi jumlah lapisan berbeda, yaitu 3, 4, 6, dan 3 blok pada masing-masing tahap [80]. Setiap blok mengandung *convolutional layers*, *batch normalization*, dan fungsi aktivasi, yang dipadukan dengan koneksi *residual*. Selain itu, ResNet-50 menggunakan teknik *bottleneck* pada *residual blocks*, yaitu penggunaan *convolutional layers* berukuran 1x1 di awal dan akhir blok untuk mengurangi dimensi fitur sebelum dilakukan operasi konvolusi utama. Pendekatan ini memungkinkan peningkatan efisiensi komputasi tanpa mengorbankan performa model. Keunggulan ResNet-50 terletak pada

kemampuannya untuk menangani representasi fitur yang lebih kompleks, menjadikannya pilihan ideal untuk berbagai aplikasi *computer vision*, seperti deteksi objek, segmentasi semantik, dan klasifikasi gambar skala besar.

2.3.6 *Loss Function*

Fungsi kerugian adalah komponen matematis inti dalam pelatihan jaringan saraf dalam (*deep neural network*) yang memiliki fungsi untuk mengukur seberapa besar perbedaan antara *output* prediksi model dengan nilai label yang sesungguhnya. Nilai yang dihasilkan oleh *loss function* inilah yang menjadi sinyal kesalahan (*error signal*) yang kemudian dipropagasi kembali melalui jaringan lewat algoritma *backpropagation* agar bobot (*weights*) model dapat diperbarui ke arah yang memperkecil kesalahan tersebut [81]. Secara prinsip, semakin kecil nilai *loss*, semakin baik model dalam memetakan *data input* ke label yang benar; namun nilai *loss* yang terlalu kecil pada data latih tanpa diimbangi performa yang baik pada data validasi merupakan indikasi *overfitting*.

Pemilihan jenis loss function yang tepat sangat bergantung pada sifat dari tugas yang dihadapi. Untuk tugas regresi, fungsi seperti *Mean Squared Error* (MSE) atau *Mean Absolute Error* (MAE) umumnya digunakan. Sementara untuk tugas klasifikasi, keluarga fungsi *cross-entropy* menjadi pilihan yang paling dominan karena secara teoritis konsisten (*Bayes-consistent*) sebagai fungsi pengganti (*surrogate*) dari *zero-one loss* yang tidak dapat didiferensiasikan [82]. Dalam konteks penelitian ini yang melibatkan tugas klasifikasi multi-kelas fraktur tulang, categorical cross-entropy loss menjadi pilihan yang paling relevan dan banyak digunakan.

2.3.7 *Cross-Entropy Loss*

Cross-entropy loss merupakan salah satu fungsi kerugian yang populer digunakan dalam pelatihan model klasifikasi. Secara konseptual, fungsi ini mengukur perbedaan antara dua distribusi probabilitas: distribusi probabilitas prediksi yang dihasilkan oleh model dan distribusi probabilitas label aktual. Semakin besar perbedaan antara keduanya, semakin tinggi nilai *loss* yang

dihasilkan, dan semakin kuat sinyal yang diberikan kepada model untuk memperbaiki parameternya [82].

Dalam konteks klasifikasi *multi-class*, *cross-entropy loss* bekerja bersama dengan fungsi aktivasi *softmax* pada lapisan *output* jaringan. Fungsi *softmax* terlebih dahulu mengonversi nilai logit mentah dari lapisan terakhir menjadi distribusi probabilitas yang bernilai antara 0 dan 1 serta berjumlah 1 di seluruh kelas, yang kemudian menjadi *input* bagi *cross-entropy loss*. Secara matematis, untuk klasifikasi C kelas, *categorical cross-entropy loss* untuk satu sampel didefinisikan sebagai:

$$L = -\sum_{c=1}^{C} y_c \cdot \log(\hat{y}_c) \tag{2.5}$$

Rumus 2.5 Persamaan *Cross-Entropy Loss*

Berikut adalah notasi serta keterangan persamaan (2.5):

1. L, yang merupakan keluaran nilai *cross-entropy loss*
2. C, yang merupakan jumlah kelas yang ada dalam klasifikasi
3. y_c, yaitu label aktual (1 pada sampel termasuk kelas C).
4. $\hat{y}_c$, merupakan probabilitas prediksi model untuk kelas C.

2.3.8 *Adam Optimizer*

Adam Optimizer adalah algoritma optimasi yang dirancang untuk menyelesaikan masalah optimasi dengan parameter yang bersifat *high-dimensional* dan fungsi objektif yang bersifat stokastik. Algoritma ini memanfaatkan estimasi adaptif dari momen pertama (rata-rata) dan momen kedua (varian), yang membuat pembaruan parameter yang lebih stabil dan efisien [83]. Dalam praktiknya, Adam menggunakan rata-rata bergerak secara eksponensial untuk menghitung estimasi gradien dan gradien kuadrat, yang kemudian diperbaiki dengan *bias correction* agar akurat sejak awal pelatihan. *Hyperparameter* utama Adam, yaitu *learning rate* (α), eksponensial *decay rate* untuk momen pertama (β_1), dan momen kedua (β_2), masing-masing memiliki nilai awal: 0,001, 0,9, dan 0,999 secara berurutan.

Keunggulan utama Adam terletak pada kemampuannya untuk menyesuaikan *learning rate* secara otomatis selama proses pelatihan data. Selain itu, Adam menggabungkan kelebihan metode AdaGrad, yang baik dalam menangani data yang sedikit, dan RMSProp, yang cocok untuk data yang terus berubah [84]. Dengan sifatnya yang fleksibel, Adam bekerja baik meski pengaturan awalnya tidak sempurna, sehingga sering digunakan dalam pelatihan model *deep learning* yang membutuhkan waktu lama atau bekerja dengan *dataset* besar. Algoritma ini juga lebih stabil dan cepat dalam menemukan solusi terbaik dibandingkan metode optimasi lainnya seperti *Stochastic Gradient Descent* (SGD).

2.3.9 *Attention Mechanism*

Attention mechanism adalah sebuah pendekatan dalam *deep learning* yang terinspirasi dari cara kerja sistem visual manusia, di mana otak secara selektif memusatkan perhatian pada bagian-bagian yang paling relevan dari suatu pemandangan visual tanpa perlu memproses seluruh informasi secara setara [85]. Dalam *neural network*, *attention mechanism* dapat dipandang sebagai proses penyesuaian bobot secara dinamis berdasarkan fitur dari data *input*, dimana bagian yang dianggap lebih informatif mendapatkan bobot perhatian yang lebih besar, sementara bagian yang kurang relevan diminimalkan. Mekanisme ini telah terbukti memberikan peningkatan performa yang signifikan pada berbagai tugas *computer vision*, termasuk klasifikasi citra, deteksi objek, segmentasi semantik, dan pemahaman video [85].

Jika dijabarkan, *attention mechanism* dalam *computer vision* dapat diklasifikasikan ke dalam beberapa kategori utama: (1) *channel attention*, yang memodelkan ketergantungan antar *channel* fitur untuk menentukan *channel* mana yang paling informatif; (2) *spatial attention*, yang memperhitungkan hubungan antar posisi *spatial* dalam peta fitur untuk menentukan lokasi mana yang perlu diprioritaskan; (3) *self-attention* atau *non-local attention*, yang menangkap ketergantungan jarak jauh (*long-range dependencies*) antar semua posisi dalam peta fitur; serta (4) *branch attention*, yang mengalokasikan perhatian di antara jalur-jalur arsitektur yang berbeda [85][86]. Masing-masing kategori ini memiliki

karakteristik komputasi dan kelebihan tersendiri tergantung pada sifat tugas dan data yang dihadapi. Sebuah tinjauan komprehensif pada [86] menegaskan bahwa *attention mechanism* merupakan komponen yang dapat diterapkan secara luas di berbagai arsitektur dan domain tugas *deep learning*.

Dalam konteks deteksi fraktur tulang, *attention mechanism* memiliki relevansi klinis yang tinggi. Citra *X-ray* fraktur seringkali mengandung latar belakang yang kompleks (jaringan lunak, artefak radiografi) yang dapat mengganggu representasi fitur konvolusi. Dengan *attention mechanism*, model dapat belajar untuk secara otomatis meminimalisir pengaruh *region* latar belakang tersebut dan memfokuskan ekstraksi fitur pada area diskontinuitas yang menjadi penanda utama fraktur, sehingga menghasilkan representasi yang lebih bermakna secara diagnostik.

2.3.10 *Convolutional Block Attention Module*

Convolutional Block Attention Module (CBAM) adalah salah satu modul ringan dari *attention mechanism* yang dirancang untuk jaringan konvolusi *feed-forward*, yang secara efektif meningkatkan kualitas representasi fitur melalui proses *feature enhanement* [87]. CBAM beroperasi dengan cara menyimpulkan *attention maps* secara berurutan melalui dua dimensi yang berbeda: dimensi *channel* dan dimensi *spatial*, kemudian mengalikan *attention map* yang dihasilkan tersebut dengan *feature map input* untuk proses penyempurnaan fitur secara adaptif [88].

2.3.10.1 *Channel Attention Module*

Pada tahap *channel attention*, model menentukan pentingnya setiap *channel* fitur yang ada. Proses ini dilakukan dengan menggunakan operasi *global average pooling* (AvgPool) dan *global max pooling* (MaxPool), yang kemudian diproses melalui *multi-layer perceptron* (MLP). Secara matematis, *channel attention* dapat dinyatakan sebagai [88]:

$$M_c(F) = \sigma(W_1(W_0 \cdot AvgPool(F)) + W_1(W_0 \cdot MaxPool(F))) \quad (2.6)$$

Rumus 2.6 Persamaan *Channel Attention*
Sumber: [88]

Berikut adalah notasi serta penjelasan persamaan (2.6):

1. σ, adalah fungsi aktivasi *sigmoid*
2. W_n, yang merupakan bobot MLP ke n
3. F, yaitu adalah *feature map*.

2.3.10.2 *Spatial Attention Module*

Setelah kontribusi setiap *channel* yang relevan ditemukan oleh *channel module*, proses dilanjutkan menuju proses *spatial attention*. Pada tahap ini, model berfokus untuk mengidentifikasi dan menentukan lokasi-lokasi penting (*spatial feature*) di dalam citra. Proses ini dilakukan dengan menggabungkan hasil *pooling* dan menerapkan *convolution*:

$$M_s(\grave{F}) = \sigma(f^{7\times7}[AvgPool(\grave{F});\ MaxPool(\grave{F})]) \qquad (2.7)$$

Rumus 2.7 Persamaan *Spatial Attention*
Sumber: [88]

Berikut adalah notasi serta penjelasan persamaan (2.7):

1. $\grave{F}$, yang merupakan *feature map* yang sudah disempurnakan
2. $f^{7\times7}$, adalah operasi konvolusi dengan ukuran kernel 7×7

Validasi ekstensif pada dataset ImageNet-1K, MS COCO, dan VOC 2007 membuktikan bahwa CBAM secara konsisten meningkatkan performa klasifikasi dan deteksi pada berbagai arsitektur model [88]. Karena CBAM dirancang sebagai modul yang ringan dan bersifat umum, modul ini dapat diintegrasikan secara mulus ke dalam arsitektur seperti ResNet dengan *overhead parameter* yang minimal, serta dapat dilatih secara *end-to-end*. Dalam pengaplikasian untuk deteksi fraktur tulang, integrasi CBAM pada arsitektur ResNet-50 terbukti mampu meningkatkan kemampuan model dalam memfokuskan perhatian pada area fraktur [89].

2.3.11 *Evaluation Metrics*

Dalam penelitian berbasis *deep learning*, metrik evaluasi digunakan untuk mengukur performa model dalam menyelesaikan tugas tertentu. Beberapa metrik yang sering digunakan meliputi *accuracy*, *precision*, *recall*, dan F1-*score*. Dalam

konteks medis, metrik *recall* menjadi sangat krusial karena mampu meminimalkan risiko fraktur yang terlewat dengan mendeteksi seluruh kondisi patah tulang yang sebenarnya terjadi pada pasien. Melalui kombinasi seluruh nilai ini, peneliti dapat mengevaluasi tingkat keberhasilan generalisasi model secara menyeluruh.

2.3.11.1 *Accuracy*

Accuracy merupakan metrik evaluasi yang mengukur proporsi prediksi model yang benar terhadap keseluruhan prediksi yang dibuat. Semakin tinggi nilai akurasi, semakin baik model tersebut dalam melakukan prediksi yang benar, dengan nilai yang berkisar antara 0 hingga 1. Persamaann untuk menghitung akurasi adalah sebagai berikut [90]:

$$Accuracy = \frac{TruePositive + TrueNegative}{TruePositive + TrueNegative + FalsePositive + FalseNegative} \quad (2.8)$$

Rumus 2.8 Persamaan *Accuracy*
Sumber: [90]

Pada persamaan (2.8), *True Positive* merupakan prediksi positif yang benar dan *True Negative* adalah prediksi negatif yang benar. Kedua metrik tersebut dibagi dengan *False Positive* dimana prediksi positif dikatakan salah dan *False Negative* adalah prediksi negative yang salah. Metrik ini menunjukkan kualitas model dalam memberikan prediksi benar dengan tidak memperhitungkan distribusi kelas [91].

2.3.11.2 *Precision*

Precision adalah metrik yang mengukur bagian dari prediksi positif yang benar-benar merupakan positif asli dari seluruh prediksi yang diklasifikasikan sebagai positif oleh model, dengan kisaran nilai 0 hingga 1. Secara matematis, *precision* didefinisikan sebagai [92][93]

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive} \quad (2.9)$$

Rumus 2.9 Persamaan *Precision*
Sumber: [92][93]

Persamaan (2.9) menunjukkan persamaan dengan kesimpulan nilai *precision* yang tinggi berarti model jarang menghasilkan prediksi positif yang salah (*false positive*). Dalam konteks deteksi fraktur, *precision* yang rendah berarti model sering mengidentifikasi citra tulang normal sebagai fraktur, yang dapat menyebabkan kekeliruan yang tidak perlu pada pasien dan penanganan yang sia-sia.

2.3.11.3 *Recall (Sensitivity)*

Recall, atau yang juga disebut sebagai sensitivitas adalah metrik evaluasi yang menghitung rasio kasus positif yang diklasifikasikan oleh model dari keseluruhan kasus yang benar-benar positif yang memang positif (*true positive*). Rumus dari metrik evaluasi *recall* adalah [92][93]:

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative} \tag{2.10}$$

Rumus 2.10 Persamaan *Recall*
Sumber: [92][93]

Berdasarkan persamaan (2.10), semakin besar nilai *recall* (mendekati 1) maka semakin baik juga model dalam mendeteksi kasus positif. Jika diaplikasikan dalam konteks deteksi fraktur, *recall* menjadi metrik yang sangat penting dikarenakan nilai *recall* yang terlalu rendah menunjukkan bahwa model gagal mendeteksi fraktur sesungguhnya (*false negative*) yang berpotensi memperparah kondisi pasien karena keterlambatan diagnosis.

2.3.11.4 F1-*score*

Metrik evaluasi F1-*score* merupakan rata-rata gabungan dari metrik *precision* dan *recall*, yang memberikan satu nilai harmonik atau sebagai penyeimbang nilai kedua metrik. Metrik ini sangat berguna ketika terdapat ketidakseimbangan distribusi kelas, karena F1-*score* akan memberikan penalti terhadap model yang mengoptimalkan salah satu metrik dengan mengorbankan yang lainnya. Berikut adalah persamaan matematis dari metrik F1-*score* [92]

$$F1 - score = 2 \times \frac{(Precision \times Recall)}{(Precision + Recall)} \quad (2.11)$$

Rumus 2.11 Persamaan F1-*score*
Sumber: [92]

Hasil *output* dari persamaan matematis F1-*score* seperti persamaan yang tertulis pada (2.11), akan menghasilkan angka dengan kisaran 0 hingga 1. Nilai tersebut mencerminkan kondisi ideal dari ketika *precision* dan *recall* sama-sama memiliki performa yang maksimal dan seimbang. Sebaliknya, apabila terdapat selisih yang besar pada salah satu metrik, maka perhitungan *harmonic mean* pada F1-*score* secara otomatis akan cenderung turun mendekati nilai terendahnya.

2.4 *Tools* Penelitian

2.4.1 Python

Python merupakan salah satu bahasa pemrograman yang terkenal karena fleksibilitas dan sintaksnya yang mudah dipahami bahkan oleh pemula sekalipun. Python mendukung berbagai paradigma pemrograman, seperti pemrograman berorientasi objek, prosedural, dan fungsional, sehingga bahasa ini cocok untuk berbagai jenis proyek mulai dari pengembangan web hingga kecerdasan buatan. Python juga dapat diakses dengan mudah karena sifatnya yang *open-source* [94]. Popularitas Python terus meningkat secara konsisten hingga menempati posisi pertama pada indeks PYPL (PopularitY of Programming Language) dan TIOBE pada tahun 2022, mencerminkan adopsinya yang luar biasa luas di kalangan komunitas ilmuwan data, peneliti, dan insinyur perangkat lunak di seluruh dunia [95].

Dalam konteks penelitian berbasis computer vision, Python telah menjadi bahasa pilihan utama karena ekosistem *library* ilmiahnya. Lingkungan implementasi *deep learning* pada umumnya memanfaatkan *library* Python seperti TensorFlow, PyTorch, dan Keras, yang menyediakan dukungan komprehensif untuk berbagai tugas analisis citra medis. Di samping ketiganya, NumPy berperan fundamental sebagai *backbone* dari komputasi numerik. Hampir seluruh *framework deep learning* menggunakan NumPy [95]. Matplotlib menyediakan fungsionalitas

visualisasi data yang esensial untuk menganalisis distribusi dataset, memantau kurva pelatihan, dan menampilkan hasil prediksi model. Selain itu, *library* seperti OpenCV dan Pillow memungkinkan manipulasi dan pra-pemrosesan citra secara efisien langsung dalam ekosistem Python [96]. Dalam penelitian ini, Python digunakan sebagai bahasa utama untuk seluruh tahapan implementasi, mulai dari pra-pemrosesan data, pembangunan dan pelatihan arsitektur ResNet-50 yang diintegrasikan dengan CBAM, hingga evaluasi dan visualisasi hasil model.

2.4.2 Visual Studio Code

Visual Studio Code (VSCode) adalah *software* penyunting kode sumber (*source code editor*) berbobot ringan namun berfitur lengkap yang dikembangkan oleh Microsoft dan dirilis pertama kali pada tahun 2015. VSCode dirancang untuk mendukung proses pengembangan perangkat lunak secara efisien melalui antarmuka yang intuitif dan dapat dikustomisasi secara luas [97]. VSCode menghadirkan fitur-fitur esensial yang sangat mendukung produktivitas pengembang, di antaranya: syntax highlighting untuk lebih dari ratusan bahasa pemrograman, IntelliSense berbasis kecerdasan buatan untuk penyelesaian kode (*code completion*) yang kontekstual, fasilitas debugging terintegrasi dengan *breakpoint* dan inspeksi variabel, serta integrasi kendali versi Git secara bawaan [97].

Salah satu keunggulan terbesar VSCode terletak pada ekosistem ekstensinya yang sangat kaya. Melalui *extensions marketplace*, pengguna dapat menambahkan fungsionalitas secara modular sesuai kebutuhan proyek, mulai dari dukungan bahasa pemrograman tambahan (Python, Java, C++, R), *linter* dan *formatter* untuk memastikan konsistensi gaya penulisan kode, hingga integrasi dengan *framework deep learning* seperti TensorFlow dan PyTorch secara langsung di dalam *editor*. Ketersediaan lintas sistem operasi menjadikan VSCode sebagai pilihan lingkungan pengembangan yang fleksibel, portabel, dan banyak diadopsi pada beragam konteks mulai dari pengembangan *web*, aplikasi *mobile*, hingga riset sains data dan kecerdasan buatan [98].

Dalam penelitian ini, VSCode digunakan sebagai lingkungan pengembangan utama untuk penulisan, pengujian, dan pengelolaan kode implementasi model deteksi fraktur berbasis ResNet-50 dan CBAM. Ekstensi Python dan Jupyter yang tersedia di VSCode memfasilitasi alur kerja penelitian secara terintegrasi, mulai dari eksplorasi data awal, pembangunan arsitektur model, proses pelatihan, hingga analisis dan visualisasi hasil evaluasi.

2.4.3 Streamlit

Streamlit adalah *framework* berbasis Python yang dirancang untuk mempermudah pengembangan aplikasi web interaktif, khususnya untuk keperluan visualisasi data dan penerapan model *machine learning* [99]. Berbeda dengan *framework* pengembangan web konvensional seperti Flask atau Django yang mengharuskan pemisahan antara logika *back-end* dan *front-end*, Streamlit dirancang pada seluruh komponen aplikasi dibangun dalam satu skrip Python yang sama, sehingga proses pengembangan prototipe menjadi jauh lebih mudah dan efisien. Streamlit bekerja dengan cara mengeksekusi ulang seluruh skrip setiap kali pengguna melakukan interaksi pada halaman *web*, yang dikenal sebagai mekanisme *reactive execution*, sehingga pembaruan tampilan berlangsung secara otomatis dan real-time tanpa memerlukan penanganan event yang kompleks.

Dalam konteks penerapan model *deep learning*, Streamlit menyediakan berbagai komponen antarmuka bawaan (*built-in widgets*) seperti *file uploader*, *slider*, *button*, dan area tampilan gambar yang dapat diintegrasikan langsung dengan pustaka *machine learning* populer seperti PyTorch dan TensorFlow. Hal ini menjadikan Streamlit sebagai pilihan yang tepat untuk membangun antarmuka *deployment* model klasifikasi citra medis, di mana pengguna dapat mengunggah citra X-ray secara langsung melalui antarmuka *web* dan memperoleh hasil prediksi beserta visualisasi interpretasi model secara langsung.