

**PENGEMBANGAN DETEKSI POSISI JATUH PADA LANSIA  
MENGGUNAKAN *GRAPH CONVOLUTIONAL NETWORK*  
BERBASIS *POSE LANDMARK***

**TUGAS AKHIR**

Diajukan sebagai syarat menyelesaikan jenjang strata Satu (S-1) di  
Program Studi Teknik Informatika, Fakultas Teknologi Industri,  
Institut Teknologi Sumatera

**Oleh:**

**Leonardo Alfontus Mende Sirait**

**121140028**



**PROGRAM STUDI TEKNIK INFORMATIKA  
FAKULTAS TEKNOLOGI INDUSTRI  
INSTITUT TEKNOLOGI SUMATERA  
LAMPUNG SELATAN  
2025**

## KATA PENGANTAR

Puji syukur kehadirat Tuhan Yang Maha Esa atas limpahan rahmat, karunia, serta petunjuk-Nya sehingga penyusunan tugas akhir ini telah terselesaikan dengan baik. Dalam penyusunan tugas akhir ini penulis telah banyak mendapatkan arahan, bantuan, serta dukungan dari berbagai pihak. Oleh karena itu pada kesempatan ini penulis mengucapkan terima kasih kepada:

1. Orang tua, abang dan adek yang telah mendukung penulis dari awal Perkuliahan hingga Penyelesaian penelitian tugas akhir ini.
2. Penulis sendiri yang terus berjuang untuk menyelesaikan penelitian tugas akhir ini hingga akhir, menghadapi segala hambatan dan kebuntuan dalam pengerjaan penelitian ini.
3. Bapak Martin Clinton Tosima Manullang, Ph.D. selaku Dosen Pembimbing utama atas ide topik penelitian, waktu, tenaga dan perhatian yang luar biasa, serta masukan yang telah disumbangsihkan kepada penulis untuk menyelesaikan penelitian ini.
4. Ibu Leslie Anggraini, S.Kom., M.Cs. Selaku Dosen Pembimbing Pendamping atas waktu, tenaga dan perhatian yang mendalam pada pengerjaan laporan penelitian bersama penulis hingga penyelesaian tugas akhir ini.
5. Teman-teman penulis yang membantu selama masa perkuliahan dan mendorong penulis untuk tidak menyerah dalam melakukan penelitian tugas akhir ini yang tidak bisa disebutkan satu persatu.

Akhir kata penulis berharap semoga tugas akhir ini dapat memberikan manfaat bagi kita semua. Penulis menyadari bahwa tugas akhir ini tidak luput dari kekurangan dan kelemahan, dan penulis terbuka untuk menerima saran, kritik, dan masukan.

## ABSTRAK

### PENGEMBANGAN DETEKSI POSISI JATUH PADA LANSIA MENGGUNAKAN *GRAPH CONVOLUTIONAL NETWORK* BERBASIS *POSE LANDMARK*

Leonardo Alfontus Mende Sirait

Kejadian jatuh pada lansia merupakan masalah serius yang memerlukan perhatian khusus dalam upaya pencegahan cedera. Menurut WHO, sekitar 37,3 juta kejadian jatuh memerlukan perhatian medis. Diperlukan Model Deteksi yang akurat dan cepat terhadap kejadian jatuh dan tidak jatuh untuk meningkatkan kualitas hidup lansia. Penelitian sebelumnya mengembangkan model deteksi jatuh berbasis visi dengan CNN dan LSTM, namun terdapat kelemahan pada kebutuhan daya komputasi yang tinggi untuk pengolahan citra. Penelitian ini bertujuan untuk mengembangkan Model deteksi berbasis *Graph Convolutional Network* (GCN) menggunakan data *pose landmark* yang secara teknik lebih ringan dari citra. Pada penelitian ini, telah dilakukan pengembangan model GCN dengan dua konfigurasi hyperparameter, konfigurasi default dan konfigurasi hasil studi ablasi. Hasil eksperimen menunjukkan bahwa konfigurasi studi ablasi menghasilkan peningkatan kinerja yang signifikan dibandingkan dengan konfigurasi default. Secara keseluruhan, akurasi meningkat dari 0,665 menjadi 0,683 (peningkatan 0,018), presisi meningkat dari 0,694 menjadi 0,718 (peningkatan 0,024), dan F1 meningkat dari 0,638 menjadi 0,687 (peningkatan 0,049). Peningkatan terbesar terlihat pada recall, yang meningkat dari 0,590 menjadi 0,683 (peningkatan 0,093), menunjukkan bahwa model dengan konfigurasi studi ablasi lebih baik dalam mendeteksi kejadian jatuh dan mengurangi kejadian terlewati. Hasil ini mengindikasikan bahwa konfigurasi studi ablasi dapat meningkatkan kinerja model GCN dalam mendeteksi kejadian jatuh pada lansia, yang memiliki potensi aplikasi yang besar dalam pengembangan sistem pemantauan kesehatan lansia yang berfokus pada jatuh dan tidak jatuh di masa depan.

**Kata Kunci:** Lansia, Jatuh, Deep Learning, Graph Convolution Network, Mediapipe, Pose Landmark, studi ablasi

## ABSTRACT

### DEVELOPMENT OF FALL POSITION DETECTION IN THE ELDERLY USING A POSE LANDMARK-BASED CONVOLUTIONAL GRAPH NETWORK

Leonardo Alfontus Mende Sirait

Falls among the elderly are a serious problem that requires special attention in injury prevention efforts. According to the WHO, approximately 37.3 million falls require medical attention. An accurate and rapid detection model for falls and non-falls is needed to improve the quality of life of the elderly. Previous studies have developed vision-based fall detection models using CNN and LSTM, but there are weaknesses in the high computing power required for image processing. This study aims to develop a detection model based on Graph Convolutional Network (GCN) using landmark pose data, which is technically lighter than images. In this study, a GCN model was developed with two hyperparameter configurations, a default configuration and an ablation study configuration. The results of the experiment show that the ablation study configuration produces a significant improvement in performance compared to the default configuration. Overall, accuracy increased from 0.665 to 0.683 (an increase of 0.018), precision increased from 0.694 to 0.718 (an increase of 0.024), and F increased from 0.638 to 0.687 (an increase of 0.049). The largest improvement was seen in recall, which increased from 0.590 to 0.683 (an increase of 0.093), indicating that the model with the ablation study configuration was better at detecting falls and reducing missed events. These results indicate that the ablation study configuration can improve the performance of the GCN model in detecting falls in the elderly, which has great potential for application in the development of future fall- and non-fall-focused elderly health monitoring systems.

**Keywords:** Eldery, Fall, Deep Learning, Graph Convolution Network, Mediapipe, Pose Landmark, Ablation Study

## DAFTAR ISI

<b>KATA PENGANTAR .....</b>	<b>ii</b>
<b>ABSTRAK .....</b>	<b>iii</b>
<b>ABSTRACT .....</b>	<b>iv</b>
<b>DAFTAR ISI .....</b>	<b>v</b>
<b>DAFTAR TABEL .....</b>	<b>viii</b>
<b>DAFTAR GAMBAR .....</b>	<b>ix</b>
<b>DAFTAR RUMUS .....</b>	<b>xi</b>
<b>DAFTAR KODE .....</b>	<b>xi</b>
<b>BABI I PENDAHULUAN.....</b>	<b>1</b>
1.1 Latar Belakang.....	1
1.2 Rumusan masalah.....	5
1.3 Tujuan .....	6
1.4 Batasan Masalah .....	6
1.5 Manfaat Penelitian .....	7
1.6 Sistematika Penulisan.....	7
1.6.1 Bab I Pendahuluan .....	7
1.6.2 Bab II Tinjauan Pustaka.....	8
1.6.3 Bab III Metode Penelitian .....	8
1.6.4 Bab IV Hasil dan Pembahasan.....	9
1.6.5 Bab V Kesimpulan dan Saran.....	9
<b>BAB II TINJAUAN PUSTAKA .....</b>	<b>10</b>
2.1 Tinjauan Pustaka.....	10
2.2 Dasar Teori .....	16

2.2.1	Lansia .....	16
2.2.2	Jatuh.....	17
2.2.3	<i>Computer Vision</i> .....	17
2.2.4	<i>Deep Learning</i> .....	18
2.2.5	Mediapipe Pose.....	18
2.2.6	<i>Graph Convolutional Network</i> .....	19
2.2.7	Data Graf .....	20
2.2.8	Pengukuran Performa Model .....	21
2.2.8.1	<i>Confussion Matrix</i> .....	21
<b>BAB III METODE PENELITIAN .....</b>	<b>24</b>	
3.1	Alur Penelitian .....	24
3.2	Penjabaran Alur Penelitian .....	25
3.2.1	Identifikasi Masalah .....	25
3.2.2	Studi Literatur .....	26
3.2.3	Pengumpulan Dataset.....	26
3.2.4	Prapemrosesan Data .....	27
3.2.4.1	Ekstraksi Citra Dari Video.....	27
3.2.4.2	Deteksi 33 <i>Landmark</i> Dengan <i>Mediapipe Pose</i> .....	29
3.2.4.3	Konversi <i>Pose Landmark</i> Ke Graf .....	33
3.2.4.4	Pembagian <i>K-Fold</i> Pada Data .....	35
3.2.5	Perancangan Model GCN .....	36
3.2.6	Pelatihan Model.....	40
3.2.7	Evaluasi Model .....	41
3.2.7.1	Percobaan Utama dengan Pengaturan Hyperparamter Default.....	41
3.2.7.2	Percobaan Tambahan Berdasarkan Studi Ablasi.....	45
3.3	Alat dan Bahan Tugas Akhir .....	49
3.3.1	Alat.....	49

3.3.2	Bahan.....	49
3.4	Ilustrasi Metode Pengembangan/Pengukuran.....	49
<b>BAB IV HASIL DAN PEMBAHASAN .....</b>		<b>56</b>
4.1	Hasil Implementasi.....	56
4.1.1	Analisis Deskriptif .....	56
4.1.2	Prapemrosesan Dataset .....	58
4.1.2.1	Ekstraksi Citra Dari video .....	59
4.1.2.2	Deteksi 33 <i>Landmark</i> Dengan <i>Mediapipe Pose</i> .....	61
4.1.2.3	Konversi <i>Pose Landmark</i> Ke Graf .....	62
4.1.2.4	Pembagian K-Fold Pada Dataset .....	63
4.1.3	Perancangan Model Klasifikasi .....	66
4.2	Evaluasi Hasil .....	68
4.2.1	Evaluasi Model dengan <i>Hyperparameter Default</i> .....	68
4.2.2	Uji Model Terbaik Pada Data Primer .....	70
4.2.3	Evaluasi Studi Ablasi .....	74
4.2.3.1	<i>Optimizer</i> .....	75
4.2.3.2	<i>Learning Rate</i> .....	76
4.2.3.3	<i>Batch Size</i> .....	76
4.2.3.4	<i>Weight Decay</i> .....	77
4.2.3.5	<i>Dropout</i> .....	78
4.3	Perbandingan Hasil <i>Hyperparameter Default</i> dengan Studi Ablasi	78
<b>BAB V KESIMPULAN DAN SARAN .....</b>		<b>81</b>
5.1	Kesimpulan.....	81
5.2	Saran .....	82
<b>DAFTAR PUSTAKA.....</b>		<b>84</b>
<b>LAMPIRAN.....</b>		<b>88</b>
A	Program <i>train_gcn_kfold</i> .....	88

## **DAFTAR TABEL**

Tabel 2.1 Literasi Penelitian Terdahulu.....	13
Tabel 3.1 Nilai Default <i>hyperparameter</i> model.....	42
Tabel 3.2 Variasi nilai <i>hyperparameter</i> model.....	47
Tabel 3.3 contoh nilai <i>Confussion Matrix</i> .....	53
Tabel 4.1 Penjabaran data video jatuh.....	57
Tabel 4.2 Penjabaran data video tidak jatuh.....	58
Tabel 4.3 Hasil penbagian E-Fold pada dataset per folder .....	65
Tabel 4.4 Konfigurasi <i>hyperparameter</i> untuk pelatihan model GCN..	68
Tabel 4.5 Hasil evaluasi per fold dan rerata.....	69
Tabel 4.6 Hasil evaluasi testing .....	71
Tabel 4.7 Hasil Prediksi Model Akhir.....	72
Tabel 4.8 Hasil evaluasi studi ablasi optimizer .....	75
Tabel 4.9 Hasil evaluasi studi ablasi learning rate .....	76
Tabel 4.10 Hasil evaluasi studi ablasi batch size .....	76
Tabel 4.11 Hasil evaluasi studi ablasi Weight Decay .....	77
Tabel 4.12 Hasil evaluasi studi ablasi Dropout .....	78
Tabel 4.13 Perbandingan Konfigurasi <i>hyperparameter</i> Default dan hasil Studi Ablasi .....	79
Tabel 4.14 Hasil evaluasi studi ablasi per fold dan rerata .....	79
Tabel 4.15 Perbandingan Hasil evaluasi konfigurasi hyperparamter default dan studi ablasi .....	80

## DAFTAR GAMBAR

Gambar 2.1 Defenisi landmark dalam mediapipe pose [26].....	19
Gambar 2.2 Visualisasi data graf.....	20
Gambar 2.3 Confusion matrix.....	22
Gambar 3.1 Alur penelitian.....	24
Gambar 3.2 contoh citra dari video jatuh.....	29
Gambar 3.3 Alur proses deteksi pose landmark.....	30
Gambar 3.4 Arsitektur jaringan BlazePose [26].....	31
Gambar 3.5 <i>Preview</i> hasil deteksi <i>landmark</i> terhadap citra.....	32
Gambar 3.6 Contoh data <i>pose landmark</i> .....	33
Gambar 3.7 Alur konversi data <i>pose landmark</i> ke data graf.....	33
Gambar 3.8 Contoh tampilan data graf.....	35
Gambar 3.9 Pembagian dataset menggunakan <i>5-Fold</i> .....	36
Gambar 3.10 Alur perancangan model.....	37
Gambar 3.11 Rancangan arsitektur model GCN.....	38
Gambar 4.1 Contoh dataset jatuh.....	56
Gambar 4.2 Contoh dataset tidak jatuh.....	57
Gambar 4.3 Kode pergeseran <i>start_time</i> .....	59
Gambar 4.4 Kode ekstrak frame acak .....	60
Gambar 4.5 Kode ekstrak <i>skeleton</i> dari citra.....	61
Gambar 4.6 <i>Preview</i> konversi citra ke <i>pose landmark</i> .....	61
Gambar 4.7 Kode inisialisasi struktur <i>skeleton</i> .....	62
Gambar 4.8 Kode konversi <i>skeleton</i> ke graf.....	63
Gambar 4.9 Kode membuat pembagian K-Fold .....	64
Gambar 4.10 Kode inisialisasi arsitektur model GCN .....	66
Gambar 4.11 Kode forward propagation model GCN .....	67

Gambar 4.12 akurasi hasil pelatihan model GCN dengan hyperparameter default .....	70
Gambar 4.13 Confusion matrix data test .....	71
Gambar 0.1 aggregate histories across folds .....	88
Gambar 0.2 build optimizer .....	88
Gambar 0.3 run grid search part 1 .....	89
Gambar 0.4 run grid search part 2 .....	90
Gambar 0.5 run grid search part 3 .....	91
Gambar 0.6 save checkpoint .....	92
Gambar 0.7 save model .....	92
Gambar 0.8 param groups with weight decay .....	92
Gambar 0.9 train epoch .....	93
Gambar 0.10 train single fold part 1 .....	93
Gambar 0.11 train single fold part 2 .....	94
Gambar 0.12 train single fold part 3 .....	95
Gambar 0.13 fungsi main .....	96
Gambar 0.14 manage saved models .....	97
Gambar 0.15 validate epoch .....	97

## **DAFTAR RUMUS**

Rumus 2.1 Matriks derajat.....	20
Rumus 2.2 Matriks ketetanggaan.....	20
Rumus 2.3 propagasi GCN .....	20
Rumus 2.4 accuracy .....	22
Rumus 2.5 <i>Precision</i> .....	22
Rumus 2.6 <i>recall</i> .....	23
Rumus 2.7 <i>F1-Score</i> .....	23

## **DAFTAR KODE**

# **BAB I**

## **PENDAHULUAN**

### **1.1 Latar Belakang**

Lanjut usia (lansia) merupakan fase alami dalam siklus kehidupan manusia yang akan dialami oleh setiap individu. *World Health Organization* (WHO) mengelompokkan populasi yang memasuki usia 60 tahun keatas sebagai lansia [1]. Populasi tersebut memiliki kecenderungan penurunan fungsi karakteristik fisik dan kondisi kesehatan yang semakin signifikan seiring bertambahnya usia. Perubahan fisiologis pada lansia seperti berkurangnya massa otot, penurunan kepadatan tulang, dan gangguan keseimbangan membuat mereka lebih rentan terhadap berbagai masalah kesehatan [2]. Kondisi ini membuat lansia lebih rentan terhadap masalah kesehatan dan keselamatan, seperti risiko terjatuh.

Jatuh merupakan suatu kondisi di mana seseorang tanpa sadar tiba-tiba berbaring atau terduduk ke permukaan apapun, seperti tanah, lantai, tempat tidur, atau permukaan lainnya. Jatuh juga didefinisikan sebagai suatu kejadian dimana seseorang tiba-tiba jatuh baik disengaja maupun tidak disengaja yang mengakibatkan luka atau cedera pada orang tersebut [3]. Orang dewasa dan lansia sangat rentan mengalami cedera akibat jatuh karena berbagai faktor fisik dan lingkungan. Penurunan fungsi sensorik, gangguan keseimbangan, berkurangnya massa otot, dan efek samping obat-obatan menjadi penyebab utama tingginya risiko jatuh pada populasi ini [4].

Menurut data statistik dari WHO, sekitar 37,3 juta kejadian jatuh yang memerlukan perhatian medis, bahkan terdapat 684.000 orang meninggal akibat terjatuh di seluruh dunia, di mana lebih dari 80% di antaranya terjadi di negara-negara berkembang [5]. Studi yang dilakukan *Australian Centre for Evidence Based Care* (ACEBAC) menyatakan bahwa Lansia yang tinggal di panti jompo lebih sering jatuh dari pada lansia yang tinggal dirumah mencapai 30-50%

setiap tahun dan meningkat 40% mengalami jatuh berulang [6]. Berdasarkan data survei *Indonesian Family Life Survey* (IFLS), Angka kejadian jatuh pada lansia di Indonesia yang berusia 65 tahun ke atas sekitar 30% dan usia 80 tahun ke atas sekitar 50% [1].

Di panti jompo atau lingkungan perawatan lansia, terdapat periode waktu di mana subjek lansia sedang sendirian dan tidak dalam pengawasan langsung oleh pengasuh atau perawat. Sebuah studi yang dilakukan Albasha, menunjukkan bahwa tingkat jatuh di fasilitas perawatan jangka panjang, termasuk panti jompo, jauh lebih tinggi dibandingkan dengan lingkungan pribadi atau keluarga, dengan rata-rata 1,7 atau hampir 2 kali kejadian jatuh per penghuni setiap tahun. Jatuh lebih sering terjadi pada malam hari atau selama pergantian *shift* karena rendahnya tingkat staf yang bertugas, mengakibatkan kurangnya pengawasan langsung terhadap penghuni [7]. Kondisi ini meningkatkan risiko keterlambatan penanganan karena ketidadaan sistem peringatan dini yang efektif untuk mendeteksi kejadian jatuh.

Keterlambatan penanganan pada lansia yang mengalami jatuh dapat menyebabkan komplikasi serius hingga kematian. Menurut penelitian oleh Fleming, fungsi otot lansia yang terjatuh akan menurun sehingga mengurangi kemampuan untuk bangun. Jika tidak mampu bangun selama lebih dari satu jam akan mengalami dehidrasi, hipotermia, pneumonia, dan uklus tekanan. dilakukan eksperimen dengan 20 orang yang berbaring di lantai selama lebih dari satu jam setelah jatuh, 36% (5 dari 14) pindah ke perawatan jangka panjang dalam satu tahun, dan 53% (8 dari 15) pindah ke perawatan jangka panjang pada akhir masa penelitian [8]. Hal ini menggarisbawahi dibutuhkannya sistem pendeteksi jatuh yang dapat berfungsi 24 jam sehari, terutama di area-area yang tidak selalu dalam pengawasan langsung.

Permasalahannya secara umum adalah bagaimana cara memberikan peringatan langsung kepada pengasuh atau petugas medis ketika subjek lansia mengalami jatuh. Sejauh ini teknologi deteksi jatuh telah dikembangkan dengan

3 pendekatan, antara lain *wearable sensors*, *vision* atau *computer vision* dan gabungan keduanya. Pada pendekatan *wearable*, seperti penggunaan perangkat sensor *accelerometer* dan IMU (*Inertial Measurement Unit*) memungkinkan pengumpulan data yang akurat selama aktivitas sehari-hari termasuk kejadian jatuh. Namun terdapat beberapa kelemahan yang perlu diperhatikan seperti sensor yang dipasang pada tubuh, seperti di pergelangan tangan atau pinggang, harus nyaman untuk dipakai sehari-hari, atau pengguna akan enggan menggunakannya. Selain itu terdapat ketergantungan pada kalibrasi membutuhkan penyesuaian manual, serta perangkat *wearable* memiliki masa pakai baterai yang terbatas sehingga ada kemungkinan di mana kejadian jatuh terjadi pada saat daya baterai habis [9].

Berbagai kekurangan dari pendekatan *wearable* menunjukkan kesempatan pada pendekatan *computer vision* atau visi komputer, yaitu bidang teknologi yang berfokus pada pengembangan algoritma dan sistem yang memungkinkan komputer untuk memperoleh, memproses, dan memahami informasi dari gambar atau video secara otomatis [10]. Teknologi ini meniru fungsi penglihatan manusia dengan cara mengekstrak informasi visual dari input yang tersedia dan mengubahnya menjadi format yang dapat diproses lebih lanjut oleh sistem komputer. Demi terwujudnya prinsip ini dibutuhkan teknik pengembangan algoritma seperti *Deep Learning* yang dapat menjadi inti dari sistem deteksi jatuh.

*Deep learning* adalah metode pembelajaran yang mengandalkan jaringan saraf tiruan dalam menganalisis dan memprediksi data melalui banyak lapisan pemrosesan [11]. *Deep learning* memanfaatkan arsitektur jaringan yang kompleks untuk mengekstraksi fitur dari data, yang dapat mengarah pada peningkatan kinerja dalam berbagai aplikasi, seperti pengenalan suara, pengenalan objek visual, dan berbagai aplikasi lain di bidang ilmu kehidupan dan industri. Beberapa sistem deteksi jatuh berbasis *Deep Learning* telah dikembangkan sebelumnya, namun masih memiliki berbagai

keterbatasan. Salah satu penelitian terkini yang relevan dalam bidang ini yaitu pengembangan model deteksi jatuh berbasis visi dengan memanfaatkan *Convolutional Neural Network* (CNN) dan *Long Short-Term Memory* (LSTM) untuk mengelola informasi temporal dari urutan gambar [12]. Meskipun inovatif, penelitian ini juga menunjukkan beberapa kelemahan signifikan yang perlu dicermati.

Salah satu kelemahan utama sistem tersebut terletak pada kebutuhan akan daya komputasi yang tinggi untuk menjalankan model deep learning untuk pengolahan citra. Menurut Chhetri, implementasi teknik *deep learning* pada model deteksi jatuh berbasis *computer vision* memerlukan sumber daya komputasi yang besar, yang mungkin tidak selalu tersedia dalam perangkat keras sederhana atau untuk penggunaan dalam skala besar [12]. Ini bisa menjadi kendala signifikan ketika mencoba menerapkan sistem pada perangkat mobile yang memiliki keterbatasan energi dan proses.

Pengolahan citra menggunakan MediaPipe *pose landmark* menawarkan solusi yang lebih efisien untuk deteksi jatuh. MediaPipe adalah framework *open-source* yang dikembangkan oleh Google untuk analisis pose tubuh atau wajah manusia melalui penerapan deteksi *landmark*, yang menggunakan model *deep learning* untuk mengekstraksi titik-titik kunci (*keypoints*) pada tubuh manusia dari gambar atau video. Dibandingkan dengan pendekatan pengolahan citra konvensional, model pose landmark menghasilkan data graph dengan ukuran yang jauh lebih kecil karena hanya menyimpan koordinat titik-titik kunci, bukan seluruh citra [13]. Penelitian terbaru di tahun 2024 menunjukkan bahwa optimasi arsitektur MediaPipe mampu mengurangi waktu pemrosesan per frame hingga 30%, sambil meningkatkan akurasi estimasi pose sebesar 20% berdasarkan metrik *Intersection over Union* (IoU) [14]. MediaPipe menunjukkan peningkatan ketahanan terhadap variasi pencahayaan dan oklusi parsial dengan memprioritaskan deteksi *keypoint* di atas analisis tingkat piksel, memanfaatkan jaringan resolusi tinggi (HRNet) untuk mempertahankan akurasi

bahkan ketika bagian tubuh dikaburkan [15].

*Graph Convolutional Network* (GCN) adalah metode *deep learning* yang dapat digunakan untuk memproses data berbentuk graf, seperti *pose landmark* tubuh manusia. GCN pertama kali diperkenalkan oleh Kipf dan Welling sebagai metode untuk mengklasifikasikan *node* dalam graf besar dengan memanfaatkan informasi topologi graf dan fitur *node* [16]. Dalam konteks deteksi jatuh, Kerangka manusia dapat direpresentasikan sebagai sebuah graf dengan sendi-sendi sebagai *node* dan tulang-tulang sebagai *edge*, lalu membuat pemodelan GCN untuk mempelajari hubungan spasial dan temporal antar titik-titik kunci tubuh manusia, sehingga dapat mengenali pola postur tubuh yang mengindikasikan posisi jatuh [17]. Penelitian yang dilakukan Zhang et al. (2023) menunjukkan bahwa pendekatan *Graph Convolutional Network* berbasis *pose landmark* mencapai akurasi 96,3% pada dataset MCF, lebih tinggi dibandingkan metode konvensional yang hanya mencapai 93,4% [18].

Penelitian ini berfokus pada pengembangan model deteksi jatuh dengan menerapkan *Graph Convolutional Network* (GCN) berbasis *pose landmark*. Pengembangan model ini diharapkan mampu mengidentifikasi posisi jatuh yang lebih akurat sehingga menjadi dasar pengembangan sistem yang baru untuk peringatan dini kepada pengasuh atau petugas medis ketika subjek lansia mengalami jatuh, sehingga pertolongan dapat segera diberikan. Penelitian ini juga diharapkan dapat menjadi dasar pengembangan teknologi yang lebih canggih sehingga dapat mengurangi risiko komplikasi serius dan kematian akibat keterlambatan penanganan pasca-jatuh pada lansia di panti jompo, rumah sakit, atau bahkan rumah pribadi di masa depan, seiring dengan semakin meningkatnya populasi lansia di Indonesia dan dunia.

## 1.2 Rumusan masalah

Berdasarkan latar belakang yang telah dijelaskan, maka rumusan masalah dalam penelitian ini sebagai berikut.

1. Bagaimana cara mengembangkan model menggunakan *Graph Convolutional Network* (GCN) berbasis *pose landmark* Mediapipe untuk mendeteksi posisi jatuh pada lansia?
2. Bagaimana hasil evaluasi performa, terutama berdasarkan akurasi dari *confusion matrix* pada model deteksi posisi jatuh GCN berbasis *pose landmark* Mediapipe dengan membandingkan konfigurasi *hyperparameter* default dan hasil studi ablasi?

### **1.3 Tujuan**

Tujuan dari penelitian ini sebagai berikut.

1. Mengembangkan model deteksi posisi jatuh pada lansia menggunakan GCN berbasis pose landmark Mediapipe.
2. Melakukan evaluasi performa, terutama berdasarkan akurasi dari *confusion matrix* pada model deteksi posisi jatuh GCN berbasis *pose landmark* Mediapipe dengan membandingkan Hyperparameter default dan studi ablasi.

### **1.4 Batasan Masalah**

Agar penelitian ini lebih terfokus, beberapa batasan masalah yang ditetapkan sebagai berikut

1. Penelitian ini hanya menggunakan dataset simulasi jatuh yang telah dibuat oleh Baldewijns (2016), terdiri dari 275 video jatuh dan 85 video tidak jatuh.
2. Penelitian ini hanya berfokus pada pipeline pembangunan model dan eksplorasi hyperparameter saja, tidak melakukan perbandingan efisiensi komputasi dengan penilitian terdahulu.
3. Frame citra yang diproses adalah citra yang berisi 1 orang dengan pose keseluruhan tubuh yang jelas dan tidak tertutup oleh objek apapun.
4. Penelitian ini tidak menerapkan multimodal dalam pengembangan dan

pelatihan model.

5. Hanya berfokus pada pengolahan dataset kumpulan video dimana tiap video akan diekstrak menjadi frame citra, lalu penerapan *framework* Mediapipe *Pose Landmark* untuk mendeteksi pose manusia dalam tiap citra dan terakhir pembangunan model berbasis GCN untuk memprediksi posisi jatuh dan tidak jatuh pada manusia.
6. Framework Mediapipe Pose Landmark hanya sebatas pemakaian produk.
7. Model yang dikembangkan hanya melakukan klasifikasi antara 2 kelas saja, jatuh dan tidak jatuh.

## 1.5 Manfaat Penelitian

Penelitian ini diharapkan memberikan manfaat sebagai berikut:

1. Memberikan informasi bagaimana proses pengembangan model deteksi kejatuhan pada lansia menggunakan Graph Convolutional Network (GCN) berbasis pose landmark Mediapipe.
2. Mempelajari dan mengevaluasi performa model Graph Convolutional Network (GCN) berbasis pose landmark Mediapipe agar menjadi bahan referensi untuk pengembangan model deteksi posisi jatuh pada lansia di masa depan.

## 1.6 Sistematika Penulisan

Pada subbab ini akan menjelaskan isi dari setiap bab dalam laporan penelitian ini secara umum. Adapun sistematika penulisan dalam laporan ini adalah sebagai berikut.

### 1.6.1 Bab I Pendahuluan

Bab ini memaparkan latar belakang, rumusan masalah, tujuan penelitian, batasan masalah, manfaat penelitian, dan sistematika penulisan. Permasalahan penelitian dirumuskan untuk memberikan fokus penelitian yang jelas, sementara tujuan penelitian menentukan pencapaian yang diharapkan. Batasan

masalah mengarahkan ruang lingkup penelitian, dan manfaat penelitian diharapkan memberikan kontribusi positif baik teori maupun praktik.

### **1.6.2 Bab II Tinjauan Pustaka**

Bab ini mengulas landasan teori, tinjauan pustaka, serta defenisi teori dan teknologi yang digunakan dalam penelitian, yang mencakup konsep-konsep dasar yang relevan untuk memahami topik yang dibahas. Dalam bagian ini, akan dijelaskan mengenai pose landmark dan Graph Convolutional Network (GCN) yang menjadi teknologi utama yang diterapkan dalam penelitian ini. Penjelasan mengenai pose landmark akan menguraikan cara teknologi ini mendeteksi dan menganalisis posisi tubuh manusia, sementara GCN akan dibahas dalam konteks bagaimana jaringan konvolusional graf dapat digunakan untuk memproses data spasial yang kompleks. Selain itu, studi literatur terkait deteksi kejatuhan akan dibahas untuk memberikan gambaran mengenai penelitian terdahulu yang relevan dan bagaimana temuan-temuan tersebut mendasari pengembangan penelitian ini.

### **1.6.3 Bab III Metode Penelitian**

Bab ini menjelaskan secara rinci metode penelitian yang digunakan, yang mencakup langkah-langkah sistematis dalam proses penelitian. Pertama, dijelaskan mengenai pengumpulan data yang dilakukan untuk memperoleh informasi yang diperlukan, dengan menggambarkan teknik pengumpulan data serta sumber-sumber yang relevan untuk mendukung penelitian ini. Selanjutnya, proses prapemrosesan data akan diuraikan agar siap digunakan dalam analisis lebih lanjut. Implementasi metode GCN juga akan dijelaskan secara mendalam, mencakup bagaimana GCN diterapkan untuk memproses data yang diperoleh dan mengidentifikasi pola-pola dalam deteksi posisi jatuh, serta pengoptimalan model untuk mencapai hasil yang akurat dan efisien.

#### **1.6.4 Bab IV Hasil dan Pembahasan**

Bab ini berisi hasil penelitian yang diperoleh melalui penerapan metode yang telah dijelaskan sebelumnya, serta analisis mendalam terhadap temuan-temuan yang didapatkan. Dalam bagian ini, akan dipaparkan hasil eksperimen yang dilakukan untuk menguji kemampuan model dalam mendeteksi kejadian jatuh dan tidak jatuh, termasuk visualisasi hasil deteksi yang dihasilkan oleh model. Evaluasi performa model akan dilakukan dengan menggunakan metrik yang relevan, seperti akurasi, presisi, *recall*, dan *F1-score*, untuk menilai sejauh mana model dapat membedakan antara kedua kondisi tersebut.

#### **1.6.5 Bab V Kesimpulan dan Saran**

Bab ini menyajikan kesimpulan yang merangkum temuan-temuan utama dari penelitian yang telah dilakukan, serta implikasi dari hasil yang diperoleh dalam pengembangan deteksi posisi jatuh menggunakan metode GCN berbasis *pose landmark*. Kesimpulan ini mengidentifikasi pencapaian utama penelitian, termasuk keberhasilan model dalam mendeteksi kejadian jatuh dan tidak jatuh, serta evaluasi terhadap efektivitas metode yang diterapkan. Selain itu, bab ini juga menyajikan saran untuk pengembangan lebih lanjut, meliputi potensi peningkatan model dengan memanfaatkan data tambahan atau teknik lain yang dapat meningkatkan akurasi deteksi. Saran ini juga mencakup arahan untuk penelitian lanjutan, serta aplikasi teknologi yang lebih luas dalam bidang deteksi kecelakaan pada lansia atau di lingkungan lain yang relevan, guna memberikan manfaat yang lebih besar bagi masyarakat.

## **BAB II**

### **TINJAUAN PUSTAKA**

#### **2.1 Tinjauan Pustaka**

Subbab ini menyajikan tinjauan pustaka yang bertujuan memberikan landasan teoretis bagi penelitian ini. Referensi yang digunakan diambil dari penelitian-penelitian serupa yang telah dilakukan sebelumnya sebagai dasar untuk penelitian ini.

Pada penelitian yang dilakukan oleh O. Reyad, H.I. Shehata, dan M.E. Karar (2023), mengangkat permasalahan tentang jatuh sebagai salah satu peristiwa berbahaya yang sering dialami lansia, dengan data *World Health Organization* (WHO) menunjukkan sekitar 30% orang berusia di atas 65 tahun mengalami minimal satu kali jatuh per tahun, meningkat hingga 50% untuk usia di atas 80 tahun. Peneliti mengembangkan arsitektur *Internet of Health Things (IoHT)* dengan sistem deteksi jatuh berbasis *ensemble machine learning*, khususnya model *multi-stage random forest*. Metodologi penelitian menggunakan dataset *SmartFall* yang mengumpulkan data dari *accelerometer* tiga sumbu pada *smartwatch*, kemudian membandingkan model yang diusulkan dengan tiga model *machine learning* lainnya: *K-nearest neighbors (KNN)*, *decision tree (DT)*, dan *standard random forest (SRF)*. Hasil penelitian menunjukkan bahwa model *ensemble random forest* yang diusulkan mencapai akurasi tertinggi sebesar 98,4%, mengungguli model *machine learning* lainnya dalam deteksi jatuh pada lansia [19].

Sementara itu, Penelitian yang berjudul *Human Fall Detection from Sequences of Skeleton Features using Vision Transformer* oleh A Raza et al (2023), mengatasi tantangan ketergantungan metode deteksi jatuh konvensional pada sensor *depth* (seperti *Kinect*) dan model berbasis LSTM/CNN yang memerlukan komputasi intensif. Permasalahan utama termasuk keterbatasan

akurasi dalam lingkungan dengan variasi pencahayaan, oklusi, serta kebutuhan akan sistem non-intrusif yang menjaga privasi pengguna dengan memanfaatkan data *skeleton*. Metode yang diusulkan menggabungkan *BlazePose* (untuk ekstraksi 33 titik *skeleton* dari video RGB) dengan *multi-headed transformer encoder* yang memproses urutan fitur *skeleton*. Preprocessing menggunakan normalisasi posisi relatif (RP) dan interpolasi linear untuk mengatasi titik *skeleton* yang hilang (reduksi dari 22,38% menjadi 1,8%). Hasil eksperimen pada dataset *UR-FALL* dan *UP-FALL* menunjukkan akurasi masing-masing 96,12% dan 97,36%, mengungguli metode berbasis CNN (95%) dan LSTM (91,07%) [20].

Kemudian penelitian berjudul "*Pose-Based Fall Detection System: Efficient Monitoring on Standard CPUs*" oleh V. Mali dan S. Jaiswal (2024) mengatasi permasalahan ketergantungan sistem deteksi jatuh konvensional pada sensor *wearable*, *pressure mat*, atau model CNN berbasis GPU yang mahal dan tidak praktis untuk implementasi di panti jompo. Penelitian ini mengusulkan sistem non-intrusif berbasis *pose estimation* menggunakan *MediaPipe* untuk ekstraksi 33 *landmark* tulang dari video *RGB*. Metode utama meliputi analisis fitur berbasis *threshold* (rasio tinggi badan, sudut torso-kaki, jarak kepala-lantai) dan mekanisme voting 20-frame buffer untuk meminimalkan *false positive*. Hasil pengujian pada dataset GMDCSA24 (130 video aktivitas lansia) menunjukkan akurasi 91,54%, presisi 88,24%, dan recall 98,68%, dengan spesifikasi 81,48%. Sistem ini berjalan pada CPU standar tanpa memerlukan GPU atau sensor tambahan, menjadikannya solusi hemat biaya untuk fasilitas perawatan lansia [21].

Terdapat juga Penelitian yang berjudul "*Vision-Based Fall Detection Using Dense Block With Multi-Channel Convolutional Fusion Strategy*" oleh X. Cai et al (2021) mengatasi masalah kehilangan informasi pada jaringan *neural* dalam untuk deteksi jatuh berbasis visi. Permasalahan utama terletak pada degradasi informasi selama propagasi fitur di lapisan jaringan

yang dalam, yang mengurangi kemampuan ekstraksi fitur dan menurunkan akurasi deteksi. Untuk mengatasi hal ini, penulis mengusulkan arsitektur MCCF-DenseBlock (Multi-Channel Convolutional Fusion Dense Block) yang menggabungkan koneksi padat (dense connections) dengan strategi fusi konvolusional multi-saluran. Arsitektur ini dilengkapi improved transition layer untuk mengurangi redundansi data melalui *multi-level downsampling*. Hasil eksperimen menggunakan *UR Fall Dataset* menunjukkan kinerja superior dengan F-score 0.973, mengungguli lima metode pembanding: dua berbasis fitur hand-crafted (GLR-FD dan MEWMA-FD) dan tiga berbasis deep learning (CNN-FD, Bi-LSTM-FD, CNN-LSTM-FD) [22].

Terakhir, Penelitian berjudul *Fall Detection Method for Infrared Videos Based on Spatial-Temporal Graph Convolutional Network* oleh J. Yang et al (2024) mengatasi permasalahan keterbatasan kamera cahaya tampak (RGB) dalam deteksi jatuh, terutama terkait pelanggaran privasi, adaptasi lingkungan rendah cahaya, dan akurasi yang tidak optimal. Penulis mengusulkan metode berbasis video inframerah (*near-infrared/NIR* dan *thermal-infrared/TIR*) yang menggabungkan *AlphaPose* untuk ekstraksi kerangka tubuh 2D dan jaringan *Spatial-Temporal Graph Convolutional Network* (ST-GCN) yang dimodifikasi. Modifikasi ST-GCN meliputi strategi partisi spasial yang diperluas (*expanded spatial partitioning*), unit konvolusi temporal multi-skala, serta representasi kerangka dalam sistem koordinat Kartesius dan polar. Hasil eksperimen pada dataset *IR-Fall (self-collected)* menunjukkan akurasi tertinggi 96.37%, sedangkan pada dataset publik NTU-RGB+D-120 mencapai 94.64%, mengungguli metode GCN lain seperti ST-GCN dasar (95.00% pada *IR-Fall*), CPS-GCN (95.35%), dan AS-GCN (94.67%). Metode ini juga dioptimalkan untuk *real-time* dengan *pruning* lapisan jaringan, mengurangi parameter model hingga 1.766 MB tanpa mengorbankan akurasi [23]. Berbagai referensi tersebut dijabarkan pada Tabel 2.1.

Tabel 2.1 Literasi Penelitian Terdahulu

No	Peneliti	Judul Penelitian	Permasalahan	Metode	Hasil
1.	Yang, He, Zhu, Lv dan Jin (2024)	<i>Fall Detection Method for Infrared Videos Based on Spatial-Temporal Graph Convolutional Network</i> [23]	Dibutuhkan sistem deteksi jatuh yang akurat, <i>real-time</i> , menjaga privasi, dan dapat beroperasi baik pada video <i>near-infrared</i> (NIR) maupun <i>thermal-infrared</i> (TIR)	<i>AlphaPose</i> dan ST-GCN	Metode tersebut mencapai akurasi 96.37% pada dataset <i>IR-Fall</i> dan 94.64% pada NTU-RGB+D-120.

No	Peneliti	Judul Penelitian	Permasalahan	Metode	Hasil
2.	Raza, Yousaf, Velastin, and Viriri (2023)	<i>Human Fall Detection from Sequences of Skeleton Features using Vision Transformer</i> [20]	Ketergantungan pada sensor <i>depth</i> dan model berbasis LSTM/CNN	<i>BlazePose multi-headed transformer encoder (Vision Transformer)</i>	Metode tersebut mencapai akurasi 96.12% pada dataset <i>UR-FALL</i> dan akurasi 97.36% pada dataset <i>UP-FALL</i> .
3.	Mali dan Jaiswal (2025)	<i>Pose-Based Fall Detection System: Efficient Monitoring on Standard CPUs</i> [21]	Diperlukan sistem deteksi jatuh pada lansia di panti jompo yang lebih ringan dan praktis daripada sistem berbasis sensor atau video	<i>MediaPipe, threshold-based analysis and voting mechanism</i>	tercapai akurasi 91,54% dan tidak memerlukan perangkat keras mahal atau sensor tambahan.

No	Peneliti	Judul Penelitian	Permasalahan	Metode	Hasil
4.	Reyad, Shehata, dan Karar (2023)	<i>Developed Fall Detection of Elderly Patients in Internet of Healthcare Things [19]</i>	dibutuhkan sistem deteksi jatuh yang akurat dan efisian secara komputasi dan dapat diterapkan pada perangkat wearable	<i>Internet of Healthcare Things ensemble Machine Learning, khususnya model multi-stage ensemble random forest</i>	Model ensemble random forest menunjukkan akurasi deteksi jatuh sebesar 98,4%
5.	Cai, Liu, An, dan Han (2021)	<i>Vision-Based Fall Detection Using Dense Block With Multi-Channel Convolutional Fusion Strategy [22]</i>	kehilangan informasi pada metode deteksi jatuh berbasis deep learning, khususnya pada jaringan dengan lapisan yang dalam.	Arsitektur MCCF-DenseBlock (Multi-Channel Convolutional Fusion Dense Block)	Hasil pengujian dengan dataset UR Fall menunjukkan F-score mencapai 0.973

Berdasarkan tinjauan terhadap penelitian terdahulu, terlihat bahwa belum ada studi yang secara spesifik menerapkan *Graph Convolutional Network* berbasis *MediaPipe* untuk mendeteksi kejadian jatuh pada data frame RGB.

Sebagian besar penelitian sebelumnya tetap berfokus pada pengembangan sistem deteksi kejadian jatuh yang efisien, namun mempertimbangkan berbagai faktor, seperti perangkat *wearable*, isu keamanan dan privasi data, ketergantungan pada sensor, serta pengujian berbagai model *deep learning*. Pada penelitian ini, implementasi GCN berbasis *MediaPipe* akan menjadi fokus utama untuk mendeteksi kejadian jatuh pada lansia. Analisis pemanfaatan metode tersebut bertujuan untuk menganalisis alternatif sistem deteksi yang berkemungkinan lebih efisien dan akurat, serta dapat beroperasi dengan perangkat yang lebih ringan dan tidak mengganggu kenyamanan pengguna sekaligus memberikan kontribusi pada pengembangan sistem pemantauan kesehatan lansia yang lebih canggih dan praktis.

## 2.2 Dasar Teori

Pada subbab ini dijabarkan berbagai teori yang menjadi dasar dalam penelitian ini. teori-teori tersebut antara lain defenisi mengenai lansia, kejadian jatuh, *Computer Vision*, *Machine Learning*, *Deep Learning*, *Mediapipe*, *Graph Convolutional Network* (GCN), dan pengukuran peforma model. berikut rincian berbagai teori tersebut.

### 2.2.1 Lansia

Lanjut usia (lansia) dapat dikatakan sebagai tahap akhir dari siklus kehidupan manusia yang tidak dapat dihindari. World Health Organization (WHO) South East Asia Regional (SEAR) mengelompokan populasi lansia menjadi 3 kelompok, yaitu usia lanjut, tertua dan centenarian. Adapun usia lanjut memasuki usia 60 tahun keatas, tertua mulai di 80 tahun keatas dan centenarian di usia 100 tahun keatas. Seiring bertambahnya usia, lansia mengalami pengurangan dalam kondisi fisik mereka, yang mencakup penurunan kekuatan otot, mobilitas, dan keseimbangan [1].

### **2.2.2 Jatuh**

Jatuh merupakan suatu kondisi dimana seseorang atau subjek tanpa sadar tiba-tiba berbaring atau terduduk di permukaan tanah atau lantai. Jatuh juga definisikan sebagai suatu kejadian dimana seseorang tiba-tiba jatuh baik disengaja maupun tidak disengaja yang mengakibatkan luka atau cedera pada orang tersebut [3]. Orang dewasa dan lansia sangat rentan mengalami cedera akibat jatuh karena berbagai faktor fisik dan lingkungan. Penurunan fungsi sensorik, gangguan keseimbangan, berkurangnya massa otot, dan efek samping obat-obatan menjadi penyebab utama tingginya risiko jatuh pada populasi ini [4].

Bahaya jatuh di kalangan lansia tidak hanya terbatas pada cedera fisik, tetapi juga dapat menciptakan masalah psikologis dan sosial. Insiden jatuh dapat mengakibatkan trauma serius, seperti patah tulang atau cedera otot yang parah, yang sekaligus memperburuk ketergantungan mereka dalam aktivitas sehari-hari [24]. Sebagai contoh, biaya dan waktu yang diperlukan untuk proses penyembuhan dari cedera serius dapat menyebabkan stres dan kecemasan bagi lansia dan keluarganya, serta menurunkan kualitas hidup lansia itu sendiri [25].

### **2.2.3 Computer Vision**

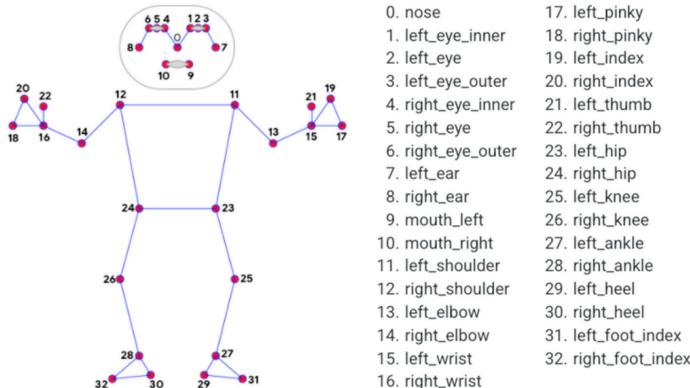
Computer vision atau visi komputer adalah bidang teknologi yang berfokus pada pengembangan algoritma dan sistem yang memungkinkan komputer untuk memperoleh, memproses, dan memahami informasi dari gambar atau video secara otomatis. Teknologi ini meniru fungsi penglihatan manusia dengan cara mengekstrak informasi visual dari input yang tersedia dan mengubahnya menjadi format yang dapat diproses lebih lanjut oleh sistem komputer. Oleh karena itu, computer vision berperan penting dalam berbagai aplikasi seperti pengenalan wajah, deteksi objek, dan pengolahan citra [10].

#### 2.2.4 Deep Learning

*Deep learning* adalah metode pembelajaran yang mengandalkan jaringan saraf tiruan dalam menganalisis dan memprediksi data melalui banyak lapisan pemrosesan. Sebagai subkategori dari Kecerdasan buatan, *deep learning* memanfaatkan arsitektur jaringan yang kompleks untuk mengekstraksi fitur dari data, yang dapat mengarah pada peningkatan kinerja dalam berbagai aplikasi, seperti pengenalan suara, pengenalan objek visual, dan berbagai aplikasi lain di bidang ilmu kehidupan dan industri. Pelatihan jaringan saraf dalam deep learning melibatkan algoritma backpropagation untuk menyesuaikan parameter internal sehingga tiap lapisan dapat belajar dari data dengan lebih efektif [11]. Dengan demikian, metode ini memungkinkan komputer untuk mengidentifikasi pola dan relasi dalam dataset besar secara otomatis, tanpa memerlukan intervensi manusia untuk mendesain fitur.

#### 2.2.5 Mediapipe Pose

MediaPipe Pose adalah *framework* yang dikembangkan oleh Google untuk mendeteksi dan memperkirakan posisi sendi tubuh manusia dalam gambar. MediaPipe Pose menggunakan teknik *machine learning* untuk memproses data dan menghasilkan estimasi koordinat tubuh manusia dalam bentuk 2D [13]. *Framework* ini dirancang untuk memberikan akurasi tinggi dengan performa cepat yang dapat dijalankan di perangkat dengan sumber daya terbatas, seperti ponsel dan komputer pribadi, menggunakan inferensi CPU.



Gambar 2.1 Defenisi landmark dalam mediapipe pose [26].

Salah satu komponen utama yang digunakan dalam MediaPipe Pose adalah BlazePose, sebuah arsitektur *machine learning* yang ringan dan efisien. *BlazePose* bertanggung jawab untuk mendeteksi 33 landmark tubuh manusia yang mencakup titik-titik penting seperti kepala, bahu, tangan, pinggul, lutut, dan kaki seperti yang terlihat pada Gambar 2.1. Landmark ini digunakan untuk menganalisis dan melacak pose tubuh secara dinamis dalam video, baik dalam mode 2D maupun 3D [26].

### 2.2.6 Graph Convolutional Network

GCN adalah pendekatan *deep learning* pada data yang terstruktur sebagai graf, yang didasarkan pada varian efisien dari *convolutional neural networks* (CNN) yang beroperasi langsung pada graf. Model ini memanfaatkan arsitektur konvolusi yang dimotivasi oleh pendekatan lokal orde pertama dari *spectral graph convolutions*. GCN mampu melakukan propagasi informasi fitur dari node-node tetangga dalam graf, sehingga dapat mempelajari representasi tersembunyi yang mengkode baik struktur lokal graf maupun fitur dari node itu sendiri. Model ini juga dioptimalkan agar skalabilitasnya linear terhadap jumlah edge dalam graf, sehingga cocok untuk graf berskala besar. [16].

Secara formal, GCN didefinisikan dengan aturan propagasi per lapisan dengan rumus Rumus 2.3.

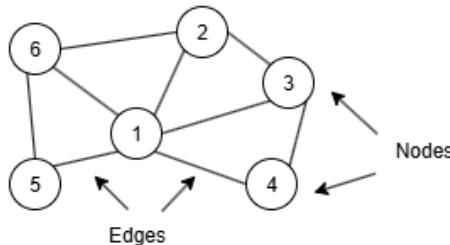
$$\hat{D} = D + I \quad (\text{Rumus 2.1})$$

$$\hat{A} = A + I \quad (\text{Rumus 2.2})$$

$$H^{(l+1)} = f(H^l, A) = \sigma \left( \hat{D}^{-\frac{1}{2}} \hat{A} \hat{D}^{-\frac{1}{2}} H^{(l)} W^{(l)} \right) \quad (\text{Rumus 2.3})$$

Matriks  $D$  dan  $A$  masing-masing merupakan matriks derajat dan matriks ketetanggaan, sedangkan  $I$  adalah matriks identitas, dan  $\hat{D}$  adalah matriks diagonal derajat simpul dari  $\hat{A}$ . Parameter jaringan  $W^{(l)}$  pada lapisan ke- $l$  dipelajari untuk menghasilkan embedding simpul  $H^{(l+1)}$  dari embedding pada langkah message-passing sebelumnya  $H^{(l)}$ , dengan fungsi aktivasi non-linear  $f$ . Normalisasi  $\hat{D}^{-1/2} \hat{A} \hat{D}^{-1/2}$  berfungsi menambahkan sambungan diri pada setiap simpul dan menjaga skala vektor fitur. Selama pelatihan, simpul-simpul yang terhubung dengan bobot sisi yang tinggi akan menjadi semakin mirip ketika melalui beberapa lapisan.

### 2.2.7 Data Graf



Gambar 2.2 Visualisasi data graf.

Dalam konteks Graph Convolutional Networks (GCN), graf dapat didefinisikan sebagai struktur yang terdiri dari simpul (nodes) dan tepi

(edges) yang menghubungkan simpul-simpul tersebut, seperti yang terlihat pada Gambar 2.2. Graf ini berfungsi sebagai representasi dari data yang memiliki hubungan atau interaksi antar elemen. GCN memanfaatkan struktur graf ini untuk melakukan pembelajaran yang lebih efisien dan efektif, terutama dalam konteks pembelajaran semi-terawasi, klasifikasi, dan rekomendasi [16].

### 2.2.8 Pengukuran Performa Model

Evaluasi performa model deteksi gerak jatuh pada lansia memerlukan metrik yang dapat menggambarkan akurasi dan kualitas prediksi secara menyeluruh. Beberapa metrik yang umum digunakan dalam konteks ini adalah *Loss*, *Accuracy*, *Precision*, dan *F1 Score*. Masing-masing metrik memiliki peran penting dalam menilai efektivitas model dalam mendeteksi kejadian jatuh pada lansia.

#### 2.2.8.1 Confusion Matrix

Confusion matrix atau matriks kebingungan adalah alat evaluasi yang sangat penting dalam bidang *Deep Learning*, terutama dalam analisis klasifikasi. Matriks ini memberikan gambaran yang lengkap tentang performa model klasifikasi dengan menunjukkan jumlah prediksi yang benar dan salah untuk setiap kelas yang dipertimbangkan. Dengan menggunakan *confusion matrix*, peneliti dapat mengevaluasi apakah modelnya cenderung membuat kesalahan dalam klasifikasi tertentu, memberikan wawasan yang jelas mengenai jenis kesalahan yang terjadi dalam prediksi. Hal ini sangat diperlukan untuk mengoptimalkan model serta menentukan langkah-langkah perbaikan yang diperlukan dalam desain algoritma(Tharwat, 2020).

Tujuan utama dari penggunaan *confusion matrix* adalah untuk memberikan gambaran evaluasi yang lebih mendetail daripada hanya sekedar akurasi keseluruhan. terlihat pada 2.3, Metrik ini secara sederhana menghitung 4 nilai utama, antara laian.

		Label Prediksi	
		1 (Positive)	0 (Negative)
Label Sebenarnya	1 (Positive)	TP (True Positive)	FN (False Negative)
	0 (Negative)	FP (False Positive)	FN (True Negative)

Gambar 2.3 Confusion matrix.

1. *True Positive* (TP): Total prediksi positif yang benar
2. *True Negative* (TN): total prediksi negatif yang benar
3. *False Positive* (FP): total prediksi positif yang salah
4. *False Negative* (FN): total prediksi negatif yang salah[7].

*Confusion matrix* berperan krusial dalam analisa diagnostik, di mana pemahaman yang mendalam tentang bagaimana dan mengapa suatu klasifikasi bisa gagal sangat penting untuk pengembangan model yang lebih baik di masa mendatang. Berdasarkan 4 nilai utama metrik tersebut, terdapat beberapa rumus Perhitungan lanjutan untuk mendeskripsikan peforma model secara lebih mendalam, berbagai rumus itu antara lain

1. *Accuracy* mengukur proporsi prediksi yang benar dari total prediksi yang dilakukan oleh model. Metrik ini dihitung dengan rumus Rumus 2.4.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (\text{Rumus 2.4})$$

2. *Precision* mengukur akurasi dari prediksi positif yang dilakukan oleh model. Metrik ini dihitung dengan rumus Rumus 2.5.

$$\text{Precision} = \frac{TP}{TP + FP} \quad (\text{Rumus 2.5})$$

3. *Recall* atau sensitivitas mengukur kemampuan model dalam mendekripsi

semua kejadian positif yang sebenarnya. Metrik ini dihitung dengan rumus Rumus 2.6.

$$\text{Recall} = \frac{TP}{TP + FN} \quad (\text{Rumus 2.6})$$

4. *F1 Score* adalah rata-rata harmonik dari *precision* dan *recall*, memberikan keseimbangan antara keduanya. Metrik ini dihitung dengan rumus Rumus 2.7.

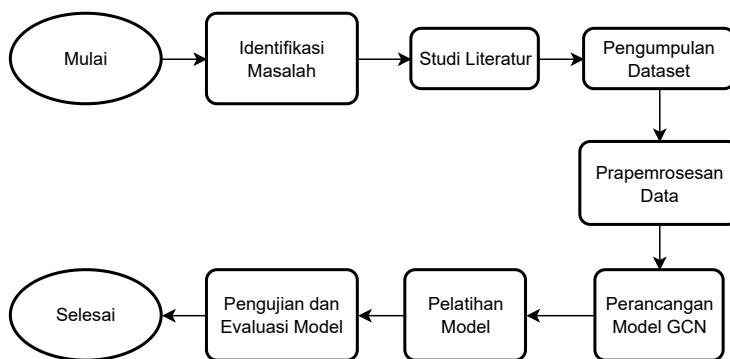
$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (\text{Rumus 2.7})$$

## BAB III

### METODE PENELITIAN

#### 3.1 Alur Penelitian

Alur penelitian ini menggambarkan tahapan yang dilakukan dari awal hingga akhir untuk menyelesaikan penelitian mengenai deteksi kejatuhan manusia menggunakan *Graph Convolutional Network* (GCN) berbasis *pose landmark*. Alur penelitian digambarkan dalam bentuk *flowchart* berikut.



Gambar 3.1 Alur penelitian.

Tahapan-tahapan yang dilakukan dalam penelitian secara keseluruhan terlihat pada gambar 3.1. Proses dimulai dengan identifikasi masalah dan tujuan penelitian untuk menentukan fokus dan arah dari penelitian. Selanjutnya, dilakukan Studi Literatur untuk memahami penelitian sebelumnya yang relevan agar menambah wawasan dan menemukan kekurangan atau area yang dapat ditingkatkan dalam bidang tersebut. Setelah itu, dilakukan Pengumpulan dataset yang mencakup pengumpulan dataset video yang dibutuhkan untuk eksperimen.

Tahapan berikutnya adalah prapemrosesan data dan ekstraksi fitur, di mana

data yang terkumpul diolah dan dipersiapkan untuk digunakan dalam model. Ini mencakup ekstraksi *frame* dan pengolahan data *pose landmark*. Kemudian Membangun model GCN dilakukan untuk merancang dan menyusun model yang akan digunakan dalam penelitian ini. Selanjutnya, dilakukan pelatihan model menggunakan dataset yang telah diproses, diikuti dengan evaluasi performa model untuk mengukur akurasi dan kinerja model dalam mendeteksi posisi jatuh pada lansia. Proses ini berulang, dengan model terus diperbaiki berdasarkan hasil evaluasi. Akhirnya, setelah seluruh tahapan selesai.

### **3.2 Penjabaran Alur Penelitian**

Subbab ini akan menjelaskan setiap langkah dalam alur penelitian secara detail dan menyeluruh. Subbab ini bertujuan untuk memperjelas alur penelitian dan mencegah munculnya berbagai pemikiran atau pendapat yang keliru mengenai penelitian ini. Seluruh langkah penelitian yang telah ditampilkan pada subbab 3.1 akan dijabarkan secara rinci sebagai berikut.

#### **3.2.1 Identifikasi Masalah**

Tahap pertama dari penelitian ini adalah memulai proses yang mencakup pemahaman mendalam tentang tujuan dari penelitian ini serta pengaturan langkah-langkah yang akan diambil. Penelitian ini dimulai dengan menetapkan gambaran besar dan objektif penelitian, dengan masalah utama yang dihadapi adalah tingginya angka kecelakaan yang terjadi pada lansia, terutama yang disebabkan oleh kejadian jatuh yang kurang diperhatikan. Jatuh pada lansia sering kali mengakibatkan cedera serius dan bahkan kematian.

Berdasarkan permasalahan jatuh tersebut, penelitian ini berfokus untuk mengembangkan model deteksi posisi jatuh yang dapat membantu mengidentifikasi kejadian jatuh dengan cepat, yang berpotensi menyelamatkan nyawa. Berdasarkan seluruh informasi tersebut, maka tujuan dari penelitian ini adalah merancang dan membangun model yang dapat menganalisis data

*pose landmark* tubuh dari lansia menggunakan Mediapipe, lalu memanfaatkan teknologi GCN untuk mendeteksi posisi jatuh berdasarkan data tersebut.

### 3.2.2 Studi Literatur

Langkah ini dapat dikatakan sebagai pondasi utama dalam penelitian ini. Lankah-lankah selanjut nya dalam alur penelitian ini tentunya membutuhkan pengetahuan mendalam yang bisa didapatkan dari berbagai sumber, salah satunya literatur penelitian. Berbagai literatur terdahulu yang berkaitan dengan penelitian ini harus dipelajari, dengan kalimat kunci antara lain *Graph Convolutional Network* (GCN), Kejadian jatuh pada lansia, Mediapipe, *pose Landmark*, dan sebagainya.

### 3.2.3 Pengumpulan Dataset

Langkah selanjutnya adalah pengumpulan dataset yang diperlukan untuk penelitian. Ditemukan dataset<sup>1</sup> yang telah disediakan dalam Paper "Bridging the gap between real-life data and simulated data by providing realistic fall dataset for evaluating camera-based fall detection algorithms". Dataset ini berupa video gerakan tubuh yang diambil menggunakan kamera 2 dimensi. Pembuatan dataset ini bertujuan untuk mendokumentasikan berbagai skenario dalam lingkungan yang realistik. Skenario jatuh yang berbeda dirancang untuk meniru insiden jatuh yang sebenarnya dengan seakurat mungkin.

Dataset tersebut terdiri dari 54 skenario jatuh dan 17 skenario tidak jatuh. Masing-masing skenario direkam dengan 5 sudut pandang sehingga menghasilkan 270 video jatuh dan 85 video tidak jatuh. Video skenario jauh memiliki rentang durasi 1 - 5 menit, sedangkan skenario tidak jatuh berkisar 10 - 30 menit. Seluruh video tersebut memiliki ukuran fram 800 x 480 piksel. Disediakan juga file meta yang berisi informasi rinci dari setiap video, misalnya

---

<sup>1</sup><https://iiw.kuleuven.be/onderzoek/advise/datasets#High%20Quality%20Fall%20Simulation%20Data>

setiap video jatuh disediakan informasi aktor, kondisi mulai jatuh, kondisi akhir jatuh dan deskripsi skenario. Terdapat juga penanda waktu mulai jatuh, tepat jatuh, berakhirnya jatuh dan durasi kejadian jatuh dalam satuan detik. Sama halnya dengan video tidak jatuh disediakan deskripsi skenario, durasi video dan aktor.

### 3.2.4 Prapemrosesan Data

Pada subbab ini akan dijelaskan secara rinci alur prapemrosesan dataset agar dapat diterima oleh model untuk melakukan pelatihan. Secara garis besar terdapat 3 tahap yaitu mengekstraksi citra yang diperlukan dari setiap video yang ada dalam dataset, lalu menggunakan Mediapipe Pose untuk mendekripsi 33 *pose landmark* manusia dari tiap citra, dan terakhir melakukan konversi data *pose landmark* tersebut ke dalam bentuk graf. Seluruh tahapan harus dilakukan agar model GCN dapat memproses dan mempelajari pola dan hubungan spasial tiap *landmark* sehingga dapat memprediksi posisi jatuh dan tidak jatuh pada manusia secara akurat.

#### 3.2.4.1 Ekstraksi Citra Dari Video

Prapemrosesan awal terhadap dataset dilakukan dengan mengekstrak 30 citra dari rentang detik di setiap video dalam dataset. Langkah ini bertujuan untuk mengurangi ukuran data yang akan diproses dan memastikan bahwa analisis dilakukan hanya pada *frame* citra yang relevan, baik kejadian jatuh maupun tidak jatuh. Mencapai tujuan tersebut, program Python ekstraksi *frame* perlu dikembangkan yang dapat secara otomatis berdasarkan informasi penanda waktu yang disediakan dalam file meta.

File meta ini berisi data penanda waktu yang menandakan waktu mulai dan berakhirnya kejadian yang khusus untuk video jatuh, namun sayangnya tidak disediakan penanda waktu untuk video tidak jatuh sehingga harus ditentukan secara manual penanda waktu mulai dan berakhir suatu kejadian dalam video

tidak jatuh tersebut. Data ini dapat digunakan untuk menentukan waktu yang tepat untuk ekstraksi citra. Program Python ini akan membaca file meta untuk memperoleh informasi penanda waktu jatuh dan kemudian memilih 30 *frame* citra secara acak di sepanjang periode tersebut.

Program ekstraksi citra akan dimulai dengan membuka dan membaca file meta dataset tersebut menggunakan *library Pandas*. Informasi penanda waktu mulai dan berakhirnya kejadian jatuh serta tidak jatuh sebagai acuan program untuk menentukan titik awal dan akhir kejadian di setiap video. Video jatuh berjumlah 270 dan video tidak jatuh sebanyak 85. Berdasarkan penanda waktu dalam file meta, program akan memilih 30 *frame* yang tersebar merata secara acak selama durasi kejadian jatuh untuk mendapatkan data yang representatif dari kejadian tersebut. Misalnya, jika durasi jatuh adalah 2 detik, maka 30 *frame* akan dipilih selama rentang waktu tersebut. Sementara itu, jumlah video jatuh dan tidak jatuh berbeda, sehingga perlu melakukan penyesuaian jumlah video dengan membagi masing-masing video dengan mengambil durasi tertentu berdasarkan aksi aktor, sehingga didapatkan total video tidak jatuh sebanyak 270.

Program kemudian akan mengekstrak *frame* sesuai dengan penanda waktu yang telah ditentukan menggunakan *library OpenCV*, mengidentifikasi posisi *frame* pada penanda waktu yang ditetapkan, dan menyimpan hasil ekstraksi citra dalam format yang sesuai seperti JPG. Berikut perhitungan sederhana untuk melihat jumlah frame citra yang akan didapatkan.

$$\begin{aligned}\text{Total citra} &= (\text{total video jatuh} \times 30) + (\text{total video tidak jatuh} \times 30) \\ &= (270 \times 30) + (270 \times 30) = 8.100 + 8.100 = 16.200\end{aligned}$$

berdasarkan perhitungan tersebut, diperkirakan total frame citra yang akan didapatkan sebanyak 16.200. Hasil ekstraksi *frame* ini akan disimpan dalam folder yang terorganisir dengan baik berdasarkan video dan informasi terkait, sehingga memudahkan analisis lebih lanjut. Contoh *frame* citra yang akan

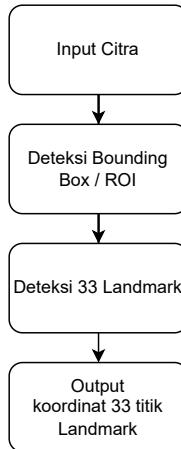
didapatkan dari video dataset dapat dilihat pada Gambar 3.2.



Gambar 3.2 contoh citra dari video jatuh.

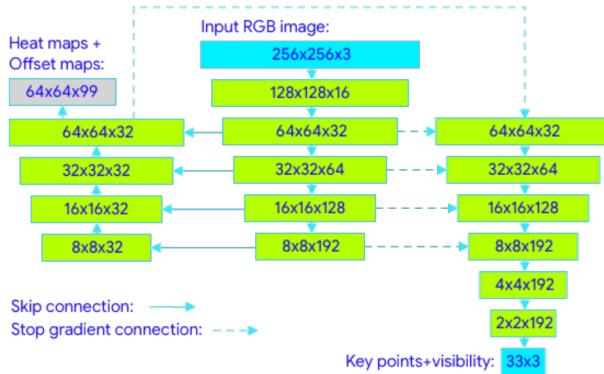
### 3.2.4.2 Deteksi 33 *Landmark* Dengan *Mediapipe Pose*

Proses selanjutnya yaitu seluruh citra tersebut akan diproses menggunakan *framework mediapipe* untuk mendapatkan data *pose landmark* bertipe data json. Tetapi sebelum itu setiap citra akan dilakukan prapemrosesan dengan konversi warna, jika citra tersebut awalnya dalam format BGR akan diubah menjadi RGB. Hal ini karena MediaPipe memerlukan input dalam format RGB. Kemudian citra tersebut diproses untuk mendeteksi *pose landmark*, alur proses deteksi pose landmark Mediapipe dapat dilihat pada Gambar 3.3.



Gambar 3.3 Alur proses deteksi pose landmark.

Terdapat 2 proses utama yang akan dilakukan dalam *Mediapipe Pose* dengan memanfaatkan *Blazepose*. Pertama, *Blazepose* akan mendeteksi keberadaan tubuh manusia dalam gambar dan menghasilkan output berupa *Bounding box* (kotak pembatas) yang mengelilingi tubuh serta titik referensi seperti pusat tubuh dan orientasi tubuh. *BlazePose* akan menentukan *Region of Interest* (ROI) yang berfokus pada area tubuh manusia dalam citra. Kedua, *BlazePose* akan mendeteksi 33 titik koordinat (*landmark*) dari tubuh manusia dalam ROI.



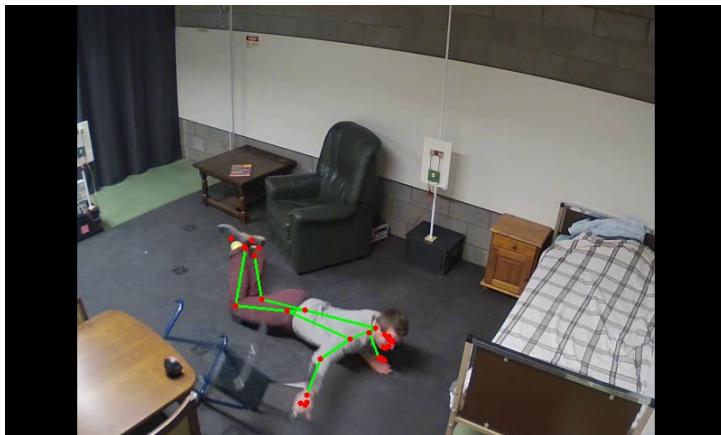
Gambar 3.4 Arsitektur jaringan BlazePose [26].

Arsitektur jaringan *BlazePose* menggunakan pendekatan kombinasi antara *heatmap*, *offset*, dan *regression* untuk meningkatkan akurasi dan efisiensi dalam mendekripsi pose tubuh, seperti yang terlihat pada Gambar 3.4. Proses dimulai dengan menerima input citra RGB dengan dimensi  $256 \times 256 \times 3$  yang telah diproses secara otomatis dalam *framework Mediapipe Pose*, tahapan awal dalam pipeline menyediakan proposal penyelarasan tubuh (*person alignment*), yang membantu model dalam memfokuskan perhatian pada area tubuh yang relevan. Kemudian lapisan *heatmap* hanya digunakan pada masa pelatihan model *BlazePose* untuk mengarahkan proses pembelajaran dengan memberikan peta probabilitas yang menunjukkan kemungkinan posisi setiap *landmark*, sementara *offset loss* digunakan untuk mengoreksi perbedaan antara prediksi dan lokasi yang sebenarnya [26].

Namun Pada saat ini, lapisan output yang terkait dengan *heatmap* dan *offset* dihapus karena model sudah menyelesaikan pelatihan dan masuk pada tahap inferensi, sehingga hanya jaringan *regression* yang digunakan untuk memperkirakan titik koordinat (*landmark*) dengan dimensi  $33 \times 3$ . Jaringan ini memanfaatkan *skip-connections* di seluruh tahap untuk menggabungkan fitur

tingkat tinggi dan rendah, memungkinkan model untuk menangkap informasi penting dari berbagai tingkat kompleksitas [26]. Berdasarkan struktur ini, *BlazePose* berhasil mendeteksi pose *landmark* tubuh manusia dengan akurasi tinggi, meskipun pada perangkat dengan sumber daya terbatas.

contoh *preview* hasil deteksi *landmark* dalam citra dapat dilihat pada Gambar 3.5.



Gambar 3.5 *Preview* hasil deteksi *landmark* terhadap citra.

Setiap *landmark* memiliki atribut x, y, dan z, di mana atribut x dan y menggambarkan posisi landmark terhadap sumbu simetri dalam citra, sementara atribut z menggambarkan kedalaman (3D) terhadap *midpoint* dengan titik ukurnya berada pada area pinggul. Jika semakin kecil nilai z maka seakan-akan *landmark* tersebut lebih dekat ke kamera. Seluruh *landmark* yang berhasil dideteksi dalam masing-masing citra akan disimpan dalam file *JSON* secara terpisah. Contoh data yang telah disimpan dapat dilihat pada Gambar 3.6.

```

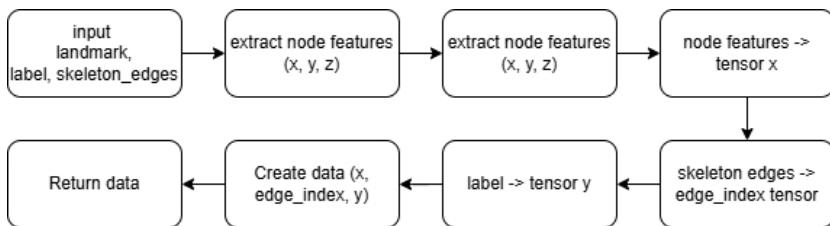
    "nose": {
      "x": 0.2670310437679291,
      "y": 0.5680254697799683,
      "z": -0.07775390893220901
    },
    "left_eye_inner": {
      "x": 0.2585166927337646,
      "y": 0.5588680505752563,
      "z": -0.0824333131313324
    },
    "left_eye": {
      "x": 0.2565276622772217,
      "y": 0.5596891641616821,
      "z": -0.0824654963951111
    },
    "left_eye_outer": {
      "x": 0.253826737408386963,
      "y": 0.5610659718513489,
      "z": -0.08251271396875381
    },
    "right_eye_inner": {
      "x": 0.26170666502571106,
      "y": 0.5586954355239868,
      "z": -0.10219831764698029
    },
    "right_eye": {
      "x": 0.261971116065979,
      "y": 0.5589655041694641,
      "z": -0.1022362709454102
    },
  },
}

```

Gambar 3.6 Contoh data *pose landmark*.

### 3.2.4.3 Konversi *Pose Landmark* Ke Graf

Setelah mendapatkan 33 *landmark* dari masing-masing citra, data tersebut harus dikonversi terlebih dahulu menjadi format graf, karena model GCN memerlukan representasi data dalam bentuk graf untuk memproses hubungan antar node. Konversi ini melibatkan transformasi data menjadi tensor yang mencakup informasi *node* dan *edge*. Kemudian data tersebut dapat digunakan untuk pelatihan dan prediksi dengan model GCN. Alur konversinya dapat dilihat pada Gambar 3.7.



Gambar 3.7 Alur konversi data *pose landmark* ke data graf.

Alur konversi *pose landmark* ke graf dimulai dengan menerima input berupa data *landmark*, label, dan *skeleton edges*. *Landmark* mewakili titik-titik pada tubuh manusia yang digunakan untuk menentukan pose, sementara *skeleton edges* menggambarkan hubungan atau koneksi antar *landmark* tersebut. Langkah pertama adalah mengekstrak fitur node untuk setiap *landmark*, yang mencakup koordinat 3D (x, y, z). Fitur ini kemudian diubah menjadi sebuah tensor yang merepresentasikan posisi *landmark* dalam ruang tiga dimensi. Setelah itu, *skeleton edges*, yang menunjukkan hubungan antar *landmark*, diubah menjadi tensor yang berisi indeks *edge* (hubungan antar node). Data untuk graf kemudian dibuat dengan menggabungkan tensor yang berisi fitur *node* (x), indeks *edge* (*edge\_index*), dan label (y) yang akan diprediksi. Label ini juga diubah menjadi tensor agar dapat diproses lebih lanjut. Setelah semua data dikonversi ke format tensor, data tersebut dikembalikan sebagai output yang siap digunakan dalam pemodelan graf atau analisis lebih lanjut, seperti yang terlihat pada Gambar 3.8. Dengan cara ini, data pose manusia dikonversi menjadi representasi graf yang siap diproses.

```

First item node features: tensor([[ 0.1320,  0.5301, -0.1914],
[ 0.1347,  0.5178, -0.1849],
[ 0.1368,  0.5159, -0.1849],
[ 0.1389,  0.5139, -0.1851],
[ 0.1279,  0.5196, -0.1828],
[ 0.1257,  0.5192, -0.1827],
[ 0.1234,  0.5188, -0.1827],
[ 0.1408,  0.5054, -0.1060],
[ 0.1197,  0.5126, -0.0964],
[ 0.1359,  0.5327, -0.1578],
[ 0.1280,  0.5348, -0.1553],
[ 0.1680,  0.5278, -0.0371],
[ 0.1987,  0.5620, -0.0167],
[ 0.1980,  0.5805, -0.0128],
[ 0.1171,  0.6389, -0.0161],
[ 0.2146,  0.6515, -0.0585],
[ 0.1289,  0.6490, -0.1189],
[ 0.2210,  0.6707, -0.0750],
[ 0.1202,  0.6576, -0.1383],
[ 0.2180,  0.6739, -0.0920],
[ 0.1280,  0.6528, -0.1582],
[ 0.2145,  0.6680, -0.0656],
[ 0.1219,  0.6500, -0.1298],
[ 0.1779,  0.6865,  0.0014],
[ 0.1396,  0.7054, -0.0012],
[ 0.1926,  0.7185, -0.2599],
[ 0.1440,  0.7476, -0.2070],
[ 0.2212,  0.8537, -0.2275],
[ 0.1691,  0.8804, -0.1621],
[ 0.2199,  0.8724, -0.2279],
[ 0.1757,  0.8924, -0.1607],
[ 0.2394,  0.8900, -0.3411],
[ 0.1722,  0.9270, -0.2662]]])
First item node features shape: torch.Size([33, 3])
First item edge index: tensor([[ 0,  0,  1,  2,  0,  0,  5,  7,  6,  8,  5,  6, 11, 13, 15, 15, 15,
19, 11, 23, 25, 27, 23, 24, 26, 28, 28],
[ 1,  2,  3,  4,  5,  6,  7,  9,  8, 10, 11, 12, 13, 15, 17, 19, 21, 19,
21, 23, 25, 27, 29, 31, 24, 26, 28, 30, 32]]])
First item edge index shape: torch.Size([2, 29])
First item label: tensor([0.])

```

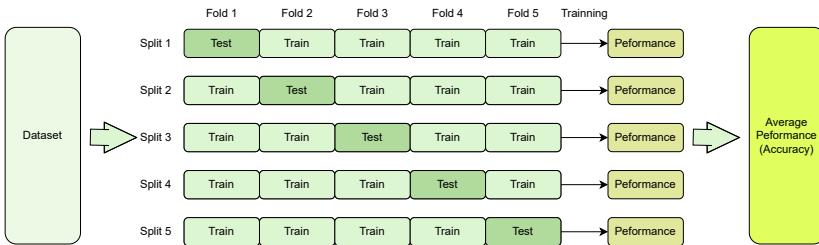
Gambar 3.8 Contoh tampilan data graf.

### 3.2.4.4 Pembagian *K-Fold* Pada Data

Tahap selanjutnya yaitu menerapkan teknik pembagian dataset *K-Fold*. Ini adalah teknik yang bertujuan untuk membagi dataset menjadi beberapa bagian yang disebut "*fold*", artinya teknik ini dapat membuat sebanyak  $k$  variasi subset data jatuh dan tidak jatuh untuk membantu generalisasi model sekaligus mengurangi potensi *overfitting*. Namun terdapat Konsekuensi dimana siklus pelatihan membutuhkan waktu dan sumber daya komputasi lebih besar dibandingkan skema pembagian data pelatihan konvensional. Selain itu terdapat pertimbangan lain, yaitu Jika nilai  $k$  terlalu besar, pembagian data menjadi lebih banyak lipatan akan mengakibatkan ukuran data di setiap lipatan menjadi lebih kecil, yang dapat menghasilkan estimasi varians yang lebih tinggi, terutama jika dataset tidak cukup besar. Sebaliknya, jika nilai  $k$  terlalu kecil, maka risiko bias terhadap estimasi kesalahan model dapat meningkat [27].

Penggunaan K-fold dengan  $K = 5$  telah terbukti menjadi pilihan yang efektif

dalam banyak aplikasi dan masih merupakan "standar yang baik" dalam banyak model. Hal ini memberi keseimbangan yang baik antara bias dan varians, memungkinkan efisiensi waktu komputasi yang lebih baik daripada pengaturan yang lebih tinggi [27]. Oleh karena itu, penelitian ini ditetapkan  $k = 5$ . Langkah pertama penerapan K-fold adalah mengelompokkan dataset *pose landmark* berdasarkan *prefix* folder—yang merepresentasikan sesi perekaman—agar data dari subjek yang sama tidak tercampur antara pelatihan dan validasi. Daftar *prefix* tersebut kemudian dibagi ke dalam lima *fold* proporsional memakai fungsi *KFold* dari *scikit-learn*.



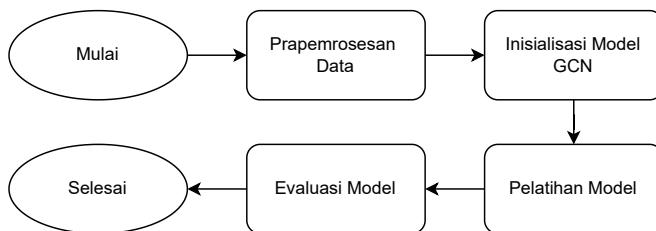
Gambar 3.9 Pembagian dataset menggunakan *5-Fold*

Ilustrasi pembagian dataset dengan teknik k-fold dapat dilihat pada Gambar 3.9. Keseluruhan dataset jatuh dan tidak jatuh dibagi menjadi 5 fold dengan tiap fold berisi 20% dari keseluruhan data. Lalu 5 fold tersebut disusun dengan variasi 5 Split yang masing-masing terdiri dari 80% data pelatihan dan 20% data evaluasi. Setiap 1 kali iterasi dilakukan proses pelatihan sebanyak 5 kali menggunakan *5 fold* yang berbeda, dengan setiap *fold* melatih model baru, artinya dalam pelatihan Kfold ini ada 5 model yang telah dilatih menggunakan 5 variasi pembagian dataset.

### 3.2.5 Perancangan Model GCN

Pada tahap ini dipaparkan rancangan pipeline pembangunan model secara keseluruhan. Perancangan memanfaatkan Graph Convolutional Network

(GCN) dengan arsitektur yang terinspirasi dari ResGCN untuk menangkap hubungan antar-node secara efektif melalui koneksi residual. Penyetelan model dilakukan dengan mengatur sejumlah hyperparameter kunci, meliputi jenis optimizer, learning rate, weight decay, batch size, dropout, jumlah epoch, ukuran hidden channel, serta pengaktifan residual untuk memperoleh kinerja optimal.

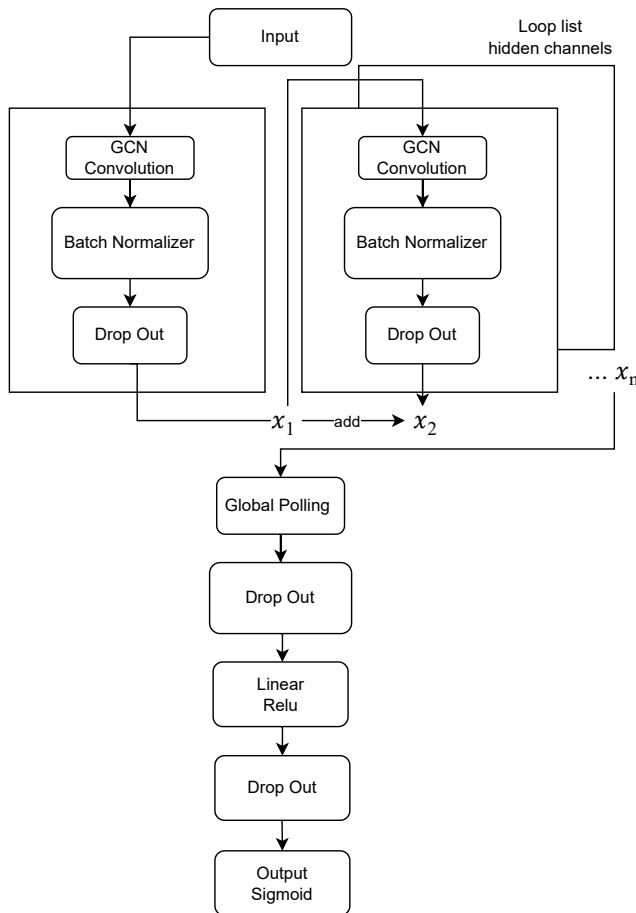


Gambar 3.10 Alur perancangan model

Berdasarkan rancangan pada Gambar 3.10, alur perancangan model dimulai dari dataset awal berupa kumpulan video berlabel jatuh dan tidak jatuh yang harus diproses terlebih dahulu agar dapat digunakan dalam pelatihan model. Tahap prapemrosesan dilakukan dengan mengekstrak 30 citra dari setiap klip waktu pada tiap video sesuai panduan pada *meta file*. Setiap citra kemudian diproses menggunakan inferensi *MediaPipe Pose* untuk memperoleh rangka tubuh (*skeleton*) yang memuat koordinat *node* serta struktur *edge*. Seluruh koordinat disimpan sebagai berkas JSON terpisah (satu JSON per citra). Dataset selanjutnya dibagi menggunakan skema *5-Fold* guna menilai kualitas pelatihan model. Sebelum diproses lebih lanjut, data koordinat tersebut dikonversi ke representasi graf sehingga dapat diterima oleh model GCN.

Model klasifikasi berbasis GCN dilatih untuk tugas klasifikasi biner "jatuh atau tidak jatuh" pada tiap graf. Skema 5-fold digunakan untuk membagi data latih dan validasi dengan pengelompokan citra per video guna mencegah kebocoran identitas antar video. Selama pelatihan, digunakan *Binary Cross-*

*Entropy (BCE) loss* sebagai fungsi kerugian. Untuk menilai konsistensi dan efisiensi pembelajaran, pelatihan diulang sebanyak lima kali dengan inisialisasi ulang dan masing-masing dijalankan pada *fold* yang berbeda. Evaluasi difokuskan pada metrik akurasi dan nilai *loss* untuk menilai kemampuan model dalam memprediksi posisi jatuh dan tidak jatuh pada setiap citra.



Gambar 3.11 Rancangan arsitektur model GCN.

Model GCN dirancang seperti yang terlihat pada Gambar 3.11. Lapisan pertama menerima input data *pose landmark* yang telah dikonversi menjadi

representasi graf. Lalu Pada lapisan GCN *Convolution*, data input melalui proses konvolusi dimana GCN bekerja dengan memperbarui fitur setiap node berdasarkan fitur tetangga sekitarnya, memungkinkan model untuk memanfaatkan struktur graf untuk membuat prediksi yang lebih baik. Setelah itu lapisan *Batch Normalizer* digunakan untuk menormalkan output dari lapisan sebelumnya agar membantu mengurangi pergeseran distribusi data selama pelatihan, sehingga mempercepat proses pelatihan dan meningkatkan stabilitas model.

Lapisan *Dropout* digunakan untuk mencegah overfitting dengan secara acak mematikan beberapa unit pada lapisan tersebut selama pelatihan. Ini membantu model untuk tidak terlalu bergantung pada fitur tertentu dan membuatnya lebih generalisasi. Lapisan *Global Pooling* digunakan untuk mengurangi dimensi data setelah proses konvolusi, dengan cara mengambil informasi dari seluruh graf dan merangkum fitur-fitur penting dalam representasi yang lebih ringkas. Selanjutnya ditambahkan lapisan *Linear* dengan aktivasi *Relu*, bertugas untuk menghubungkan informasi yang telah diproses sebelumnya ke dalam bentuk representasi yang dapat digunakan untuk klasifikasi atau prediksi. Lapisan terakhir, *fully connected* dengan fungsi aktivasi sigmoid, mengubah output model menjadi probabilitas yang dapat digunakan untuk klasifikasi dua kelas, yaitu jatuh atau tidak jatuh.

Pemilihan arsitektur ini dilakukan karena model ini mampu mencegah penurunan kinerja yang kerap muncul pada jaringan graf melalui penerapan *residual learning*, yakni *skip-connection* yang menjaga aliran gradien dan mencegah *oversmoothing* fitur node. Arsitektur ini mengekstraksi representasi tiap node melalui lapisan *GCNConv*, menstabilkannya dengan *Batch Normalization*, lalu merangkum seluruh node menjadi satu vektor graf memakai *global pooling*. *Skip-connection residual* dipakai untuk menahan gejala *oversmoothing* saat lapisan bertambah banyak, sebelum representasi akhir diproses *MLP head* untuk menghasilkan probabilitas prediksi.

### 3.2.6 Pelatihan Model

Setelah arsitektur model dibangun, langkah berikutnya adalah pelatihan model. Pada tahap ini, model GCN akan dilatih menggunakan dataset yang sudah diproses dan telah dibagi menggunakan teknik 5-Fold seperti pada Subbab 3.2.4. Proses pelatihan akan melibatkan pembelajaran pola-pola posisi tubuh dan aktivitas lainnya pada lansi berdasarkan data *pose landmark*. Model ini dilatih dengan menggunakan beberapa *hyperparameter* seperti yang dapat dilihat pada Tabel 3.3 di bawah ini.

*Optimizer* adalah algoritma yang digunakan untuk meminimalkan fungsi kerugian (*loss function*) dengan tujuan mencari input yang menghasilkan output dengan nilai maksimum atau minimum pada suatu fungsi. *Optimizer* memiliki peran dalam menentukan laju konvergensi dan stabilitas model menuju solusi yang optimal, serta mempengaruhi efisiensi proses pelatihan [28]. Salah satu faktor yang mempengaruhi kinerja *optimizer* adalah ukuran *batch size*, yang merujuk pada jumlah sampel pelatihan yang diproses dalam satu iterasi saat data melewati jaringan *neural*. Ukuran *batch size* yang sering digunakan dalam banyak model Saat ini adalah 32 dan 64 [29].

Selain itu, *learning rate* adalah parameter lain yang memainkan peran kunci dalam proses pelatihan, karena menentukan seberapa besar perubahan yang diterapkan pada bobot model setiap kali iterasi. Pemilihan *learning rate* yang optimal sangat penting; jika *learning rate* terlalu kecil, proses pelatihan bisa memakan waktu lebih lama untuk mencapai konvergensi. Sebaliknya, jika *learning rate* terlalu besar, model bisa melewatkkan titik minimum yang seharusnya dicapai, sehingga menyebabkan proses pembelajaran menjadi kurang efektif atau bahkan tidak stabil [30]. Oleh karena itu, pemilihan *learning rate* yang tepat sangat menentukan keberhasilan pelatihan model dan keberhasilan optimasi dalam mencapai solusi yang optimal. Pada penelitian ini digunakan *learning rate* bawaan, yaitu 0,001 untuk *optimizer Adam*.

### 3.2.7 Evaluasi Model

Setelah model dilatih, langkah selanjutnya adalah evaluasi performa model. Pada tahap ini, model diuji menggunakan data yang tidak digunakan dalam pelatihan yaitu 20% data uji yang disediakan dalam setiap *fold* untuk mengukur *accuracy*, *precision*, *recall*, dan *F-1 Score*. Hasil evaluasi ini akan difokuskan pada metriks *accuracy* untuk menentukan seberapa tepatnya model mampu mendeteksi posisi jatuh pada lansia. Pengukuran performa model secara keseluruhan, 5 nilai metriks *accuracy* dari hasil pelatihan masing-masing *fold* akan dirata-ratakan sebagai nilai performa akhir.

#### 3.2.7.1 Percobaan Utama dengan Pengaturan Hyperparamter Default

Subbab ini menyajikan nilai hyperparameter untuk percobaan utama agar menguji kemampuan model dalam mengklasifikasikan posisi jatuh dan tidak jatuh terhadap keseluruhan dataset. Pelatihan ini dilakukan menggunakan pengaturan hyperparameter *default* yang dirumuskan dari praktik umum di literatur dan dokumentasi *Pytorch*. Rincian nilai yang digunakan meliputi *Optimizer*, *Learning Rate*, *Weight Decay*, *Batch Size*, *DropOut*, *Epoch*, *Hidden Channel*, serta *Residual* dirangkum pada Tabel 3.1 berikut sebagai acuan eksperimen utama sebelum eksplorasi lebih lanjut.

Tabel 3.1 Nilai Default *hyperparameter* model.

No	Hyperparameter	Nilai Default	Keterangan
1	<i>Optimizer</i>	AdamW	disarankan oleh H. Vo et al pada tahun 2024 dalam paper berjudul <i>Exploring multiple optimization algorithms in transfer learning with EfficientNet models for agricultural insect classification</i> [31] karena secara konsisten menunjukkan kinerja yang lebih unggul dibandingkan dengan algoritma <i>optimizer</i> lain.
2	<i>Learning Rate</i>	1e-3	Tidak ada nilai default dari dokumentasi Pytorch, sehingga nilai default diambil dari paper <i>DeepGCNs: Making GCNs Go as Deep as CNNs</i> oleh G.Li et al pada tahun 2021 [32]

No	Hyperparameter	Nilai Default	Keterangan
3	weight decay	1e-5	Tidak ada nilai default dari dokumentasi Pytorch, sehingga diambil nilai default dari paper <i>Hybrid protein-ligand binding residue prediction with protein language models: Does the structure matter?</i> oleh H. Gamouh et al pada tahun 2025 [33]
4	Batch Size	32	Tidak ada nilai default dari dokumentasi Pytorch, sehingga diambil nilai default dari paper <i>Hybrid protein-ligand binding residue prediction with protein language models: Does the structure matter?</i> oleh H. Gamouh et al pada tahun 2025 [33]
5	DropOut	0.3	Tidak ada nilai default dari dokumentasi Pytorch, sehingga nilai default diambil dari paper <i>DeepGCNs: Making GCNs Go as Deep as CNNs</i> oleh G.Li et al pada tahun 2021 [32]

No	Hyperparameter	Nilai Default	Keterangan
6	epoch	100	Tidak ada nilai default dari dokumentasi Pytorch, sehingga nilai default diambil dari paper <i>DeepGCNs: Making GCNs Go as Deep as CNNs</i> oleh G.Li et al pada tahun 2021 [32]
7	Hidden Channel	[64, 32, 32, 32, 32]	Tidak ada nilai default dari dokumentasi Pytorch, sehingga nilai default diambil dari paper <i>DeepGCNs: Making GCNs Go as Deep as CNNs</i> oleh G.Li et al pada tahun 2021 [32]
8	Residual	True	Tidak ada nilai default dari dokumentasi Pytorch, sehingga nilai default diambil dari paper <i>DeepGCNs: Making GCNs Go as Deep as CNNs</i> oleh G.Li et al pada tahun 2021 [32] karena menghasilkan stabilitas pelatihan yang konsisten di semua kedalaman layer

Tabel 3.1 merangkum pengaturan dasar yang digunakan pada percobaan utama sebagai titik awal yang konsisten untuk klasifikasi jatuh/tidak jatuh berbasis GCN. AdamW dipilih sebagai *optimizer* karena stabil

pada pembelajaran graf skala kecil-menengah dan cenderung memberikan generalisasi yang baik. *Learning rate* 1e-3 dipakai sebagai nilai awal yang agresif namun masih stabil, sedangkan *weight decay* 1e-5 ditetapkan untuk memberikan regularisasi ringan guna menekan *overfitting* pada data *skeleton pose*. Batch size 32 menyeimbangkan stabilitas estimasi gradien dan keterbatasan memori, sementara dropout 0,3 diterapkan pada lapisan representasi untuk mengendalikan kompleksitas model. Proses pelatihan dijalankan selama 100 epoch namun tetap menerapkan *early stopping* untuk memberi kesempatan konvergensi tanpa overtraining, dengan konfigurasi *hidden channel* [64, 32, 32, 32, 32] sebagai kompromi antara kapasitas representasi dan efisiensi komputasi. Selain itu, *residual connection* diaktifkan (*True*) guna memperlancar aliran gradien pada tumpukan lapisan GCN dan meningkatkan stabilitas pelatihan. Konfigurasi ini menjadi acuan baku bagi eksperimen dasar sebelum dilakukan penelusuran lebih lanjut pada subbab 3.2.7.2.

### 3.2.7.2 Percobaan Tambahan Berdasarkan Studi Ablasi

Subbab ini memaparkan percobaan tambahan dalam bentuk studi ablati untuk menilai kontribusi dan sensitivitas komponen-komponen kunci pada model GCN. Studi ablati adalah prosedur eksperimen yang memvariasikan satu faktor pada satu waktu, sementara faktor lain ditahan pada konfigurasi *default* guna mengisolasi pengaruh faktor tersebut terhadap kinerja model. Tujuan utamanya yaitu mengidentifikasi komponen yang paling berpengaruh terhadap performa, memahami trade-off antara akurasi, serta memperoleh konfigurasi yang lebih *robust* terhadap variasi data. Pendekatan ini membantu menakar apakah perbaikan kinerja berasal dari penyesuaian tertentu (misalnya *optimizer* atau *learning rate*) atau sekadar efek samping dari kombinasi pengaturan yang tidak terkontrol.

Fokus ablati pada subbab ini mencakup lima *hyperparameter* inti yang

paling berpotensi memengaruhi dinamika pembelajaran, yakni *Optimizer*, *Learning Rate*, *Weight Decay*, *Batch Size*, serta *DropOut*. Setiap variasi diuji dengan menahan komponen lain pada pengaturan *default* yang telah didefinisikan pada percobaan utama, dan dievaluasi menggunakan metrik yang konsisten (misalnya akurasi, F1, dan *loss*) agar hasilnya terbandingkan secara adil. Rincian cakupan nilai untuk tiap *hyperparameter* yang akan diuji beserta skenario variasinya dirangkum pada Tabel 3.2 berikut sebagai acuan pelaksanaan dan pembacaan hasil studi ablasi.

Tabel 3.2 Variasi nilai *hyperparameter* model.

No	Hyperparameter	Variasi	Keterangan
1	Optimizer	1. AdamW 2. RMSProp 3. LION	Peneliti ingin membandingkan kinerja beberapa <i>optimizer</i> dengan membatasi pada 3 <i>optimizer</i> tersebut
2	Learning Rate	1. 0.01 atau 1e-2 2. 0.001 atau 1e-3 3. 0.0001 atau 1e-4	Penentuan variasi untuk melihat pengaruh kenaikan dan penurunan nilai hyperparameter terhadap akurasi pelatihan model
3	weight decay	1. 1e-4 2. 1e-5 3. 1e-6	Penentuan variasi untuk melihat pengaruh kenaikan dan penurunan nilai hyperparameter terhadap akurasi pelatihan model
4	Batch Size	1. 16 2. 32 3. 64	Penentuan variasi untuk melihat pengaruh kenaikan dan penurunan nilai hyperparameter terhadap akurasi pelatihan model
5	DropOut	1. 0,2 2. 0,3 3. 0,4	Penentuan variasi untuk melihat pengaruh kenaikan dan penurunan nilai hyperparameter terhadap akurasi pelatihan model

No	Hyperparameter	Variasi	Keterangan
-	Epoch	100	Hanya nilai default untuk pembatasan variasi
-	Hidden Channel	[64, 32, 32, 32, 32]	Hanya nilai default untuk pembatasan variasi
-	Residual	True	Hanya nilai default untuk pembatasan variasi

Tabel 3.2 merangkum ruang variasi yang digunakan pada studi ablati ini, di mana setiap percobaan hanya memodifikasi satu *hyperparameter* sementara yang lain dikunci pada konfigurasi *default*. Variasi *Optimizer* membandingkan AdamW, RMSProp, dan LION untuk menilai perbedaan dinamika gradien, stabilitas konvergensi, dan regularisasi implisit pada graf pose. *Learning rate* divariasikan pada skala 1e-2, 1e-3, 1e-4 guna mengamati sensitivitas kecepatan belajar terhadap akurasi dan kestabilan pelatihan. *Weight decay* diuji pada tiga nilai 1e-4, 1e-5, 1e-6 untuk melihat pengaruh regularisasi terhadap generalisasi. *Batch size* 16, 32, 64 dipilih untuk mengevaluasi kompromi antara noise gradien, memori, dan throughput, sedangkan *DropOut* 0,2; 0,3; 0,4 ditelaah sebagai kontrol kompleksitas representasi graf. Adapun epoch 100, hidden channel [64, 32, 32, 32, 32], dan residual *True* tidak divariasikan dan dipertahankan tetap sebagai pembatas ruang pencarian agar perbandingan tidak terlalu luas. Seluruh variasi dievaluasi menggunakan metrik akurasi, dan *loss* pada skema 1-fold, di mana pembagian 1-fold diambil dari fold yang memberikan pelatihan dengan akurasi terbaik pada percobaan utama. Hasil studi ablati ini diharapkan menjadi dasar untuk mengidentifikasi konfigurasi paling berpengaruh dan kandidat pengaturan terbaik.

### 3.3 Alat dan Bahan Tugas Akhir

Berbagai alat dan bahan yang akan digunakan dalam penelitian ini dijabarkan sebagai berikut.

#### 3.3.1 Alat

1. Code editor Microsoft Visual Studio Code sebagai tempat pembuatan program model.
2. Github untuk menampung dan melacak perkembangan hasil penelitian.
3. Python versi 3.10 dengan pustaka utama PyTorch versi 2.9.0 dan Mediapipe untuk membangun program utama model.
4. Jupyter Notebook untuk mempercepat eksperimen program hasil penelitian.

#### 3.3.2 Bahan

Bahan sangat penting karena digunakan sebagai acuan pengembangan penelitian. Bahan yang akan digunakan dalam penelitian ini hanya berupa kumpulan video simulasi jatuh dan aktivitas sehari hari dalam lingkungan panti jompo. Penjelasan detail mengenai Datasetnya dapat dilihat pada Subbab 3.2.3

### 3.4 Ilustrasi Metode Pengembangan/Pengukuran

Pada subbab ini akan dijabarkan berbagai perhitungan penting dalam proses pengembangan dan pelatihan model. Berikut adalah contoh perhitungan pada lapisan konvolusi Graph Convolutional Network (GCN) untuk mempermudah pemahaman proses konvolusi menggunakan data dari 4 landmark, dengan masing-masing memiliki 3 fitur koordinat (x, y, z). Fitur-fitur ini digunakan untuk membentuk matriks  $H^{(0)}$ , yang akan menjadi input pertama dalam model. Matriks ini merepresentasikan posisi landmark manusia dalam koordinat tiga dimensi (x, y, z) untuk setiap landmark yang dianalisis.

$$H^{(0)} = \begin{bmatrix} 0,5 & 0,5 & 0,5 \\ 1,0 & 1,0 & 1,0 \\ 1,5 & 1,5 & 1,5 \\ 2,0 & 2,0 & 2,0 \end{bmatrix}$$

Matriks adjacency menggambarkan hubungan antara landmark dalam graf. Misalnya, koneksi antar node (landmark) adalah sebagai berikut: node 1 terhubung dengan node 2 dan node 4; node 2 terhubung dengan node 1 dan node 3; node 3 terhubung dengan node 2 dan node 4; dan node 4 terhubung dengan node 1 dan node 3. Sementara itu, matriks derajat  $D$  adalah matriks diagonal yang menunjukkan jumlah hubungan yang dimiliki oleh setiap node. Dalam kasus ini, node 1 terhubung dengan dua node lainnya (derajat = 2), node 2 terhubung dengan dua node lainnya (derajat = 2), node 3 terhubung dengan dua node lainnya (derajat = 2), dan node 4 terhubung dengan dua node lainnya (derajat = 2).

$$A = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}, D = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix}$$

Pada lapisan pertama, bobot  $W^{(0)}$  dianggap sebagai matriks berukuran  $3 \times 3$ , yang setiap elemennya diinisialisasi dengan nilai acak. Matriks bobot ini digunakan untuk memproyeksikan fitur input dari matriks  $H^{(0)}$  ke dalam ruang fitur yang lebih tinggi. Proses ini sangat penting dalam jaringan saraf, karena bobot yang teroptimasi dengan baik akan memungkinkan model untuk mengenali pola-pola penting dalam data dan memperbaiki prediksi seiring dengan bertambahnya jumlah iterasi. Bobot yang diinisialisasi secara acak ini akan dipelajari dan disesuaikan selama proses pelatihan untuk mencapai hasil yang optimal.

$$W^{(0)} = \begin{bmatrix} 0,1 & 0,2 & 0,3 \\ 0,4 & 0,5 & 0,6 \\ 0,7 & 0,8 & 0,9 \end{bmatrix}$$

Normalisasi Matriks Adjacency dan Derajat dilakukan untuk memastikan bahwa informasi yang diteruskan melalui jaringan dihitung dengan cara yang seimbang dan terkontrol. Matriks derajat ter-normalisasi  $\hat{D}$  diperoleh dengan menambahkan matriks identitas  $I$  ke dalam matriks derajat  $D$ . Penambahan matriks identitas ini bertujuan untuk mempertahankan informasi asal node, sehingga setiap node dapat mempertahankan kontribusinya dalam proses propagasi informasi. Sementara itu, matriks adjacency ter-normalisasi  $\hat{A}$  dibentuk dengan menambahkan matriks identitas  $I$  ke dalam matriks adjacency  $A$ , yang mencerminkan hubungan antar node. Langkah ini sangat penting agar setiap node dapat memperhitungkan keterhubungannya dengan node lainnya, sekaligus mempertimbangkan pengaruh dari dirinya sendiri selama propagasi informasi dalam jaringan.

$$\hat{D} = D + I = \begin{bmatrix} 3 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 3 \end{bmatrix}, \hat{A} = A + I = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix}$$

Setelah normalisasi matriks adjacency dan derajat dilakukan, langkah berikutnya dalam proses propagasi fitur adalah perhitungan normalisasi lebih lanjut. Matriks normalisasi ini merupakan invers dari akar kuadrat matriks derajat yang telah dinormalisasi. Proses ini bertujuan untuk mengatur bobot hubungan antar node dalam graf, sehingga informasi yang dipropagasi antar node tidak terdistorsi oleh ketidakseimbangan jumlah koneksi setiap node. Dengan normalisasi ini, setiap node dapat memberikan kontribusi yang proporsional terhadap hasil akhir, berdasarkan hubungannya dengan node lainnya. Langkah ini sangat penting untuk memastikan bahwa kontribusi

informasi dari masing-masing node dihitung secara adil selama proses propagasi.

$$\hat{D}^{-1/2} = \begin{bmatrix} 0,577 & 0 & 0 & 0 \\ 0 & 0,577 & 0 & 0 \\ 0 & 0 & 0,577 & 0 \\ 0 & 0 & 0 & 0,577 \end{bmatrix}$$

Setelah langkah normalisasi selesai, proses selanjutnya adalah propagasi informasi melalui jaringan. Pada tahap ini, informasi dari matriks fitur  $H^{(0)}$  dipropagaskan dengan menggunakan rumus Rumus 2.3 berikut.

$$H^{(1)} = \sigma \left( \hat{D}^{-\frac{1}{2}} \hat{A} \hat{D}^{-\frac{1}{2}} H^{(0)} W^{(0)} \right)$$

Di sini, matriks  $H^{(0)}$ , yang berisi fitur input pada lapisan pertama, digabungkan dengan matriks ter-normalisasi  $\hat{D}^{-1/2}$  dan matriks adjacency  $\hat{A}$ . Pengalihan ini bertujuan untuk memperhitungkan hubungan antar node dalam graf dengan cara yang proporsional dan terkontrol. Selanjutnya, bobot  $W^{(0)}$  diterapkan pada hasil penggabungan tersebut untuk memproyeksikan fitur ke ruang yang lebih tinggi. Proses ini memungkinkan informasi yang ada pada node-node dalam graf untuk dipertukarkan, disesuaikan, dan diperbarui berdasarkan keterhubungan mereka, menghasilkan representasi yang lebih kompleks dan bermakna pada lapisan berikutnya.

Setelah propagasi selesai, hasil output dari proses ini adalah matriks  $H^{(1)}$ , yang berisi fitur yang telah diperbarui dan dipropagasi ke seluruh node. Matriks output ini, yang berukuran  $4 \times 3$ , menunjukkan fitur hasil propagasi untuk masing-masing node pada lapisan pertama, dengan nilai-nilai yang telah diperbarui sesuai dengan hubungan antar node dan bobot yang dipelajari. Matriks ini menggambarkan bagaimana informasi dari node-node dalam graf saling mempengaruhi satu sama lain dan diperbarui sesuai dengan struktur graf yang ada. Sebagai contoh, hasilnya adalah sebagai berikut.

$$H^{(1)} = \begin{bmatrix} 1,4 & 1,75 & 2,1 \\ 1,2 & 1,5 & 1,8 \\ 1,8 & 2,25 & 2,7 \\ 1,6 & 2,0 & 2,4 \end{bmatrix}$$

Selain perhitungan pada lapisan Graph Convolutional Network (GCN), terdapat juga perhitungan dalam rangkaian pelatihan model yang bertujuan untuk mengevaluasi kinerja model dalam mendeteksi posisi jatuh dan tidak jatuh pada lansia. Evaluasi kinerja dilakukan dengan mengukur akurasi deteksi terhadap dua kategori utama, yaitu posisi jatuh dan tidak jatuh. Tujuan dari evaluasi ini adalah untuk memastikan bahwa model dapat secara efektif membedakan antara kejadian jatuh dan aktivitas normal, sehingga dapat diandalkan dalam aplikasi pemantauan kesehatan lansia.

Prosedur evaluasi dimulai dengan pelatihan lima model GCN, di mana setiap model dilatih menggunakan data dari satu fold dalam skema 5-fold cross-validation. Selanjutnya, setiap model dievaluasi menggunakan data validation yang sudah disediakan dalam masing-masing fold untuk mengukur kinerjanya. Metrik evaluasi yang digunakan mencakup akurasi, presisi, recall, dan F1-score, yang dihitung berdasarkan hasil prediksi dan label sebenarnya dari data validation masing-masing fold. Setelah evaluasi, hasil dari kelima model tersebut dirata-rata untuk mendapatkan nilai kinerja keseluruhan. Terakhir, kinerja model GCN yang dikembangkan dibandingkan dengan metode deteksi jatuh lainnya yang ada dalam literatur untuk menilai keunggulan dan potensi perbaikan yang dapat dicapai melalui pendekatan berbasis GCN.

<b>Label Sebenarnya</b>	<b>Label Prediksi</b>	
	1 (Positive)	0 (Negative)
1 (Positive)	50	5
0 (Negative)	10	200

Tabel 3.3 contoh nilai *Confussion Matrix*

Penilaian kinerja model menggunakan metrik evaluasi pada Gambar 2.3, dapat diilustrasikan seperti pada Tabel 3.3 di mana hasil evaluasi untuk model pertama deteksi posisi jatuh pada lansia menunjukkan distribusi prediksi sebagai berikut.

1. 50 True Positives (TP), yaitu terdapat 50 kasus jatuh yang benar-benar terdeteksi sebagai jatuh.
2. 10 False Positives (FP), artinya ada 10 kasus tidak jatuh yang salah terdeteksi sebagai jatuh.
3. 5 False Negatives (FN), berarti terdapat 5 kasus jatuh yang salah terdeteksi sebagai tidak jatuh.
4. 200 True Negatives (TN), maksudnya ada 200 kasus tidak jatuh yang benar-benar terdeteksi sebagai tidak jatuh.

berdasarkan data tersebut, perhitungan evaluasi lanjutan dapat dilakukan sebagai berikut.

#### 1. Accuracy

$$\begin{aligned} \text{Accuracy} &= \frac{TP + TN}{TP + TN + FP + FN} \\ &= \frac{50 + 200}{50 + 200 + 10 + 5} = \frac{250}{265} = 0.8571 \quad \text{atau} \quad 85.71\% \end{aligned}$$

#### 2. Precision

$$\begin{aligned} \text{Precision} &= \frac{TP}{TP + FP} \\ &= \frac{50}{50 + 10} = 0,8333 \quad \text{atau} \quad 83,3\% \end{aligned}$$

#### 3. Recall

$$\begin{aligned} \text{Recall} &= \frac{TP}{TP + FN} \\ &= \frac{50}{50 + 5} \times 100\% = 0,9091 \quad \text{atau} \quad 90,91\% \end{aligned}$$

#### 4. F1-Score

$$\begin{aligned} F1 &= 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \\ &= 2 \times \frac{0,8333 \times 0,9091}{0,8333 + 0,9091} = 0,8696 \text{ atau } 86,96\% \end{aligned}$$

Hasil perhitungan confusion matrix menunjukkan bahwa model berhasil mengidentifikasi 50 kasus jatuh dengan benar (True Positives), sementara 10 kasus tidak jatuh salah terdeteksi sebagai jatuh (False Positives). Selain itu, terdapat 5 kasus jatuh yang tidak terdeteksi (False Negatives), dan 200 kasus tidak jatuh yang terkласifikasi dengan benar sebagai tidak jatuh (True Negatives). Berdasarkan perhitungan ini, akurasi model adalah 85,71%, yang menunjukkan bahwa model dapat mengklasifikasikan 85,71% dari seluruh data dengan benar. Presisi sebesar 83,33% mengindikasikan bahwa 83,33% dari semua prediksi jatuh yang dilakukan oleh model memang benar-benar jatuh, sedangkan recall sebesar 90,91% menunjukkan bahwa model berhasil mendeteksi 90,91% dari seluruh kejadian jatuh yang sebenarnya. F1-score yang dihitung sebesar 86,96% mencerminkan keseimbangan antara presisi dan recall, menunjukkan kinerja yang baik dalam mendeteksi jatuh dengan mempertimbangkan kedua metrik tersebut.

## BAB IV

# HASIL DAN PEMBAHASAN

### 4.1 Hasil Implementasi

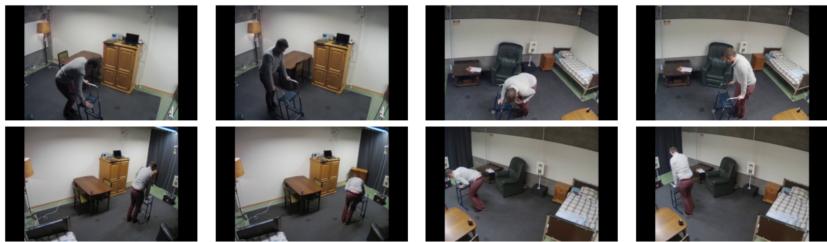
Penelitian ini dirancang melalui serangkaian tahapan sebagaimana ditunjukkan pada Gambar 3.1. Luaran utamanya berupa model Graph Convolutional Network (GCN) berbasis pose landmark yang mampu mendekripsi dan mengklasifikasikan posisi jatuh pada lansia. Uraian lebih rinci mengenai proses pengembangan serta hasil penelitian dipaparkan pada bagian berikut.

#### 4.1.1 Analisis Deskriptif

Dataset yang digunakan berasal dari paper “*Bridging the gap between real-life data and simulated data by providing realistic fall dataset for evaluating camera-based fall detection algorithms*”, berupa video gerakan tubuh yang direkam dengan kamera 2 dimensi untuk mendokumentasikan skenario jatuh dalam lingkungan realistik. Dataset ini mencakup 275 video jatuh dan 85 video tidak jatuh. Video jatuh berdurasi 1–5 menit, sedangkan video tidak jatuh berdurasi 10–30 menit, informasi lengkapnya dapat dilihat pada Subbab 3.2.3. Beberapa contoh frame dari video dataset dapat dilihat pada Gambar 4.1 dan Gambar 4.2.



Gambar 4.1 Contoh dataset jatuh



Gambar 4.2 Contoh dataset tidak jatuh

Distribusi data pada kategori jatuh dan tidak jatuh disajikan secara lebih spesifik sesuai dengan karakteristik masing-masing video. Pada data jatuh, distribusi difokuskan pada posisi awal aktor sebelum terjadinya insiden jatuh, sehingga dapat terlihat pola awal yang memicu terjadinya jatuh. Sementara itu, pada data tidak jatuh, distribusi dihitung berdasarkan posisi aktor sepanjang durasi video untuk merepresentasikan variasi aktivitas normal yang dilakukan tanpa adanya insiden jatuh. informasi lengkap nya dapat dilihat pada Tabel 4.1 dan 4.2.

Tabel 4.1 Penjabaran data video jatuh

No	Posisi awal	Total
1	Berdiri	120
2	Transisi berdiri–duduk	35
3	Transisi duduk–berdiri	10
4	Berlutut	30
5	Membungkuk	25
6	Duduk	50
<b>Total keseluruhan</b>		<b>270</b>

Terlihat pada Tabel 4.1 menyajikan rincian distribusi posisi awal aktor sebelum terjadinya insiden jatuh pada data video jatuh. Dari tabel tersebut terlihat bahwa mayoritas kejadian jatuh diawali dari posisi berdiri sebanyak 120 kali, diikuti dengan transisi berdiri–duduk sebanyak 35 kali, serta posisi duduk sebanyak 50 kali. Beberapa skenario lainnya juga melibatkan posisi berlutut (30 kali), membungkuk (25 kali), dan transisi duduk–berdiri (10 kali). Sementara

Tabel 4.2 Penjabaran data video tidak jatuh

No	Posisi sepanjang video (awal–akhir)	Total
1	Berdiri	65
2	Transisi berdiri–duduk	80
3	Transisi duduk–berdiri	35
4	Membungkuk	75
5	Duduk	15
<b>Total keseluruhan</b>		<b>270</b>

itu, Tabel 4.2 menggambarkan distribusi posisi aktor sepanjang durasi video tidak jatuh yang berfungsi sebagai pembanding. Pada skenario ini, posisi yang paling dominan adalah membungkuk (75 kali) dan transisi berdiri–duduk (80 kali), kemudian diikuti oleh berdiri (65 kali), transisi duduk–berdiri (35 kali), dan duduk (15 kali). Perbandingan kedua tabel ini menunjukkan adanya perbedaan kecenderungan posisi awal yang dapat menjadi indikator penting dalam membedakan aktivitas jatuh dan tidak jatuh.

#### 4.1.2 Prapemrosesan Dataset

Pada bagian ini akan dipaparkan hasil dari proses prapemrosesan dataset. Proses tersebut menghasilkan kumpulan citra yang berhasil diekstraksi dari video, kemudian mengekstrak 33 *pose landmark* manusia yang diperoleh melalui deteksi *Mediapipe Pose*. Selanjutnya, *landmark* tersebut dikonversi menjadi bentuk graf sehingga dapat merepresentasikan hubungan spasial antar titik tubuh secara lebih terstruktur. Hasil prapemrosesan ini menjadi tahap penting karena menentukan kualitas data yang akan digunakan oleh model GCN dalam membedakan pola posisi jatuh dan tidak jatuh.

#### 4.1.2.1 Ekstraksi Citra Dari video

```

def adjust_start_sec(start_sec, end_sec, percentage_base, scale_control):
    """
    Menggeser start_sec mendekati end_sec dengan pergeseran yang bergantung pada selisih waktu.
    Semakin besar selisih (end_sec - start_sec), semakin besar pergeseran start_sec.
    """
    # Hitung selisih waktu
    time_diff = end_sec - start_sec

    # Tentukan faktor pengali berdasarkan panjang rentang waktu (time_diff)
    # Rentang waktu kecil: pergeseran kecil. Rentang waktu besar: pergeseran besar.
    if time_diff > 2:
        scaling_factor = percentage_base * (time_diff / scale_control)

    # Pastikan scaling_factor tidak lebih kecil dari percentage_base (untuk rentang waktu yang sangat
    # kecil)scaling_factor = max(scaling_factor, percentage_base)
        scaling_factor = scaling_factor + percentage_base
    else:
        scaling_factor = percentage_base

    # Hitung seberapa besar perubahan yang perlu diterapkan
    shift = time_diff * (scaling_factor / 100)

    # Geser start_sec mendekati end_sec
    new_start_sec = start_sec + shift

    # Pastikan start_sec tidak lebih besar dari end_sec
    return min(new_start_sec, end_sec)

```

Gambar 4.3 Kode pergeseran *start\_time*

Potongan code pada Gambar 4.3 merupakan fungsi *adjust\_start\_sec* yang dipakai khusus untuk video berlabel jatuh guna “menggeser” waktu mulai (*start\_sec*) mendekati waktu akhir (*end\_sec*) secara proporsional terhadap panjang rentang kejadian. logika utamanya yaitu bila durasi interval besar, bagian awal yang kurang relevan dikurangi lebih banyak sehingga sampling frame lebih terfokus ke momen inti menjelang jatuh. Nilai balik dibatasi agar tidak melewati *end\_sec*.

Fungsi *extract\_random\_frames* pada Gambar 4.4 adalah inti ekstraksi. Fungsi ini membuka video, menghitung indeks frame berdasarkan *start\_secend\_sec*, lalu memilih secara acak sejumlah *num\_frames* dari rentang tersebut. Jika frame unik tidak cukup, program mengizinkan pengambilan ulang secara acak hingga kuota terpenuhi. Setiap frame terpilih disimpan sebagai citra di folder keluaran dengan penamaan yang menyertakan label kelas.

Ketika program ekstraksi ini dijalankan, *metadata* video dibaca dari file

```

def extract_random_frames(video_path, start_sec, end_sec, output_dir, label,
    num_frames):
    cap = cv2.VideoCapture(video_path)
    video_fps = cap.get(cv2.CAP_PROP_FPS)
    total_frames = int(cap.get(cv2.CAP_PROP_FRAME_COUNT))
    start_frame = int(start_sec * video_fps)
    end_frame = min(int(end_sec * video_fps), total_frames)

    selected_frames = []
    while len(selected_frames) < num_frames:
        available_frames = list(range(start_frame, end_frame))
        remaining_frames = list(set(available_frames) - set(selected_frames))
        list_frames = remaining_frames if remaining_frames else available_frames
        if list_frames:
            random_frame = np.random.choice(list_frames)
            selected_frames.append(random_frame)
        else:
            print("Tidak ada frame yang tersedia untuk dipilih."); exit(1)

    frame_count = 0
    for frame_idx in selected_frames:
        cap.set(cv2.CAP_PROP_POS_FRAMES, frame_idx)
        ret, frame = cap.read()
        if not ret: continue
        output_filename = f"{label}_{frame_count}.jpg"
        cv2.imwrite(os.path.join(output_dir, output_filename), frame)
        frame_count += 1
    cap.release()

```

Gambar 4.4 Kode ekstrak frame acak

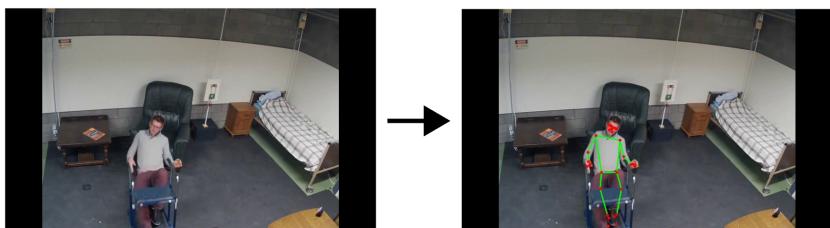
Excel yang berisi path video, waktu mulai/akhir, dan label. Waktu dikonversi ke detik, lalu khusus kelas jatuh diterapkan *adjust\_start\_sec* untuk memfokuskan rentang sampling. Terakhir, fungsi *extract\_random\_frames* dipanggil untuk mengekstraksi 30 citra per video, menghasilkan dataset gambar yang siap diproses *Mediapipe Pose* pada tahap berikutnya, seperti yang terlihat pada Gambar 4.1 dan 4.2.

#### 4.1.2.2 Deteksi 33 Landmark Dengan Mediapipe Pose

```
def extract_skeleton(image_path: str) -> Optional[Dict[str, Dict[str, float]]]:
    mp_pose = mp.solutions.pose
    try:
        image = cv2.imread(image_path)
        if image is None:
            print(f"Failed to load image: {image_path}")
            return None
        image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
        with mp_pose.Pose(static_image_mode=True, min_detection_confidence=0.5) as pose:
            results = pose.process(image_rgb)
            if results.pose_landmarks:
                landmarks = {}
                for idx, landmark in enumerate(results.pose_landmarks.landmark):
                    landmark_name = mp_pose.PoseLandmark(idx).name.lower()
                    landmarks[landmark_name] = {'x': landmark.x, 'y': landmark.y}
    except Exception as e:
        print(f"Error processing {image_path}: {str(e)}")
    return landmarks
```

Gambar 4.5 Kode ekstrak *skeleton* dari citra

Pada Gambar 4.5 terlihat Fungsi *extract\_skeleton* berperan sebagai inti deteksi pose. Fungsi ini membaca citra, mengonversi ruang warna dari BGR ke RGB agar sesuai dengan ekspektasi *MediaPipe*, lalu menjalankan *MediaPipe Pose* pada *static\_image\_mode=True* sehingga setiap citra diperlakukan independen tanpa temporal smoothing antar-frame—tepat untuk pipeline prapemrosesan berbasis citra. Ambang *min\_detection\_confidence=0.5* dipilih sebagai kompromi antara *recall* dan *precision* karena dianggap cukup rendah untuk menangkap pose pada citra yang tidak ideal, misal sedikit blur atau *occlusion* ringan, tetapi tetap membatasi *false positives*.



Gambar 4.6 Preview konversi citra ke *pose landmark*

Hasil *pose.process* mengembalikan *results.pose\_landmarks* yang berisi 33 titik anatomi terurut dalam bentuk koordinat ternormalisasi terhadap lebar dan tinggi citra pada rentang [0,1], dengan sumbu-x ke kanan dan sumbu-y ke bawah. normalisasi ini membuat representasi tidak bergantung pada resolusi asli dan memudahkan proyeksi ulang ke piksel saat visualisasi Contoh *preview* hasil ekstraksi pose landmark dapat dilihat pada Gambar .

#### 4.1.2.3 Konversi Pose Landmark Ke Graf

```
# Definisi struktur tulang (edges) antar-landmark MediaPipe (33
skeleton_edges = [
    (0, 1), (0, 2), (1, 3), (2, 4),
    (0, 5), (0, 6), (5, 7), (7, 9),
    (6, 8), (8, 10),
    (5, 11), (6, 12),
    (11, 13), (13, 15), (15, 17),
    (15, 19), (15, 21), (17, 19), (19, 21),
    (11, 23), (23, 25), (25, 27), (27, 29), (27, 31),
    (23, 24), (24, 26), (26, 28), (28, 30), (28, 32)
]
```

Gambar 4.7 Kode inisialisasi struktur *skeleton*

Pada Gambar 4.7 terlihat Daftar *skeleton.edges* yang mendefinisikan graf kerangka berdasarkan indeks *landmark MediaPipe* (0–32). Tiap pasangan (u, v) menyatakan sisi terarah (yang kemudian ditranspos menjadi bentuk [2, E] untuk PyG) antara dua sendi, mencakup koneksi utama tubuh atas (mata, bahu, siku, pergelangan) hingga tubuh bawah (pinggul, lutut, pergelangan kaki, tumit, ujung kaki). Daftar ini menyandikan pengetahuan anatomi ke dalam struktur graf sehingga GCN dapat mengeksploitasi kedekatan struktural antartitik.

```

def landmarks_to_graph(landmarks, label, skeleton_edges):
    # susun fitur node (x, y, z) berurutan sesuai 33 nama landmark
    node_features = []
    landmark_names = [
        "nose", "left_eye_inner", "left_eye", "left_eye_outer",
        "right_eye_inner", "right_eye", "right_eye_outer",
        "left_ear", "right_ear", "mouth_left", "mouth_right",
        "left_shoulder", "right_shoulder",
        "left_elbow", "right_elbow", "left_wrist", "right_wrist",
        "left_pinky", "right_pinky", "left_index", "right_index",
        "left_thumb", "right_thumb",
        "left_hip", "right_hip",
        "left_knee", "right_knee",
        "left_ankle", "right_ankle",
        "left_heel", "right_heel",
        "left_foot_index", "right_foot_index"
    ]
    for name in landmark_names:
        landmark = landmarks.get(name, {"x": 0.0, "y": 0.0, "z": 0.0})
        node_features.append([landmark['x'], landmark['y'], landmark['z']])
    x = torch.tensor(node_features, dtype=torch.float)           # [33, 3]
    edge_index = torch.tensor(skeleton_edges, dtype=torch.long).t().contiguous() # [2,
E] y = torch.tensor([label], dtype=torch.float)                 # [1]

    data = Data(x=x, edge_index=edge_index, y=y)
    return data

```

Gambar 4.8 Kode konversi *skeleton* ke graf

Fungsi *landmarks\_to\_graph* pada Gambar 4.8 mengubah hasil deteksi pose menjadi graf PyTorch Geometric (`torch_geometric.data.Data`) yang siap untuk pelatihan GCN. Pertama, urutan *landmark\_names* menegakkan indeks node yang konsisten (0–32), sehingga setiap sampel selalu menyusun fitur node [x, y, z] dalam urutan identik. Ini penting agar filter konvolusi graf melihat node yang sama di posisi yang sama antar-sampel. Tensor x berukuran [33, 3] memuat fitur koordinat 3 dimensi, sedangkan *edge\_index* adalah indeks sisi [2, E] hasil *t().contiguous()* agar kompatibel dengan kernel PyG. Label y dibuat sebagai tensor (1) bertipe float—konvensi umum untuk klasifikasi biner.

#### 4.1.2.4 Pembagian K-Fold Pada Dataset

Fungsi *generate\_kfold\_splits* Pada Gambar 4.9 membangun skema validasi silang berbasis grup untuk mencegah kebocoran data antar video. Prinsipnya yaitu nama folder di dalam data skeleton seperti diasumsikan memiliki prefix (bagian sebelum *\_*) yang merepresentasikan identitas subjek atau sesi rekaman.

```

def generate_kfold_splits(dataset_path, k=5, seed=42):
    """
    Generate k-fold cross-validation splits for the dataset.
    This ensures that files from the same subject/recording stay together in the same split.
    """
    set_seeds(seed)

    fall_category = 'fall'
    not_fall_category = 'not_fall'
    all_prefixes = set()

    fall_folders = os.listdir(os.path.join(dataset_path, fall_category))
    for folder in fall_folders:
        prefix = folder.split('_')[0]
        all_prefixes.add(prefix)

    not_fall_folders = os.listdir(os.path.join(dataset_path, not_fall_category))
    for folder in not_fall_folders:
        prefix = folder.split('_')[0]
        all_prefixes.add(prefix)

    all_prefixes = sorted(list(all_prefixes))

    prefix_to_files = {}
    for prefix in all_prefixes:
        matching_fall_folders = [f for f in fall_folders if f.split('_')[0] == prefix]
        matching_not_fall_folders = [f for f in not_fall_folders if f.split('_')[0] == prefix]
        all_files = []
        for folder in matching_fall_folders:
            folder_path = os.path.join(dataset_path, fall_category, folder)
            all_files.extend(glob(os.path.join(folder_path, "*.json")))
        for folder in matching_not_fall_folders:
            folder_path = os.path.join(dataset_path, not_fall_category, folder)
            all_files.extend(glob(os.path.join(folder_path, "*.json")))
        prefix_to_files[prefix] = all_files

    kf = KFold(n_splits=k, shuffle=True, random_state=seed)
    fold_splits = []
    prefix_list = list(all_prefixes)

    for fold, (train_idx, val_idx) in enumerate(kf.split(prefix_list)):
        train_prefixes = [prefix_list[i] for i in train_idx]
        val_prefixes = [prefix_list[i] for i in val_idx]
        train_files, val_files = [], []
        for p in train_prefixes: train_files.extend(prefix_to_files[p])
        for p in val_prefixes: val_files.extend(prefix_to_files[p])

        # sanity check overlap
        overlap = set(train_files).intersection(set(val_files))
        if overlap:
            print(f"WARNING: Found {len(overlap)} overlapping files between train and validation!")
            for file in overlap: print(f" Overlapping file: {file}")

        fold_splits.append({
            'fold': fold + 1,
            'train_prefixes': train_prefixes,
            'val_prefixes': val_prefixes,
            'train_files': train_files,
            'val_files': val_files
        })
    return fold_splits

```

Gambar 4.9 Kode membuat pembagian K-Fold

Fungsi ini mengumpulkan seluruh prefix dari kedua kategori, memetakan setiap prefix ke semua berkas .json (hasil ekstraksi *landmark*), lalu menjalankan pembagian K-Fold di level prefix. Dengan begitu, seluruh file milik satu prefix akan selalu jatuh ke satu split (train atau val), sehingga model tidak melihat data yang sama di dua subset berbeda. Prosedur juga menyertakan *sanity check* untuk mendeteksi tumpang tindih file. Jika ada overlap, peringatan dicetak agar peneliti dapat meninjau struktur data. Hasil akhirnya adalah daftar split berisi indeks fold dan daftar file train/val untuk setiap fold.

Tabel 4.3 Hasil penbagian E-Fold pada dataset per folder

Iterasi	K-Fold per Folder					Data Train	Data Validation
	1	2	3	4	5		
1	100	100	100	100	99	6,969	2,531
2	100	100	100	100	99	8,151	1,349
3	100	100	100	100	99	7,159	2,341
4	100	100	100	100	99	7,922	1,578
5	100	100	100	100	99	7,799	1,701
Total Keseluruhan						9,500	

Pada Tabel 4.3 ditunjukkan skema 5-fold cross-validation berbasis folder (bukan per file) untuk mencegah kebocoran data antar set. Pada tiap iterasi (1–5), satu kelompok folder dijadikan validasi (sel berwarna), sementara empat kelompok lainnya dipakai untuk pelatihan. Total terdapat 499 folder pada dataset, sehingga dibagi menjadi 5 kelompok: empat kelompok berisi 100 folder dan satu kelompok berisi 99 folder—itulah sebabnya kolom “K-Fold per Folder” berisi angka 100 dan 99. Meski jumlah folder per kelompok tetap, kolom Data Train dan Data Validation dapat bervariasi antar iterasi karena tiap folder menyimpan jumlah file yang berbeda. Dengan rancangan ini, setiap folder tepat sekali menjadi validasi dan empat kali masuk pelatihan, memastikan tidak ada konten dari folder yang sama muncul di train dan validasi secara bersamaan (no leakage).

### 4.1.3 Perancangan Model Klasifikasi

```

class ImprovedGCN(torch.nn.Module):
    def __init__(self,
                 num_node_features: int,
                 hidden_channels: int | list,
                 num_classes: int = 1,
                 dropout_rate: float = 0.3,
                 pool_type: str = 'mean',
                 residual: bool = True,
                 seed: int = 42):
        super(ImprovedGCN, self).__init__()
        torch.manual_seed(seed)

        if isinstance(hidden_channels, int):
            hidden_channels = [hidden_channels, hidden_channels]

        self.convs = torch.nn.ModuleList()
        self.batch_norms = torch.nn.ModuleList()
        self.residual = residual

        self.convs.append(GCNCConv(num_node_features, hidden_channels[0]))
        self.batch_norms.append(BatchNorm1d(hidden_channels[0]))

        for i in range(len(hidden_channels) - 1):
            self.convs.append(GCNCConv(hidden_channels[i], hidden_channels[i + 1]))
            self.batch_norms.append(BatchNorm1d(hidden_channels[i + 1]))

        self.dropout = Dropout(p=dropout_rate)
        self.final_lin1 = Linear(hidden_channels[-1], hidden_channels[-1] // 2)
        self.final_lin2 = Linear(hidden_channels[-1] // 2, num_classes)

        if pool_type == 'max':
            self.pool = global_max_pool
        elif pool_type == 'add':
            self.pool = global_add_pool
        else:
            self.pool = global_mean_pool
    
```

Gambar 4.10 Kode inisialisasi arsitektur model GCN

Program inisialisasi pada Gambar 4.10 membangun arsitektur GCN. Setiap lapis konvolusi graf (GCNCConv) diikuti batch normalization (BatchNorm1d) untuk menstabilkan distribusi aktivasi pada domain graf. Mekanisme dropout disiapkan untuk regularisasi, sementara bagian klasifikasi menggunakan MLP dua lapis (final.lin1 dan final.lin2) berfungsi memetakan *embedding global* ke ruang kelas. Pemilihan fungsi *global pooling* bersifat parametrik (mean, max, atau add) untuk mereduksi representasi tingkat-node menjadi vektor

tingkat-graf, sehingga model dapat menyesuaikan dengan karakteristik data. Penyelarasan seed pada awal konstruktor menjaga konsistensi inisialisasi bobot selama eksperimen.

```
def forward(self, x, edge_index, batch):
    previous = None
    for i, (conv, bn) in enumerate(zip(self.convs, self.batch_norms)):
        if i > 0 and self.residual:
            previous = x
        x = conv(x, edge_index)
        x = bn(x)
        x = F.relu(x)
        x = self.dropout(x)
        if i > 0 and self.residual and x.size(-1) == previous.size(-1):+ previous
        x = self.pool(x, batch)

    x = self.dropout(x)
    x = F.relu(self.final_lin1(x))
    x = self.dropout(x)
    x = self.final_lin2(x)

    if self.final_lin2.out_features == 1:
        return torch.sigmoid(x)
    else:
        return F.log_softmax(x, dim=-1)
```

Gambar 4.11 Kode forward propagation model GCN

Fungsi *forward* pada Gambar 4.11 merealisasikan alur inferensi dari tingkat-node hingga prediksi kelas. Setiap blok konvolusi graf menggabungkan penyebaran informasi topologis melalui *GCNConv*, normalisasi aktivasi dengan *BatchNorm1d*, fungsi aktivasi nonlinier ReLU, dan dropout untuk mengurangi *overfitting*. Opsi *residual connection* diaktifkan secara kondisional ketika dimensi fitur kompatibel, sehingga gradien dapat mengalir lebih stabil pada jaringan yang lebih dalam dan mengurangi degradasi representasi. Setelah tumpukan konvolusi, *global pool* mengagregasi fitur node menjadi embedding graf berbasis vektor batch (yang menandai keanggotaan setiap node di graf *mini-batch*). Kepala MLP kemudian memproyeksikan embedding global ke

ruang output. Aktivasi keluaran *sigmoid* untuk klasifikasi biner dengan satu neuron keluaran.

## 4.2 Evaluasi Hasil

Pada subbab ini akan dijelaskan secara rinci bagaimana hasil pelatihan model menggunakan konfigurasi hyperparameter default dan konfigurasi hyperparameter studi aborsi.

### 4.2.1 Evaluasi Model dengan *Hyperparameter Default*

Tabel 4.4 Konfigurasi *hyperparameter* untuk pelatihan model GCN

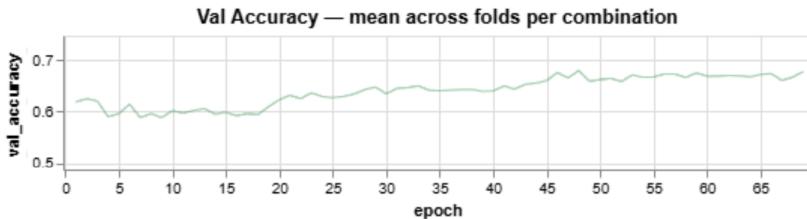
No	<i>Hyperparameter</i>	Nilai
1	<i>optimizers</i>	AdamW
2	<i>learning_rates</i>	1e-3
3	<i>weight_decays</i>	1e-5
4	<i>batch_sizes</i>	32
5	<i>dropout_rates</i>	0.3
6	<i>epochs</i>	100
7	<i>hidden_channels</i>	[64, 32, 32, 32, 32]
8	<i>residuals</i>	<i>True</i>

Berbagai nilai *Hyperparameter* pada Tabel 4.4 memperlihatkan pengaturan dasar yang digunakan pada seluruh eksperimen. Pelatihan dijalankan selama 100 epoch dengan ukuran batch 32. Skema optimisasi menggunakan AdamW dengan laju belajar  $1 \times 10^{-3}$  dan weight decay  $1 \times 10^{-5}$ , yang menjaga stabilitas pembaruan bobot sekaligus membantu menekan *overfitting*. Arsitektur GCN menerima 3 fitur per node ( $x, y, z$ ) dan menggunakan tumpukan hidden channels [64, 32, 32, 32, 32] yang dipadukan dengan dropout 0,3 sebagai regularisasi. Residual connection diaktifkan agar aliran gradien tetap stabil pada jaringan yang lebih dalam. Secara keseluruhan, konfigurasi ini merupakan kompromi yang baik: cukup ekspresif untuk mempelajari relasi spasial antartitik pose, namun tetap terkontrol dari sisi kompleksitas dan risiko *overfitting*.

Tabel 4.5 Hasil evaluasi per fold dan rerata

<b>Fold</b>	<b>Hasil Confusion Matrix</b>				<b>Metrik Evaluasi</b>			
	<b>TP</b>	<b>TN</b>	<b>FP</b>	<b>FN</b>	<b>Acc</b>	<b>Pre</b>	<b>Rec</b>	<b>F1</b>
1	558	1105	592	276	0,657	0,696	0,657	0,667
2	613	232	43	461	0,626	<b>0,812</b>	0,626	0,662
3	425	1165	299	452	0,679	0,671	0,679	0,672
4	703	419	138	318	<b>0,711</b>	0,742	<b>0,711</b>	<b>0,717</b>
5	501	594	163	443	0,644	0,674	0,644	0,641
<i>Rata-rata</i>					0,665	0,694	0,59	0,638

Tabel 4.5 menampilkan ringkasan confusion matrix dan metrik evaluasi utama untuk setiap *fold*. Kinerja model bervariasi dengan akurasi berkisar antara 0,644 hingga 0,711, dengan rerata 0,665. Rata-rata precision lebih tinggi (0,694) dibanding recall (0,590), yang mengarah pada F1-score rata-rata 0,638. Pola ini mengindikasikan bahwa model cenderung lebih fokus untuk meminimalkan false positives (FP) daripada menangkap semua kejadian positive (FN). Sebagai contoh, pada Fold 2, meskipun precision mencapai 0,812 berkat rendahnya FP (43), recall dan akurasi lebih rendah (0,626) karena banyak FN (461) yang menunjukkan bahwa model gagal mendekripsi sebagian besar kejadian jatuh. Sebaliknya, Fold 4 memberikan keseimbangan terbaik antara precision dan recall, dengan TP = 703, FN = 318, FP = 138, dan TN = 419, menghasilkan akurasi = 0,711 dan F1 = 0,717, yang menunjukkan bahwa fold ini memberikan hasil paling optimal dalam hal deteksi posisi jatuh. Variasi antara fold ini dapat dijelaskan oleh perbedaan distribusi kasus dalam setiap fold, yang mencerminkan pembagian data yang tidak selalu seimbang dalam kelompok K-Fold. Secara keseluruhan, model menunjukkan kecenderungan untuk menghindari false alarm (tinggi precision), meskipun dengan konsekuensi recall yang sedikit lebih rendah.



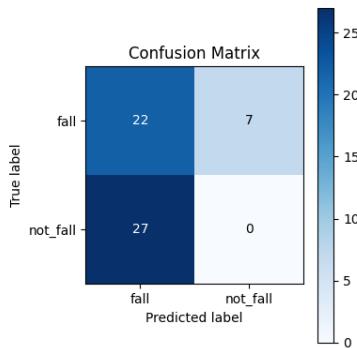
Gambar 4.12 akurasi hasil pelatihan model GCN dengan hyperparameter default

Kurva pada Gambar 4.12 menunjukkan rata-rata akurasi validasi lintas lima fold pada setiap epoch. Terlihat pola peningkatan yang relatif stabil dari kisaran awal  $\approx 0,60\text{--}0,62$  menuju  $\approx 0,68\text{--}0,69$  pada akhir pelatihan. Setelah fluktuasi kecil di awal, kurva cenderung *converge* secara bertahap mulai sekitar  $\sim$ epoch 35–40, lalu berfluktuasi tipis dengan tren naik hingga akhir. Tidak tampak indikasi kuat *overfitting* pada data validasi (tidak ada penurunan sistematis menjelang akhir), sehingga kombinasi *AdamW + learning rate*  $1 \times 10^{-3} + \text{dropout}$   $0,3 + \text{residual}$  dapat dinilai stabil pada konfigurasi ini. Dari sisi efisiensi, karena kenaikan setelah  $\sim$ epoch 50 relatif marginal, *early stopping* berbasis *patience* sebanyak 10 berpotensi memangkas waktu pelatihan tanpa mengorbankan kinerja. Secara keseluruhan, kurva ini mengonfirmasi temuan pada tabel: performa validasi konsisten pada kisaran  $0,66\text{--}0,69$  selaras dengan rerata akurasi 5-Fold (0.6647), serta menegaskan stabilitas pembelajaran model pada pengaturan hyperparameter default.

#### 4.2.2 Uji Model Terbaik Pada Data Primer

Pada penelitian ini, evaluasi model dilakukan dengan menggunakan bobot terbaik yang diperoleh berdasarkan akurasi hasil pelatihan model pada fold 4, epoch ke-2, yang menghasilkan akurasi validasi sebesar 0.71. Bobot tersebut kemudian digunakan untuk mengevaluasi kinerja model dengan hyperparameter default menggunakan dataset primer. Evaluasi akurasi model mencakup penjabaran metrix evaluasi serta pemaparan hasil prediksi terhadap setiap Citra

dalam dataset primer.



Gambar 4.13 Confusion matrix data test

Pada Gambar 4.13 terlihat bahwa hasil testing model klasifikasi posisi jatuh dan tidak jatuh pada data primer, model dapat memberikan prediksi yang tepat pada 22 gambar dengan label jatuh. Namun model tetap cenderung melakukan kesalahan prediksi ke arah "jatuh" pada data dengan label "tidak jatuh".

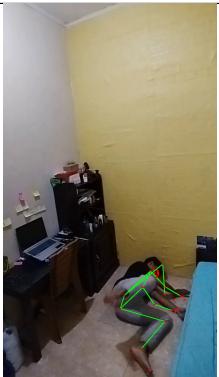
Tabel 4.6 Hasil evaluasi testing

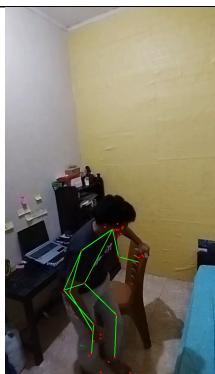
<b>Hasil confusion matrix</b>				<b>Metrik evaluasi</b>			
<b>TP</b>	<b>TN</b>	<b>FP</b>	<b>FN</b>	<b>acc</b>	<b>pre</b>	<b>rec</b>	<b>f1</b>
22	0	27	7	0,39	0,45	0,76	0,56

Metrik evaluasi model pada data testing dapat dilihat pada Tabel 4.6. Hasil metrik model dalam memprediksi data test ternyata kurang memuaskan. accuracy 0.39 menandakan kesalahan prediksi yang berat pada 1 kelas yaitu kelas jatuh dan sulit mengenali posisi tidak jatuh. Ditampilkan sebagian hasil prediksi data test menggunakan model yang telah dilatih sebelumnya pada tabel 4.7.

Tabel 4.7 Hasil Prediksi Model Akhir

No	Citra Input	Hasil Prediksi	Label Aktual
1.		tidak jatuh	jatuh
2.		jatuh	jatuh

No	Citra Input	Hasil Prediksi	Label Aktual
3.		jatuh	jatuh
4.		jatuh	tidak jatuh

No	Citra Input	Hasil	Label
		Prediksi	Aktual
5.		jatuh	tidak jatuh
6.		jatuh	tidak jatuh

#### 4.2.3 Evaluasi Studi Ablasi

Studi ablati pada penelitian ini bertujuan untuk memahami kontribusi masing-masing *hyperparameter* terhadap kinerja model *Graph Convolutional Network* (GCN) dalam tugas deteksi posisi jatuh berbasis *pose landmark* pada lansia. Lima *hyperparameter* yang dievaluasi yaitu *optimizer*, *learning rate*, *batch size*, *weight decay*, dan *dropout*. Pemilihan kelima faktor ini didasarkan pada pengaruh langsungnya terhadap stabilitas pelatihan,

kemampuan generalisasi, serta efisiensi konvergensi pada arsitektur GCN.

Seluruh eksperimen dilakukan pada susunan data fold ke-4 karena pada pelatihan default konfigurasi ini memberi akurasi tertinggi. Dengan demikian, perbandingan menjadi adil karena hanya satu *hyperparameter* diubah setiap kali (yang lain ditahan konstan). Setiap variasi dinilai terhadap *baseline* menggunakan confusion matrix yang berfokus pada metrik akurasi. Tujuannya adalah mengidentifikasi *hyperparameter* paling berpengaruh, sensitivitas model, serta merekomendasikan pengaturan optimal untuk eksperimen lanjutan dan implementasi akhir.

#### 4.2.3.1 Optimizer

Tabel 4.8 Hasil evaluasi studi ablati optimizer

<b>Optimizer</b>	<b>Hasil Confusion Matrix</b>				<b>Metrik Evaluasi</b>			
	<b>TP</b>	<b>TN</b>	<b>FP</b>	<b>FN</b>	<b>Acc</b>	<b>Pre</b>	<b>Rec</b>	<b>F1</b>
AdamW	1147	549	285	550	0.67	0.702	0.67	0.679
RMSProp	1222	531	303	475	<b>0.693</b>	0.711	0.693	0.699
LION	1158	523	311	539	0.664	0.691	0.664	0.672

Pada evaluasi optimizer yang terlihat pada Tabel 4.8, RMSProp memberikan kinerja terbaik dengan Acc 0.693, Pre 0.711, Rec 0.693, dan F1 0.699; ini tercermin dari TP tertinggi (1222) dan FN terendah (475), sehingga lebih banyak kejadian jatuh berhasil ditangkap meski spesifisitasnya moderat (0,636). AdamW berada di posisi kedua (Acc 0.670; F1 0.679) dengan FP terendah (285) dan spesifisitas tertinggi (0,658), artinya paling sedikit false alarm namun lebih banyak kejadian terlewat (FN 550, Rec 0.670). LION menghasilkan metrik terendah (Acc 0.664; F1 0.672) dengan FP dan FN sama-sama lebih tinggi (311 dan 539), menandakan konsistensi deteksi yang lebih lemah. Bedasarkan data tersebut, konfigurasi yang berfokus pada akurasi urutannya adalah RMSProp (0.693), kemudian AdamW (0.670), dan LION (0.664). RMSProp adalah pilihan terbaik, dengan AdamW sebagai alternatif

kedua dan LION tidak direkomendasikan untuk konfigurasi ini.

#### **4.2.3.2 Learning Rate**

Tabel 4.9 Hasil evaluasi studi ablasi learning rate

<b>Learning Rate</b>	<b>Hasil Confusion Matrix</b>				<b>Metrik Evaluasi</b>			
	<b>TP</b>	<b>TN</b>	<b>FP</b>	<b>FN</b>	<b>Acc</b>	<b>Pre</b>	<b>Rec</b>	<b>F1</b>
0,01	1469	427	407	228	<b>0,749</b>	0.74	0.749	0.74
0,001	1132	528	306	565	0.656	0.687	0.656	0.665
0,0001	1045	569	265	652	0.638	0.688	0.638	0.649

pada tabel 4.9 terlihat studi learning rate yang menunjukkan nilai 0,01 menghasilkan kinerja paling kuat di seluruh metrik utama, dengan nilai Acc 0,749, Pre 0,740, Rec 0,749, F1 0,740, didorong oleh TP tertinggi (1469) dan FN terendah (228) sehingga lebih banyak kejadian jatuh terdeteksi. Konsekuensinya, FP juga tertinggi (407) sehingga spesifisitas lebih rendah. Menurunkan laju belajar ke 0,001 menekan FP (306) dan menaikkan TN (528), namun FN melonjak (565) sehingga Acc turun ke 0,656 dan model menjadi lebih “hati-hati” namun sering melewatkannya kejadian. Dengan 0,0001, tren ini makin kuat: FP terendah (265) dan TN tertinggi (569) menunjukkan spesifisitas terbaik, tetapi FN meningkat tajam (652) sehingga Rec dan Acc melemah (0,638). Urutan nilai learning rate berdasarkan akurasi adalah 0,01 (0,749), 0,001 (0,656), 0,0001 (0,638).

#### **4.2.3.3 Batch Size**

Tabel 4.10 Hasil evaluasi studi ablasi batch size

<b>Batch Size</b>	<b>Hasil Confusion Matrix</b>				<b>Metrik Evaluasi</b>			
	<b>TP</b>	<b>TN</b>	<b>FP</b>	<b>FN</b>	<b>Acc</b>	<b>Pre</b>	<b>Rec</b>	<b>F1</b>
16	1149	531	303	548	0.664	0.693	0.664	0.672
32	1105	560	274	592	0.658	0.697	0.658	0.668
64	1218	521	313	479	<b>0.687</b>	0.705	0.687	0.693

Pada uji batch size dalam tabel 4.10, performa paling stabil justru muncul

saat ukuran batch diperbesar. Batch 64 mencatat TP tertinggi (1218) dan FN terendah (479), sehingga recall dan F1 ikut naik (0,687 dan 0,693). Konsekuensinya, FP sedikit naik (313) dan TN turun (521). Batch 16 memberi hasil menengah (Acc 0,664) dengan FP 303 dan FN 548. Sementara batch 32 terlihat paling “konservatif” terhadap *false alarm* (FP terendah 274, TN tertinggi 560), dikuti dengan TP terendah (1105) dan FN tertinggi (592), sehingga recall dan akurasinya ikut turun. Urutan nilai Batch size berdasarkan akurasi adalah batch 64 (0,687), batch 16 (0,664), batch 32 (0,658), sehingga dalam memaksimalkan akurasi pada konfigurasi ini, batch size 64 adalah pilihan terbaik.

#### 4.2.3.4 Weight Decay

Tabel 4.11 Hasil evaluasi studi aborsi Weight Decay

Weight Decay	Hasil Confusion Matrix				Metrik Evaluasi			
	TP	TN	FP	FN	Acc	Pre	Rec	F1
0,0001	1300	508	326	397	0.714	0.721	0.714	0.717
0,00001	1344	483	351	353	0.722	0.722	0.722	0.722
0,000001	1321	511	323	376	<b>0.724</b>	0.729	0.724	0.726

Pada studi weight decay dalam tabel 4.11, penurunan laju regularisasi memberi pola yang cukup konsisten. Nilai 0,000001 menghasilkan kinerja paling seimbang dengan Acc 0,724, Pre 0,729, Rec 0,724, dan F1 0,726; kombinasi TP 1321 / FN 376 menunjukkan deteksi kejadian jatuh yang baik tanpa terlalu banyak *false alarm* (FP 323, TN 511). Sedikit lebih besar, 0,00001 mendorong TP tertinggi (1344) dan FN terendah (353) sehingga recall naik, tetapi diimbangi oleh FP tertinggi (351) dan TN terendah (483) yang menekan spesifitas; akurasinya berada di tengah (0,722). Sementara 0,0001 membuat keseimbangan membaik di sisi FP/TN (326/508), namun FN meningkat (397) sehingga akurasi turun (0,714). Urutan nilai *weight decay* berdasarkan akurasi adalah adalah 0,000001 (0,724), 0,00001 (0,722), 0,0001 (0,714). Demi

memaksimalkan akurasi pada konfigurasi ini, weight decay 1e-6 adalah pilihan terbaik.

#### 4.2.3.5 Dropout

Tabel 4.12 Hasil evaluasi studi ablatasi Dropout

Dropout	Hasil Confusion Matrix				Metrik Evaluasi			
	TP	TN	FP	FN	Acc	Pre	Rec	F1
0,2	1178	557	277	519	<b>0.686</b>	0.713	0.686	0.693
0,3	1138	521	313	559	0.656	0.685	0.656	0.664
0,4	1102	559	275	595	0.656	0.696	0.656	0.666

Pada studi dropout dalam tabel 4.12, tingkat 0,2 memberi keseimbangan terbaik dengan metrik evaluasi Acc 0,686 (tertinggi), Pre 0,713, Rec 0,686, F1 0,693, ditopang TP 1178 dan FN 519 sehingga lebih banyak kejadian jatuh tertangkap dengan kenaikan *false alarm* yang relatif kecil (FP 277). Menaikkan dropout ke 0,3 menurunkan kinerja di hampir semua metrik (Acc 0,656, F1 0,664) dengan FN meningkat (559) dan FP juga tinggi (313). Pada 0,4, presisi sedikit membaik (0,696) dan *false alarm* paling rendah (FP 275, TN 559), tetapi banyak kejadian terlewat (FN 595), sehingga recall dan F1 turun. berdasarkan data tersebut dapat disimpulkan bahwa dropout 0,2 menghasilkan akurasi tertinggi (0,686), sedangkan 0,3 dan 0,4 setara di 0,656. jika fokus pada akurasi saja, nilai 0,2 adalah pilihan terbaik, dengan 0,4 hanya dipertimbangkan bila prioritasnya meminimalkan *false alarm*.

### 4.3 Perbandingan Hasil Hyperparameter Default dengan Studi Ablasi

Tabel 4.13 diperlihatkan perbedaan antara konfigurasi default dan hasil studi ablatasi untuk model GCN deteksi jatuh berbasis pose landmark. Perubahan utamanya berada di sisi optimisasi dan regularisasi, optimizer bergeser dari AdamW ke RMSProp yang pada eksperimen memberi konvergensi lebih stabil untuk data landmark, learning rate dinaikkan dari 1e-3 ke 1e-2 agar langkah pembaruan bobot lebih cepat, batch size diperbesar dari 32 ke 64 untuk

Tabel 4.13 Perbandingan Konfigurasi *hyperparameter* Default dan hasil Studi Ablasi

No	<i>Hyperparameter</i>	Default	Studi Ablasi
1	<i>optimizers</i>	AdamW	RMSProp
2	<i>learning rate</i>	1e-3	1e-2
3	<i>batch size</i>	32	64
4	<i>weight decay</i>	1e-5	1e-6
5	<i>dropout rate</i>	0.3	0,2
6	<i>hidden_channels</i>	[64, 32, 32, 32, 32]	[64, 32, 32, 32, 32]
7	<i>epochs</i>	100	100
8	<i>residuals</i>	<i>True</i>	<i>True</i>

menurunkan noise gradien, weight decay diturunkan dari 1e-5 ke 1e-6 guna mengurangi penalti berlebih pada bobot, dan dropout diturunkan dari 0,3 ke 0,2 agar kapasitas representasi tidak terlalu ditekan. Hasil Pelatihan ulang model menggunakan konfigurasi hyperparameter hasil studi ablati dapat dilihat pada tabel 4.14.

Tabel 4.14 Hasil evaluasi studi ablati per fold dan rerata

Fold	Hasil <i>Confusion Matrix</i>				Metrik Evaluasi			
	TP	TN	FP	FN	Acc	Pre	Rec	F1
1	458	1378	319	376	<b>0,725</b>	0,721	<b>0,725</b>	<b>0,723</b>
2	677	216	59	397	0,662	<b>0,804</b>	0,662	0,695
3	410	1181	283	467	0,68	0,67	0,68	0,67
4	719	410	147	302	0,716	0,741	0,716	0,721
5	500	573	184	444	0,631	0,656	0,631	0,628
<i>Rata-rata</i>				0,683	0,718	0,683	0,687	

Tabel 4.14 menunjukkan hasil kinerja model klasifikasi jatuh dan tidak jatuh pada skema 5-fold menggunakan kombinasi hyperparameter terbaik hasil studi ablati. Secara umum, performa berada di rentang akurasi 0,631–0,725 dengan rerata 0,683. Rata-rata precision 0,718 lebih tinggi daripada recall 0,683, sehingga F1 rata-rata mencapai 0,687. Fold 1 tampil paling kuat dengan akurasi 0,725 dan recall 0,725 (F1 0,723), ditopang TP yang relatif tinggi (458) meski FP masih ada (319). Fold 4 juga stabil (Acc 0,716, F1 0,721)

dengan keseimbangan FP/FN yang lebih baik (147/302). Sebaliknya, Fold 2 menunjukkan precision tertinggi 0,804 karena FP sangat rendah (59), namun banyak kasus positif terlewat (FN 397), sehingga akurasinya tidak menjadi yang terbaik (0,662). Fold 5 terendah (Acc 0,631), ditandai FN yang tinggi (444). Pola precision yang lebih besar dari recall konsisten di sebagian besar fold, menandakan model cenderung lebih jarang memberi alarm palsu, tetapi masih melewatkannya sebagian kejadian jatuh, kemungkinan dipengaruhi variasi distribusi sampel antar fold akibat pembagian berbasis folder.

Tabel 4.15 Perbandingan Hasil evaluasi konfigurasi hyperparamter default dan studi ablati

<b>konfigurasi <i>hyperparameter</i></b>	<b>Rerata Metrik Evaluasi</b>			
	<b>Acc</b>	<b>Pre</b>	<b>Rec</b>	<b>F1</b>
Default	0,665	0,694	0,59	0,638
Studi ablati	<b>0,683</b>	0,718	0,683	0,687

Tabel 4.15 membandingkan rerata metrik 5-fold antara konfigurasi default dan konfigurasi hasil studi ablati (nilai terbaik per hyperparameter). Secara keseluruhan, kombinasi dari studi ablati memberikan dorongan kinerja yang konsisten: akurasi naik dari 0,665 → 0,683 (peningkatan sebesar 0,018), presisi dari 0,694 → 0,718 (peningkatan sebesar 0,024), dan F1 dari 0,638 → 0,687 (peningkatan sebesar 0,049). Peningkatan terbesar terlihat pada recall, dari 0,590 → 0,683 (peningkatan sebesar 0,093), yang berarti model sedikit lebih jarang meloloskan kejadian jatuh (miss) dibanding konfigurasi awal. Karena presisi juga ikut naik, perbaikan ini tidak “dibayar” dengan lonjakan false alarm, justru prediksi positif menjadi lebih tepat sekaligus lebih lengkap. Dengan kata lain, set hyperparameter hasil studi ablati membuat model lebih andal di dua sisi, lebih akurat secara agregat dan lebih sensitif menangkap kejadian jatuh, tanpa mengorbankan ketepatan prediksi positif.

## **BAB V**

### **KESIMPULAN DAN SARAN**

#### **5.1 Kesimpulan**

Berdasarkan penelitian yang telah dilakukan, dapat dirangkum 2 poin utama kesimpulan sebagai berikut.

1. Penelitian ini berhasil merancang *pipeline end-to-end*, dimulai dari ekstraksi citra dari video berlabel jatuh/tidak jatuh, dengan sampling 30 frame pada rentang kejadian sebagai masukan ke tahap *pose estimation*. Dilakukan deteksi 33 *pose landmark* menggunakan MediaPipe Pose, lalu Pembagian dataset dengan menerapkan 5-Fold berbasis group/prefix folder untuk mencegah kebocoran data *train* dan *validation* dari folder yang sama, dan konversi koordinat landmark menjadi graf berfitur [x, y, z] dengan *skeleton edges* agar siap diproses GCN. Rangkaian ini disusul arsitektur GCN yang terinspirasi dari ResGCN (tumpukan GCNConv, BatchNorm, ReLU, Dropout, global pooling, kepala MLP, dan aktivasi sigmoid) dengan fungsi kerugian *BCELoss* untuk klasifikasi biner jatuh/tidak jatuh. Seluruh komponen tersebut membentuk prosedur yang sistematis dan dapat direplikasi untuk memetakan dinamika spasial pose menjadi keputusan klasifikasi.
2. Dalam Pelatihan model diterapkan 2 tahapan, Pertama melakukan percobaan utama menggunakan konfigurasi Hyperparameter default (Optimizer AdamW, Learning Rate 1e-3, batch size 32, weight decay 1e-5, dropout 0,3, dan residual aktif), evaluasi pelatihan 5-Fold menghasilkan rerata *accuracy* 0,665, *precision* 0,694, *recall* 0,590, dan F1 0,638. Rentang akurasi per-fold berada di 0,644–0,711. Ini menunjukkan kecenderungan model lebih menekan false positive (presisi relatif tinggi) namun masih meloloskan sebagian kasus jatuh (recall lebih

rendah). Percobaan kedua (tambahan) yaitu melakukan studi ablati dengan melatih model yang berfokus pada variasi 1 per 1 hyperparameter. Tujuan nya adalah melihat bagaimana pengaruh dari tiap hyperparameter terhadap kinerja pelatihan model agar dapat mengetahui dan memberi arahan peningkatan peforma model kedepannya. Setelah dilakukan studi ablati didapatkan berbagai nilai hyperparameter baru yang menunjukkan peningkatan kinerja model papda metrik akurasi antara lain Optimizer RMSProp, Learning Rate 1e-2, batch size 64, weight decay 1e-6 dan dropout 0,2 sementara arsitektur dan residual tetap. kombinasi hyperparamter ini meningkatkan rerata akurasi model sebesar 0,018 dari konfigurasi default ( $0,665 \rightarrow 0,683$ ), sekaligus menaikkan presisi sebesar 0,024 ( $0,694 \rightarrow 0,718$ ), recall naik sebesar 0,093 ( $0,590 \rightarrow 0,683$ ), dan F1 meningkat juga sebesar 0,049 ( $0,638 \rightarrow 0,687$ ). Peningkatan yang terjadi pada berbagai metrik, terutama akurasi menandakan model yang semakin baik dalam memberikan prediksi positif yaitu jatuh walaupun belum terlalu signifikan. Dengan demikian, konfigurasi hasil ablati dapat dikatakan lebih andal dibanding konfigurasi default untuk tugas deteksi posisi jatuh berbasis pose landmark.

## 5.2 Saran

Adapun saran yang dapat diberikan untuk penelitian lebih lanjut berdasarkan penelitian ini adalah sebagai berikut:

1. Perlu diterapkan multimodal dalam pengembangan penelitian ini agar mendapatkan informasi dan pola yang lebih bervariasi.
2. Perlu ditinjau kembali penggerjaan studi ablati dengan mengeksplorasi bagian bagian yang belum dicakup dalam penelitian ini.
3. Perlu dilakukan pendalaman terhadap dataset yang digunakan dalam penelitian ini dengan mengeksplorasi variasi posisi awal dalam dataset serta pemerataan komposisi variasi data agar pelatihan yang lebih baik.

4. Perlu dilakukan pendalaman terhadap arsitektur model agar dapat mengembangkan arsitektur model yang dapat memberikan kinerja yang lebih baik.

## DAFTAR PUSTAKA

- [1] I Gusti et al. *Hubungan Usia dan Jenis Kelamin dengan Resiko Jatuh Pada Lansia di Banjar Paang Tebel Peguyangan Kaja*. 2023.
- [2] Carlos J. Padilla Colón et al. “Muscle and Bone Mass Loss in the Elderly Population: Advances in diagnosis and treatment”. *Journal of Biomedicine* 3 (Mar. 2018), pp. 40–49.
- [3] George Forrest et al. “Falls on an Inpatient Rehabilitation Unit: Risk Assessment and Prevention”. *Rehabilitation Nursing* 37 (2 Mar. 2012), pp. 56–61. 0278-4807.
- [4] Cathleen S Colón-Emeric et al. “Risk Assessment and Prevention of Falls in Older Community-Dwelling Adults: A Review.” *JAMA* 331 (16 Apr. 2024), pp. 1397–1406. 1538-3598.
- [5] World Health Organization. *Falls: Key Facts*. Apr. 2021.
- [6] La Trobe University Australian Centre for Evidence Based Care (ACEBAC). “Falls – Standardised Care Process”. *Victorian Department of Health and Human Services* (Apr. 2023).
- [7] Neah Albasha et al. “Staff perspectives on fall prevention activities in long-term care facilities for older residents: ”Brief but often” staff education is key”. *PLoS ONE* 19 (9 Sept. 2024). 19326203.
- [8] Jane Fleming and Carol Brayne. “Inability to get up after falling, subsequent time on floor, and summoning help: Prospective cohort study in people over 90”. *BMJ* 337 (7681 Nov. 2008), pp. 1279–1282. 09598146.
- [9] Muhammad Mubashir, Ling Shao, and Luke Seed. “A survey on fall detection: Principles and approaches”. *Neurocomputing* 100 (Jan. 2013), pp. 144–152. 09252312.

- [10] Nalda Kresimo Negoro, Ema Utami, and Ainul Yaqin. “KLASIFIKASI DETEKSI PENGGUNAAN MASKER MENGGUNAKAN METODE CONVOLUTIONAL NEURAL NETWORK”. *JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)* 8 (2 May 2023), pp. 664–674. 2540-8984.
- [11] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. “Deep learning”. *Nature* 521 (7553 May 2015), pp. 436–444. 0028-0836.
- [12] Sagar Chhetri et al. “Deep learning for vision-based fall detection system: Enhanced optical dynamic flow”. *Computational Intelligence* 37 (1 Feb. 2021), pp. 578–595. 14678640.
- [13] Camillo Lugaressi et al. “MediaPipe: A Framework for Building Perception Pipelines” (June 2019).
- [14] Sandeep Singh Sengar, Abhishek Kumar, and Owen Singh. “Efficient Human Pose Estimation: Leveraging Advanced Techniques with MediaPipe” (June 2024).
- [15] Sarvesh P Raj et al. “Real-Time Body Pose Estimation Using OpenCV And Mediapipe”. *International Journal of Creative Research Thoughts* 12 (2024), pp. 2320–2882. 2320-2882.
- [16] Thomas N. Kipf and Max Welling. “Semi-Supervised Classification with Graph Convolutional Networks” (Sept. 2016).
- [17] Sania Zahan, Ghulam Mubashar Hassan, and Ajmal Mian. “Modeling Human Skeleton Joint Dynamics for Fall Detection” (Mar. 2025).
- [18] Xiaorui Zhang et al. “Dense Spatial-Temporal Graph Convolutional Network Based on Lightweight OpenPose for Detecting Falls”. *Computers, Materials and Continua* 77 (1 2023), pp. 47–61. 15462226.
- [19] Omar Reyad, Hazem Ibrahim Shehata, and Mohamed Esmail Karar. “Developed Fall Detection of Elderly Patients in Internet of Healthcare

- Things”. *Computers, Materials and Continua* 76 (2 Aug. 2023), pp. 1689–1700. 15462226.
- [20] Ali Raza et al. “Human Fall Detection from Sequences of Skeleton Features using Vision Transformer”. Vol. 5. Science and Technology Publications, Lda, 2023, pp. 591–598.
- [21] Vinayak Mali and Saurabh Jaiswal. “Pose-Based Fall Detection System: Efficient Monitoring on Standard CPUs” (Mar. 2025).
- [22] Xi Cai et al. “Vision-Based Fall Detection Using Dense Block with Multi-Channel Convolutional Fusion Strategy”. *IEEE Access* 9 (2021), pp. 18318–18325. 21693536.
- [23] Junkai Yang et al. “Fall Detection Method for Infrared Videos Based on Spatial-Temporal Graph Convolutional Network”. *Sensors* 24 (14 July 2024). 14248220.
- [24] Ahmad Said Agung Jatmiko and Ari Sapti Mei Leni. “Hubungan antara keseimbangan dengan risiko jatuh pada lansia penderita diabetes melitus tipe II”. *Journal Physical Therapy UNISA* 2 (2 May 2023), pp. 104–109. 2797-6246.
- [25] Asri Handayani Solihin Solihin, Dwi Maryani, and Biben Fikriana. “Hubungan Pengetahuan Dan Dukungan Keluarga Dengan Risiko Jatuh Pada Lansia”. *Jurnal Keperawatan 'Aisyiyah* 10 (2 Dec. 2023), pp. 131–140. 2355-6773.
- [26] Valentin Bazarevsky et al. *BlazePose: On-device Real-time Body Pose tracking*. 2020. arXiv: 2006.10204 [cs.CV].
- [27] E. Hadjisolomou et al. “Modelling freshwater eutrophication with limited limnological data using artificial neural networks”. *Water* 13 (11 2021), p. 1590.
- [28] S. Lee and T. Kim. “Impact of deep learning optimizers and hyperparameter tuning on the performance of bearing fault diagnosis”. *IEEE Access* 11 (2023), pp. 55046–55070.

- [29] R. Muralidhar, M. L. Demory, and M. M. Kesselman. “Exploring the impact of batch size on deep learning artificial intelligence models for malaria detection”. *Cureus* (2024).
- [30] Z. Xu et al. “Learning an adaptive learning rate schedule” (2019).
- [31] Hoang-Tu Vo et al. “Exploring multiple optimization algorithms in transfer learning with EfficientNet models for agricultural insect classification”. *CTU Journal of Innovation and Sustainable Development* 16 (Oct. 2024), pp. 35–41.
- [32] Guohao Li et al. “DeepGCNs: Making GCNs Go as Deep as CNNs” (May 2021).
- [33] Hamza Gamouh, Marian Novotný, and David Hoksza. “Hybrid protein–ligand binding residue prediction with protein language models: does the structure matter?” *Bioinformatics* 41.8 (July 2025). 1367-4811. eprint: <https://academic.oup.com/bioinformatics/article-pdf/41/8/btaf431/63902656/btaf431.pdf>.

## LAMPIRAN

### A Program *train\_gcn\_kfold*

```
def aggregate_histories_across_folds(histories):
    if not histories:
        return {"epoch": [], "val_accuracy": [], "train_accuracy": [], "val_f1": [], "val_loss": []}
    max_len = max(len(h) for h in histories)
    agg = {"epoch": [], "val_accuracy": [], "train_accuracy": [], "val_f1": [], "val_loss": []}
    for e in range(max_len):
        vals_vf1, vals_vloss, vals_vacc, vals_tacc = [], [], [], []
        for h in histories:
            if len(h) > e:
                vals_vacc.append(h[e]["val_accuracy"])
                vals_tacc.append(h[e]["train_accuracy"])
                vals_vf1.append(h[e]["val_f1"])
                vals_vloss.append(h[e]["val_loss"])
        if len(vals_vacc) == 0:
            continue
        agg["epoch"].append(e + 1)
        agg["val_accuracy"].append(float(np.mean(vals_vacc)))
        agg["train_accuracy"].append(float(np.mean(vals_tacc)))
        agg["val_f1"].append(float(np.mean(vals_vf1)))
        agg["val_loss"].append(float(np.mean(vals_vloss)))

    return agg
```

Gambar 0.1 aggregate histories across folds

```
def build_optimizer(name, params, lr, weight_decay, **extra):
    name = name.lower()
    if name == 'adamw':
        return torch.optim.AdamW(params, lr=lr, weight_decay=weight_decay, **extra)
    elif name == 'rmsprop':
        # momentum/alpha/eps bisa di-tune via extra jika perlu
        return torch.optim.RMSprop(params, lr=lr, weight_decay=weight_decay,
                                   momentum=extra.get('momentum', 0.0),
                                   alpha=extra.get('alpha', 0.99),
                                   eps=extra.get('eps', 1e-8),
                                   centered=extra.get('centered', False))
    elif name == 'lion':
        try:
            from lion_pytorch import Lion # pip install lion-pytorch
        except ImportError as e:
            raise ImportError("Lion belum terpasang. Jalankan: pip install lion-pytorch") from e
        return Lion(params, lr=lr, weight_decay=weight_decay,
                   betas=extra.get('betas', (0.9, 0.99)))
    else:
        raise ValueError(f"Optimizer '{name}' tidak dikenali.")
```

Gambar 0.2 build optimizer

```

def run_grid_search(config, device, wandb_sync=False):
    hyperparameter_combinations = itertools.product(
        config['optimizers'],
        config['batch_sizes'],
        config['learning_rates'],
        config['weight_decays'],
        config['dropout_rates'],
        config['residuals'],
    )

    best_val_accuracy = 0.0
    best_combination = None
    all_summaries = []

    aggregate_combo_curves = []

    for opt_name, batch_size, learning_rate, weight_decay, dropout_rate, residual in hyperparameter_combinations:
        learning_rate = round(learning_rate / 3, 4) if opt_name.lower() == 'lion' else learning_rate
        tag = f'{opt_name}_batch-size:{batch_size}_learning-rate:{learning_rate}_weight-decay:{weight_decay}_dropout-rate:{dropout_rate}_residual:{int(residual)}"'

        print(f"\nRunning with {tag}")

        current_config = config.copy()
        current_config['batch_size'] = batch_size
        current_config['learning_rate'] = learning_rate
        current_config['dropout_rate'] = dropout_rate
        current_config['residual'] = residual
        current_config['optimizer'] = opt_name
        current_config['weight_decay'] = weight_decay
        current_config['run_name'] = f"{config['run_name']}_{tag}"

        if wandb_sync and config.get("wandb_mode", "per_combo") == "per_combo":
            combo_run = wandb.init(
                entity=config.get("wandb_entity", None),
                project=config.get("wandb_project", "fall-detection"),
                name=current_config['run_name'],
                group=config['run_name'],
                config=current_config,
                reinit="finish_previous",
            )
            wandb.define_metric("epoch")
            wandb.define_metric("*", step_metric="epoch")
        else:
            combo_run = None

        fold_results = []
        best_confusion_matrices = []
        fold_histories = []

        for fold in range(1, 6):
            result = train_single_fold(fold, current_config, device, wandb_sync)
            fold_results.append(result)
            best_confusion_matrices.append(result['best_confusion_matrix'])
            fold_histories.append(result['epoch_history'])

    
```

Gambar 0.3 run grid search part 1

```

valid_cms = [cm for cm in best_confusion_matrices if cm is not None]
if len(valid_cms) == 0:
    print("WARNING: Tidak ada confusion matrix valid untuk kombinasi ini.")
    if combo_run is not None:
        combo_run.finish()
    continue
avg_best_confusion_matrix = np.mean(valid_cms, axis=0)

print("\n" + "="*50)
print("CROSS-VALIDATION SUMMARY")
print("="*50)

accuracies = [result['best_accuracy'] for result in fold_results]
f1_scores = [result['best_f1'] for result in fold_results]

mean_accuracy = float(np.mean(accuracies))
std_accuracy = float(np.std(accuracies))
mean_f1 = float(np.mean(f1_scores))
std_f1 = float(np.std(f1_scores))

print(f"Mean Accuracy across folds: {mean_accuracy:.4f} ± {std_accuracy:.4f}")
print(f"Mean F1 Score across folds: {mean_f1:.4f} ± {std_f1:.4f}")

print("\nBest Results by Fold:")
for result in fold_results:
    print(f"Fold {result['fold']}: Accuracy = {result['best_accuracy']:.4f}, F1 = {result['best_f1']:.4f}, Epoch {result['best_epoch']}")

print("\nAverage of Best Confusion Matrices:")
print(avg_best_confusion_matrix)

avg_tp, avg_fp, avg_fn, avg_tp = avg_best_confusion_matrix.ravel()
avg_accuracy = (avg_tp + avg_fn) / (avg_tp + avg_fn + avg_fp + avg_fn)
avg_precision = avg_tp / (avg_tp + avg_fp) if (avg_tp + avg_fp) > 0 else 0
avg_recall = avg_tp / (avg_tp + avg_fn) if (avg_tp + avg_fn) > 0 else 0
avg_f1 = 2 * (avg_precision * avg_recall) / (avg_precision + avg_recall) if (avg_precision + avg_recall) > 0
else 0

print(f"\nMetrics from Average Confusion Matrix:")
print(f"Accuracy: {avg_accuracy:.4f}")
print(f"Precision: {avg_precision:.4f}")
print(f"Recall: {avg_recall:.4f}")
print(f"F1 Score: {avg_f1:.4f}")

if wandb_sync:
    wandb.log(
        {
            f"{tag}/mean_accuracy": mean_accuracy,
            f"{tag}/mean_f1": mean_f1,
            f"{tag}/avg_cm_accuracy": float(avg_accuracy),
            f"{tag}/avg_cm_f1": float(avg_f1),
        }
    )

if mean_accuracy > best_val_accuracy:
    best_val_accuracy = mean_accuracy
    best_combination = {
        'optimizer': opt_name,
        'batch_size': batch_size,
        'learning_rate': learning_rate,
        'weight_decay': weight_decay,
        'dropout_rate': dropout_rate,
        'residual': residual,
        'tag': tag,
    }

print("\nBest Model Paths:")
for result in fold_results:
    print(f"Fold {result['fold']}:")
    for path in result['top_models']:
        if ".BEST" in path:
            print(f"- {path} (BEST)")
        else:
            print(f"- {path}")

np.save(f'best_avg_confusion_matrix_{current_config["run_name"]}.npy', avg_best_confusion_matrix)
print(f"\nSaved average best confusion matrix to: best_avg_confusion_matrix_{current_config['run_name']}.npy")

```

Gambar 0.4 run grid search part 2

```

agg_curve = _aggregate_histories_across_folds(fold_histories)
aggregate_combo_curves.append({"series": tag, **agg_curve})

if combo_run is not None:
    combo_run.finish()

combo_summary = {
    'tag': tag,
    'mean_accuracy': mean_accuracy,
    'std_accuracy': std_accuracy,
    'mean_f1': mean_f1,
    'std_f1': std_f1,
    'avg_cm_accuracy': float(avg_accuracy),
    'avg_cm_f1': float(avg_f1),
}
all_summaries.append(combo_summary)

if best_combination:
    print(f"\nBest combination: {best_combination}")
    print(f"Best validation accuracy: {best_val_accuracy:.4f}")
else:
    print("No best combination found.")

if wandb_sync and aggregate_combo_curves:
    agg_run = wandb.init(
        entity=config.get("wandb_entity", None),
        project=config.get("wandb_project", "fall-detection"),
        name=f"[{config['run_name']}]_AGG",
        config={"parent": config["run_name"]},
        reinit="finish_previous",
    )
    table_acc = wandb.Table(columns=["epoch", "val_accuracy", "series"])
    for combo in aggregate_combo_curves:
        for e, v in zip(combo["epoch"], combo["val_accuracy"]):
            table_acc.add_data(int(e), float(v), combo["series"])
    plot_acc = wandb.plot.line(
        table_acc, x="epoch", y="val_accuracy", title="Val Accuracy - mean across folds per combination",
        stroke="series"
    )
    wandb.log({"aggregate/val_accuracy": plot_acc})

    table_train_acc = wandb.Table(columns=["epoch", "train_accuracy", "series"])
    for combo in aggregate_combo_curves:
        for e, v in zip(combo["epoch"], combo["train_accuracy"]):
            table_train_acc.add_data(int(e), float(v), combo["series"])
    if len(table_train_acc.data) > 0:
        plot_train_acc = wandb.plot.line(
            table_train_acc, x="epoch", y="train_accuracy", title="Train Accuracy - mean across folds per combination", stroke="series"
        )
        wandb.log({"aggregate/train_accuracy": plot_train_acc})

    # Table untuk val_f1
    table_f1 = wandb.Table(columns=["epoch", "val_f1", "series"])
    for combo in aggregate_combo_curves:
        for e, v in zip(combo["epoch"], combo["val_f1"]):
            table_f1.add_data(int(e), float(v), combo["series"])
    plot_f1 = wandb.plot.line(
        table_f1, x="epoch", y="val_f1", title="Val F1 - mean across folds per combination", stroke="series"
    )
    wandb.log({"aggregate/val_f1": plot_f1})

    # Table untuk val_loss
    table_loss = wandb.Table(columns=["epoch", "val_loss", "series"])
    for combo in aggregate_combo_curves:
        for e, v in zip(combo["epoch"], combo["val_loss"]):
            table_loss.add_data(int(e), float(v), combo["series"])
    plot_loss = wandb.plot.line(
        table_loss, x="epoch", y="val_loss", title="Val Loss - mean across folds per combination", stroke="series"
    )
    wandb.log({"aggregate/val_loss": plot_loss})

    agg_run.finish()

return {
    'best_val_accuracy': best_val_accuracy,
    'best_combination': best_combination,
    'all_summaries': all_summaries,
}

```

Gambar 0.5 run grid search part 3

```

def save_checkpoint(model, optimizer, scheduler, epoch, config, path):
    os.makedirs(os.path.dirname(path), exist_ok=True)
    torch.save(
        {
            "model_state": model.state_dict(),
            "optimizer_state": optimizer.state_dict(),
            "scheduler_state": scheduler.state_dict() if scheduler is not None else
None,
            "epoch": epoch,
            "config": config,
        },
        path,
    )

```

Gambar 0.6 save checkpoint

```

def save_model(model, run_name, fold, epoch, val_score, save_dir='pth'):
    save_dir = os.path.join(os.getcwd(), save_dir)
    os.makedirs(save_dir, exist_ok=True)
    path = os.path.join(save_dir, f'{run_name}_{fold}_{epoch}_{val_score:.4f}.pth')
    return path

```

Gambar 0.7 save model

```

def param_groups_with_weight_decay(model, weight_decay):
    """Pisahkan parameter yang tidak perlu weight decay (bias/Norm)."""
    decay, no_decay = [], []
    for name, p in model.named_parameters():
        if not p.requires_grad:
            continue
        if p.ndim == 1 or name.endswith(".bias") or "norm" in name.lower() or "bn" in
name.lower():
            no_decay.append(p)
        else:
            decay.append(p)
    return [
        {'params': decay, 'weight_decay': weight_decay},
        {'params': no_decay, 'weight_decay': 0.0},
    ]

```

Gambar 0.8 param groups with weight decay

```

def train_epoch(model, loader, criterion, optimizer, device, epoch_idx=None, wandb_sync=False,
wandb_prefix="train"):
    total_loss = 0
    predictions = []
    labels = []

    for batch in loader:
        batch = batch.to(device)
        optimizer.zero_grad()
        out = model(batch.x, batch.edge_index, batch.batch)
        out = out.squeeze(1) # Make output shape match target
        loss = criterion(out, batch.y)
        loss.backward()
        optimizer.step()

        total_loss += loss.item() * batch.num_graphs
        batch_preds = (out.detach().cpu().numpy() > 0.5).astype(int)
        predictions.extend(batch_preds)
        labels.extend(batch.y.cpu().numpy().astype(int))

    avg_loss = total_loss / len(loader.dataset)
    metrics = classification_report(labels, predictions, output_dict=True, zero_division=0)
    if wandb_sync:
        wandb.log(
            {
                f"{wandb_prefix}train_loss": avg_loss,
                f"{wandb_prefix}train_accuracy": metrics["accuracy"],
                f"{wandb_prefix}train_precision": metrics["weighted avg"]["precision"],
                f"{wandb_prefix}train_recall": metrics["weighted avg"]["recall"],
                f"{wandb_prefix}train_f1": metrics["weighted avg"]["f1-score"],
                "epoch": epoch_idx,
            }
        )
    return avg_loss, metrics

```

Gambar 0.9 train epoch

```

def train_single_fold(fold_number, config, device, wandb_sync=False):
    """Train a model for a specific fold"""
    set_seeds(config['seed'])

    model = ImprovedGCN(
        num_node_features=config['num_node_features'],
        hidden_channels=config['hidden_channels'],
        dropout_rate=config['dropout_rate'],
        residual=config['residual'],
        seed=config['seed']
    )

    model = model.to(device)
    criterion = BCELoss()
    param_group = param_groups_with_weight_decay(model, config['weight_decay'])
    optimizer = build_optimizer(
        config['optimizer'],
        param_group,
        lr=config['learning_rate'],
        weight_decay=config['weight_decay']
    )
    scheduler = StepLR(optimizer, step_size=config['scheduler_step_size'],
gamma=config['scheduler_gamma'])
    monitor_metric = config.get("monitor_metric", "val_accuracy")
    mode = "max" if monitor_metric in ("val_accuracy", "val_f1") else "min"
    early_stopper = EarlyStopping(
        patience=config.get("early_stopping_patience", 10),
        min_delta=config.get("early_stopping_min_delta", 0.0),
        mode=mode,
    )

    train_dataset = FallDetectionDatasetKFold(fold_number=fold_number, is_train=True)
    val_dataset = FallDetectionDatasetKFold(fold_number=fold_number, is_train=False)

    train_loader = DataLoader(train_dataset, batch_size=config['batch_size'], shuffle=True)
    val_loader = DataLoader(val_dataset, batch_size=config['batch_size'], shuffle=False)

```

Gambar 0.10 train single fold part 1

```

print(f"\n{'='*20} Training Fold {fold_number} {'='*20}")

best_val_accuracy = 0
best_val_f1 = 0
best_val_epoch = 0
best_confusion_matrix = None
top_model_paths = []

log_data = []
best_model_path = None

epoch_history = []

ckpt_base = os.path.join(
    config.get("checkpoint_dir", "checkpoints"),
    f"{config['run_name']}",
    f"fold{fold_number}"
)
os.makedirs(ckpt_base, exist_ok=True)

for epoch in range(config['epochs']):
    print(f"\nEpoch {epoch+1}/{config['epochs']}")

    prefix = f"fold{fold_number}/"

    # Train and validate
    train_loss, train_metrics = train_epoch(
        model,
        train_loader,
        criterion,
        optimizer,
        device,
        epoch_idx=epoch + 1,
        wandb_sync=wandb_sync,
        wandb_prefix=prefix
    )

    val_loss, val_metrics, val_cm = validate_epoch(
        model,
        val_loader,
        criterion,
        device,
        epoch_idx=epoch + 1,
        wandb_sync=wandb_sync,
        wandb_prefix=prefix
    )

    current_f1 = val_metrics['weighted avg']['f1-score']
    current_accuracy = val_metrics['accuracy']

    monitor_value = (
        current_accuracy
        if monitor_metric == "val_accuracy"
        else current_f1
        if monitor_metric == "val_f1"
        else val_loss
    )

    is_best = False
    if current_accuracy > best_val_accuracy:
        best_val_accuracy = current_accuracy
        best_val_f1 = current_f1
        best_val_epoch = epoch + 1
        best_confusion_matrix = val_cm.copy()
        is_best = True
    elif abs(current_accuracy - best_val_accuracy) < 1e-6 and current_f1 > best_val_f1:
        best_val_f1 = current_f1
        best_val_epoch = epoch + 1
        best_confusion_matrix = val_cm.copy()
        is_best = True

```

Gambar 0.11 train single fold part 2

```

        save_checkpoint(
            model,
            optimizer,
            scheduler,
            epoch + 1,
            config,
            path=os.path.join(ckpt_base, "last.pth"),
        )

        model_path = save_model(
            model, config['run_name'], fold_number, epoch + 1, val_metrics['accuracy']
        )
        if is_best:
            best_model_path = os.path.join(ckpt_base, "best.pth")
            save_checkpoint(
                model,
                optimizer,
                scheduler,
                epoch + 1,
                config,
                path=best_model_path,
            )
            print(f"New best model saved: {best_model_path}")

        top_model_paths = manage_saved_models(model_path, top_model_paths)

        print(f"Train Loss: {train_loss:.4f}, Accuracy: {train_metrics['accuracy']:.4f}")
        print(f"Val Loss: {val_loss:.4f}, Accuracy: {val_metrics['accuracy']:.4f}")
        print(f"Precision: {val_metrics['weighted avg']['precision']:.4f}, " +
              f"Recall: {val_metrics['weighted avg']['recall']:.4f}, " +
              f"F1: {val_metrics['weighted avg']['f1-score']:.4f}")

        if is_best:
            print(f"👉 New best model Accuracy: ({best_val_accuracy:.4f}, F1: {best_val_f1:.4f})")

        scheduler.step()

        epoch_history.append(
            {
                "epoch": epoch + 1,
                "train_loss": float(train_loss),
                "val_loss": float(val_loss),
                "train_accuracy": float(train_metrics['accuracy']),
                "val_accuracy": float(val_metrics['accuracy']),
                "val_f1": float(val_metrics['weighted avg']['f1-score']),
            }
        )

        log_data.append({
            'optimizer': config['optimizer'],
            'fold': fold_number,
            'epoch': epoch + 1,
            'batch_size': config['batch_size'],
            'learning_rate': config['learning_rate'],
            'weight_decay': config['weight_decay'],
            'dropout_rate': config['dropout_rate'],
            'residual': config['residual'],
            'train_loss': train_loss,
            'train_accuracy': train_metrics['accuracy'],
            'val_loss': val_loss,
            'val_accuracy': val_metrics['accuracy'],
            'val_precision': val_metrics['weighted avg']['precision'],
            'val_recall': val_metrics['weighted avg']['recall'],
            'val_f1': val_metrics['weighted avg']['f1-score'],
            'best_model_path': best_model_path,
            'best_confusion_matrix': str(best_confusion_matrix) if best_confusion_matrix is not None else
None, })
    }

    if early_stopper.step(monitor_value):
        print(f"Early stopping on fold {fold_number} at epoch {epoch + 1} monitor:
{monitor_metric}={k}")

    log_training_data_to_excel(log_data, f"training_log_{config['run_name']}_{fold}{fold_number}.xlsx")

    print(f"\nFold {fold_number} Best Results:")
    print(f"Best Validation Accuracy: {best_val_accuracy:.4f} (Epoch {best_val_epoch})")
    print(f"Best F1 Score: {best_val_f1:.4f}")
    print(f"Best Confusion Matrix for Fold {fold_number}:")
    print(best_confusion_matrix)

    return {
        'fold': fold_number,
        'best_accuracy': best_val_accuracy,
        'best_f1': best_val_f1,
        'best_epoch': best_val_epoch,
        'best_confusion_matrix': best_confusion_matrix,
        'top_models': top_model_paths,
        'best_model_path': best_model_path,
        'epoch_history': epoch_history
    }
}

```

```

def main(wandb_sync=False):
    set_seeds(42)
    config = {
        'epochs': 100,
        'batch_sizes': [32],
        'learning_rates': [1e-3],
        'weight_decays': [1e-5],
        'optimizers': ['adamw'],
        'num_node_features': 3,
        'hidden_channels': [64, 32, 32, 32, 32],
        'dropout_rates': [0.3],
        'residuals': [True],
        'scheduler_step_size': 20,
        'scheduler_gamma': 0.5,
        'seed': 42,
        'run_name': f"5fold_{datetime.now().strftime('%Y%m%d%H%M%S')}",
        "early_stopping_patience": 20,
        "early_stopping_min_delta": 1e-4,
        "monitor_metric": "val_accuracy",
        "checkpoint_dir": "checkpoints",
        "wandb_mode": "per_combo",
        "wandb_project": "fall-detection-5Fold-fixed",
        "wandb_entity": "leonardosirait80-itera",
    }

    device = check_set_gpu()

    split_dir = 'splits'
    os.makedirs(split_dir, exist_ok=True)

    all_folds_exist = all(
        os.path.exists(os.path.join(split_dir, f'5fold_split_fold{fold}.json'))
        for fold in range(1, 6)
    )

    if not all_folds_exist:
        print("Generating 5-fold splits...")
        fold_splits = generate_kfold_splits(DATASET_PATH, k=5, seed=config['seed'])
        save_kfold_splits(fold_splits)
        is_valid = verify_fold_separation(fold_splits)

        if is_valid:
            print("All folds have proper separation between train and validation sets!")
        else:
            print("WARNING: Some folds have overlapping train and validation files.")
        return

    summary = run_grid_search(config, device, wandb_sync=wandb_sync)

    if summary and summary.get("best_combination"):
        print("== OVERALL BEST ==")
        print(summary["best_combination"])

if __name__ == '__main__':
    main(wandb_sync=True)

```

Gambar 0.13 fungsi main

```

def manage_saved_models(new_path, top_model_paths, max_saved=3):
    """Manage the saved model files, keeping only the top performing ones"""
    top_model_paths.append(new_path)
    # Sort paths by validation score (extract score from filename)
    sorted_paths = sorted(top_model_paths,
        key=lambda x: float(x.split('val')[-1].split('.pth')[0]),
        reverse=True)

    # Keep only the top max_saved models
    for path in sorted_paths[max_saved:]:
        if os.path.exists(path):
            os.remove(path)

    return sorted_paths[:max_saved]

```

Gambar 0.14 manage saved models

```

def validate_epoch(model, loader, criterion, device, epoch_idx=None, wandb_sync=False, wandb_prefix=""):
    model.eval()
    total_loss = 0
    predictions = []
    labels = []

    with torch.no_grad():
        for batch in loader:
            batch = batch.to(device)
            out = model(batch.x, batch.edge_index, batch.batch)
            out = out.squeeze(1) # Make output shape match target
            loss = criterion(out, batch.y)

            total_loss += loss.item() * batch.num_graphs
            batch_preds = (out.cpu().numpy() > 0.5).astype(int) # tandain sebagai int
            predictions.extend(batch_preds)
            labels.extend(batch.y.cpu().numpy().astype(int)) # tandain sebagai int

    avg_loss = total_loss / len(loader.dataset)
    metrics = classification_report(labels, predictions, output_dict=True, zero_division=0)

    # Calculate confusion matrix
    cm = confusion_matrix(labels, predictions, labels=[0, 1])

    # Calculate metrics directly from confusion matrix
    # This will be used to determine the "best" confusion matrix
    try:
        tn, fp, fn, tp = cm.ravel()
        cm_accuracy = (tp + tn) / (tp + tn + fp + fn) if (tp + tn + fp + fn) > 0 else 0
        cm_precision = tp / (tp + fn) if (tp + fn) > 0 else 0
        cm_recall = tp / (tp + fn) if (tp + fn) > 0 else 0
        cm_f1 = 2 * (cm_precision * cm_recall) / (cm_precision + cm_recall) if (cm_precision + cm_recall) > 0 else 0
    except Exception as e:
        print(f"Error calculating confusion matrix metrics: {e}")
        cm_f1 = 0
        cm_accuracy = 0

    # Add these metrics to the metrics dictionary
    metrics['cm_f1'] = cm_f1
    metrics['cm_accuracy'] = cm_accuracy

if wandb_sync:
    wandb.log(
        {
            f'{wandb_prefix}val_loss': avg_loss,
            f'{wandb_prefix}val_accuracy': metrics["accuracy"],
            f'{wandb_prefix}val_precision': metrics["weighted avg"]["precision"],
            f'{wandb_prefix}val_recall': metrics["weighted avg"]["recall"],
            f'{wandb_prefix}val_f1': metrics["weighted avg"]["f1-score"],
            f'{wandb_prefix}avg_cm_accuracy': metrics["cm_accuracy"],
            f'{wandb_prefix}avg_cm_f1': metrics["cm_f1"],
            "epoch": epoch_idx,
        }
    )

return avg_loss, metrics, cm

```

Gambar 0.15 validate epoch