



KERJA PRAKTIK - EF234603

Implementasi *SlowFast Network* dalam Klasifikasi Kejadian Hampir Kecelakaan

**Laboratorium Pemodelan dan Komputasi Terapan
Institut Teknologi Sepuluh Nopember**

Periode: 10 September 2024 - 20 Februari 2025

Oleh:

Andika Rahman Teja

5025221022

Pembimbing Jurusan

Dr. Ahmad Saikh, S.Si., M.T.

Pembimbing Lapangan

Dr. Eng. Muhamad Hilmi Muchtar Aditya Pradana, S.Kom, M.Sc.

DEPARTEMEN TEKNIK INFORMATIKA

Fakultas Teknologi Elektro dan Informatika Cerdas

Institut Teknologi Sepuluh Nopember

Surabaya 2025



KERJA PRAKTIK - EF234603

Implementasi *SlowFast Network* dalam Klasifikasi Kejadian Hampir Kecelakaan

Laboratorium Pemodelan dan Komputasi Terapan
Institut Teknologi Sepuluh Nopember

Periode: 10 September 2024 – 20 Februari 2025

Oleh:

Andika Rahman Teja

5025221022

Pembimbing Jurusan

Dr. Ahmad Saikhu, S.Si., M.T.

Pembimbing Lapangan

Dr. Eng. Muhamad Hilmi Muchtar Aditya Pradana, S.Kom, M.Sc.

DEPARTEMEN TEKNIK INFORMATIKA

Fakultas Teknologi Elektro dan Informatika Cerdas

Institut Teknologi Sepuluh Nopember

Surabaya 2025

[Halaman ini sengaja dikosongkan]

DAFTAR ISI

DAFTAR ISI	iv
DAFTAR GAMBAR	viii
DAFTAR TABEL	x
LEMBAR PENGESAHAN	Error! Bookmark not defined.
ABSTRAK	xiv
KATA PENGANTAR	xvi
BAB I PENDAHULUAN	1
1.1. Latar Belakang	1
1.2. Tujuan	1
1.3. Manfaat	2
1.4. Rumusan Masalah	2
1.5. Lokasi dan Waktu Kerja Praktik	2
1.6. Metodologi Kerja Praktik	2
1.6.1. Perumusan Masalah	2
1.6.2. Studi Literatur	3
1.6.3. Analisis dan Perancangan Sistem	3
1.6.4. Implementasi Sistem	3
1.6.5. Pengujian dan Evaluasi	3
1.6.6. Kesimpulan dan Saran	3
1.7. Sistematika Laporan	4
1.7.1. Bab I Pendahuluan	4

1.7.2.	Bab II Profil Perusahaan	4
1.7.3.	Bab III Tinjauan Pustaka	4
1.7.4.	Bab IV Implementasi Sistem	4
1.7.5.	Bab V Pengujian dan Evaluasi	4
1.7.6.	Bab VI Kesimpulan dan Saran	4
BAB II PROFIL PERUSAHAAN		6
2.1.	Profil Laboratorium Pemodelan dan Komputasi Terapan ITS.	6
2.2.	Lokasi	6
BAB III TINJAUAN PUSTAKA		8
3.1.	Python	8
3.2.	PyTorch	8
3.3.	CUDA	9
3.4.	OpenCV	9
3.5.	<i>Video Recognition</i>	10
3.6.	DADA-2000	10
BAB IV IMPLEMENTASI SISTEM		13
4.1.	Analisis Infrastruktur <i>SlowFast Network</i>	13
5.3.1.	Jalur <i>Slow</i>	13
5.3.2.	Jalur <i>Fast</i>	14
4.2.	Tahap Pemahaman Data	14
4.3.	Tahap Pemrosesan Data	17
4.4.	Tahap Pemodelan	21

4.5.	Tahap Evaluasi	24
BAB V PENGUJIAN DAN EVALUASI		28
5.1.	Tujuan Pengujian	28
5.2.	Kriteria Pengujian	28
5.3.	Skenario Pengujian	28
5.3.1.	<i>SlowFast</i> 4x16	28
5.3.2.	<i>SlowFast</i> 8x8	29
5.3.3.	<i>SlowFast</i> 8x8 stepwise	29
5.3.4.	<i>SlowFast</i> NLN 4x16	29
5.3.5.	<i>SlowFast</i> NLN 8x8	30
5.4.	Evaluasi Pengujian	30
BAB VI KESIMPULAN DAN SARAN		32
6.1.	Kesimpulan	32
6.2.	Saran	32
DAFTAR PUSTAKA		34
BIODATA PENULIS		46

[Halaman ini sengaja dikosongkan]

DAFTAR GAMBAR

Gambar 3. 1 Pembagian Temporal Window dalam Anotasi Ulang Data DADA-2000.....	11
Gambar 3. 2 Perbedaan Jumlah Kelas Klasifikasi pada Anotasi Ulang Data DADA-2000.....	11
Gambar 4. 1 Arsitektur SlowFast Network	13
Gambar 4. 2 Format Data Video Kinetics	15
Gambar 4. 3 Contoh Kelas Klasifikasi Menabrak Pejalan Kaki .	15
Gambar 4. 4 Contoh Kelas Klasifikasi Menabrak Kendaraan.....	16
Gambar 4. 5 Contoh Kelas Klasifikasi Menabrak Penghalang ...	16
Gambar 4. 6 Contoh Kelas Klasifikasi Tidak Menabrak.....	16
Gambar 4. 7 Import dan Deklarasi Variabel Konstan Untuk Pemrosesan Data	17
Gambar 4. 8 Pembuatan Loader Data.....	18
Gambar 4. 9 Proses Normalisasi Data DADA-2000	19
Gambar 4. 10 Proses Augmentasi Data DADA-2000	20
Gambar 4. 11 Tahap Pemodelan Arsitektur SlowFast Network .	24
Gambar 4. 12 Tahap Evaluasi Model SlowFast Network	26

[Halaman ini sengaja dikosongkan]

DAFTAR TABEL

Tabel 5. 1 Hasil Pengujian 5 Konfigurasi SlowFast Network.....30

[Halaman ini sengaja dikosongkan]

**LEMBAR PENGESAHAN
KERJA PRAKTIK**

**Implementasi *SlowFast Network* dalam Klasifikasi
Kejadian Hampir Kecelakaan**

Oleh:

Andika Rahman Teja

5025221022

Disetujui oleh Pembimbing Kerja Praktik:

1. Dr. Ahmad Saikhu, S.Si.,
M.T.
NIDN. 0018077105



(Pembimbing Departemen)

2. Dr. Eng. Muhamad Hilmil
Muchtar Aditya Pradana,
S.Kom, M.Sc.
NIP. 199205162024061001



(Pembimbing Lapangan)

[Halaman ini sengaja dikosongkan]

Implementasi *SlowFast Network* dalam Klasifikasi Kejadian Hampir Kecelakaan

Nama Mahasiswa : Andika Rahman Teja
NRP : 5025221022
Departemen : Teknik Informatika FTEIC-ITS
Pembimbing Departemen : Dr. Ahmad Saikhu, S.Si., M.T.
Pembimbing Lapangan : Dr. Eng. Muhamad Hilmi Muchtar
Aditya Pradana, S.Kom, M.Sc.

ABSTRAK

Teknologi mobil swakemudi terus berkembang dengan integrasi kecerdasan buatan untuk meningkatkan keselamatan berkendara. Salah satu pendekatan yang digunakan adalah penerapan arsitektur *SlowFast Network* guna memahami informasi spasial dan temporal dari rekaman video untuk mendeteksi kejadian akan kecelakaan. Pada penelitian ini mengevaluasi performa lima konfigurasi model *SlowFast Network* pada data video kecelakaan DADA-2000. Hasil pengujian menunjukkan bahwa konfigurasi *SlowFast* 4x16 mencapai akurasi tertinggi dengan Top-1 *Accuracy* sebesar 68.24%. Temuan ini menegaskan efektivitas *SlowFast Network* dalam mengolah informasi secara simultan dan dapat mendukung pengembangan sistem deteksi kecelakaan yang lebih akurat dan responsif. Untuk dapat meningkatkan potensi model *SlowFast Network* sebagai sistem deteksi kecelakaan, penulis menyarankan untuk melakukan *hyperparameter tuning* terhadap tiap konfigurasi model *SlowFast Network* dan menerapkan teknik *cross-validation* untuk evaluasi model yang lebih andal.

Kata Kunci : Mobil Swakemudi, *SlowFast Network*, Deteksi Kecelakaan

[Halaman ini sengaja dikosongkan]

KATA PENGANTAR

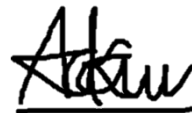
Puji syukur penulis panjatkan kepada Allah SWT atas hidayah dan karunia-Nya sehingga penulis dapat menyelesaikan salah satu kewajiban penulis sebagai mahasiswa Departemen Teknik Informatika ITS yaitu Kerja Praktik dengan judul: Implementasi *SlowFast Network* dalam Klasifikasi Kejadian Hampir Kecelakaan.

Penulis menyadari bahwa masih banyak kekurangan baik dalam melaksanakan kerja praktik maupun penyusunan buku laporan kerja praktik ini. Namun penulis berharap buku laporan ini dapat menambah wawasan pembaca dan dapat menjadi sumber referensi.

Melalui buku laporan ini penulis juga ingin menyampaikan rasa terima kasih kepada orang-orang yang telah membantu menyusun laporan kerja praktik baik secara langsung maupun tidak langsung antara lain:

1. Kedua orang tua penulis.
2. Bapak Dr. Ahmad Saikhu, S.Si., M.T. selaku dosen pembimbing kerja praktik sekaligus koordinator kerja praktik.
3. Bapak Dr. Eng. Muhamad Hilmil Muchtar Aditya Pradana, S.Kom, M.Sc. selaku pembimbing lapangan selama kerja praktik berlangsung.
4. Teman-teman penulis yang senantiasa memberikan semangat ketika penulis melaksanakan KP.

Surabaya, 20 Februari 2025



Andika Rahman Teja

[Halaman ini sengaja dikosongkan]

BAB I

PENDAHULUAN

1.1. Latar Belakang

Teknologi mobil swakemudi (*self-driving car*) telah banyak beredar di masyarakat. Berbagai teknologi modern digunakan untuk meningkatkan efisiensi, kenyamanan, dan sistem keamanan bagi pengguna (Daily, 2017). Salah satu teknologi cerdas yang diimplementasikan pada mobil swakemudi adalah penerapan algoritma *computer vision* untuk mendeteksi segala risiko jalanan yang ada. Algoritma tersebut diimplementasikan pada CCTV ataupun kamera yang ada pada dasbor mobil (Swief, 2018).

Seiring berkembangnya teknologi, algoritma *computer vision* sekarang dipadukan dengan kecerdasan buatan sehingga mampu mengidentifikasi tiap kejadian yang ada di sekitar mobil secara *real-time* (Bocca, 2019). Salah satunya adalah memberikan peringatan akan terjadinya kecelakaan yang akan dialami oleh pengendara mobil (Karuppasamy, 2021). Peristiwa ini menjadi sebuah studi baru bagi para peneliti untuk membuat sebuah sistem kecerdasan buatan yang efektif dalam mengidentifikasi kejadian akan kecelakaan. Adapun pada penelitian ini digunakanlah arsitektur *SlowFast Network* sehingga mampu memahami informasi spasial dan temporal dari rekaman video dan mengklasifikasi jenis kecelakaan tersebut secara akurat.

1.2. Tujuan

Tujuan kerja praktik ini adalah untuk menyelesaikan kewajiban nilai kerja praktik sebesar empat

SKS dan membantu perkembangan sistem keamanan pengendara dalam mendeteksi kejadian akan kecelakaan.

1.3. Manfaat

Manfaat yang diperoleh dengan adanya penelitian ini adalah meningkatkan sistem keselamatan pada pengendara dengan mengembangkan model yang mampu mengenali jenis kecelakaan yang akan terjadi sehingga membantu mengurangi risiko di jalan raya. Selain itu, penelitian ini diharapkan dapat mendukung pengembangan mobil swakemudi yang lebih aman dan andal.

1.4. Rumusan Masalah

Rumusan masalah dari kerja praktik ini adalah sebagai berikut:

1. Bagaimana kemampuan *SlowFast Network* dalam memahami informasi spasial dan temporal pada rekaman video?
2. Bagaimana penerapan arsitektur *SlowFast Network* dalam melakukan klasifikasi kejadian akan kecelakaan?

1.5. Lokasi dan Waktu Kerja Praktik

Kerja praktik ini dilaksanakan secara *remote* dengan periode kerja dimulai pada tanggal 10 September 2024 – 20 Februari 2025.

1.6. Metodologi Kerja Praktik

Metodologi dalam pembuatan buku kerja praktik meliputi :

1.6.1. Perumusan Masalah

Pada tahap ini, Bapak Dr. Eng. Muhamad Hilmil Muchtar Aditya Pradana, S.Kom, M.Sc. memberikan

pemaparan terkait alur kerja praktik yang akan penulis kerjakan. Adapun sesi pemaparan data dan label yang akan digunakan dalam kerja praktik ini.

1.6.2. Studi Literatur

Bapak Dr. Eng. Muhamad Hilmil Muchtar Aditya Pradana, S.Kom, M.Sc. telah melakukan riset pendahulu terkait klasifikasi kejadian akan kecelakaan. Riset tersebut menitikberatkan pada pemberian anotasi kembali pada data video insiden kecelakaan, yakni DADA-2000. Adapun pada riset tersebut, telah dilakukan klasifikasi insiden kecelakaan untuk menguji keakuratan dan konsistensi label data. Atas dasar riset tersebut, dilakukanlah kerja praktik ini yang digunakan untuk menguji akurasi prediksi klasifikasi dengan menggunakan arsitektur *deep learning* yang berbeda, yakni dengan *SlowFast Network*.

1.6.3. Analisis dan Perancangan Sistem

Melakukan identifikasi kebutuhan fungsional dan non-fungsional, termasuk *tools* dan *library* yang diperlukan untuk mendukung implementasi bahasa pemrograman yang digunakan.

1.6.4. Implementasi Sistem

Melakukan *setup Python environment* dan *tools* yang mendukung proses pelatihan model pada salah satu komputer yang ada di Teknik Informatika ITS.

1.6.5. Pengujian dan Evaluasi

Dilakukan beberapa skenario pengujian dengan menggunakan 5 konfigurasi arsitektur *SlowFast Network*. Adapun evaluasi yang digunakan adalah menggunakan metrik akurasi.

1.6.6. Kesimpulan dan Saran

Pengujian yang dilakukan ini telah memenuhi syarat yang diinginkan, dan berjalan dengan baik dan lancar.

1.7. Sistematika Laporan

1.7.1. Bab I Pendahuluan

Bab ini berisi latar belakang, tujuan, manfaat, rumusan masalah, lokasi dan waktu kerja praktik, metodologi, dan sistematika laporan.

1.7.2. Bab II Profil Perusahaan

Bab ini berisi gambaran umum profil dan lokasi Laboratorium Pemodelan dan Komputasi Terapan ITS.

1.7.3. Bab III Tinjauan Pustaka

Bab ini berisi dasar teori dari teknologi yang digunakan dalam menyelesaikan proyek kerja praktik.

1.7.4. Bab IV Implementasi Sistem

Bab ini berisi uraian tahap - tahap yang dilakukan untuk proses implementasi model.

1.7.5. Bab V Pengujian dan Evaluasi

Bab ini hasil uji coba dan evaluasi dari model yang telah diuji selama pelaksanaan kerja praktik.

1.7.6. Bab VI Kesimpulan dan Saran

Bab ini berisi kesimpulan dan saran yang didapat dari proses pelaksanaan kerja praktik.

[Halaman ini sengaja dikosongkan]

BAB II

PROFIL PERUSAHAAN

2.1. Profil Laboratorium Pemodelan dan Komputasi Terapan ITS.

Laboratorium Pemodelan dan Komputasi Terapan ITS merupakan laboratorium yang ada di Departemen Teknik Informatika ITS yang memiliki fungsi sebagai wadah untuk penelitian dan kerja sama industri dalam bidang pemodelan dan simulasi, peramalan ilmiah, optimasi, serta komputasi saintifik. Beberapa mata kuliah pendukung yang diajarkan meliputi Matematika Diskrit, Aljabar Linier dan Matriks, Teori Graf dan Otomata, Komputasi Numerik, Probabilitas dan Statistik, Riset Operasi, Pemodelan dan Simulasi, serta Analisis Data Multivariat. Selain itu, terdapat pula beberapa mata kuliah tingkat pascasarjana lainnya.

2.2. Lokasi

Jl. Teknik Kimia, Kelurahan Keputih, Kecamatan Sukolilo, Kota Surabaya, Jawa Timur 60111

[Halaman ini sengaja dikosongkan]

BAB III

TINJAUAN PUSTAKA

3.1. Python

Python adalah bahasa pemrograman tingkat tinggi dikembangkan oleh Guido van Rossum dan dikenalkan sejak tahun 1991. Python dirancang untuk kemudahan membaca serta efisiensi dalam pengembangan perangkat lunak. Bahasa pemrograman ini mendukung berbagai paradigma pemrograman, termasuk pemrograman berorientasi objek, pemrograman prosedural, dan pemrograman fungsional, sehingga dapat digunakan dalam berbagai domain, mulai dari pengembangan web hingga kecerdasan buatan. Selain itu, dukungan komunitas yang kuat serta sifatnya sebagai perangkat lunak yang *open-source* turut berkontribusi pada popularitas Python dan menjadikan Python sebagai pilihan utama bagi pemula maupun profesional di dunia pemrograman (Lutz, 2013).

3.2. PyTorch

PyTorch adalah kerangka kerja *open-source* yang dikembangkan oleh Facebook AI Research (FAIR) untuk komputasi numerik dan pembelajaran mesin, terutama dalam *deep learning*. Dengan desain yang intuitif dan berbasis Tensor, PyTorch memungkinkan pengembang untuk membangun dan melatih model *neural network* secara dinamis menggunakan metode *automatic differentiation* yang efisien. Adapun komunitas yang luas serta integrasi yang erat dengan pustaka seperti TorchVision dan HuggingFace Transformers membuat PyTorch telah menjadi standar industri dalam pengembangan model *deep learning* (Paszke, 2019).

3.3. CUDA

CUDA (Compute Unified Device Architecture) adalah platform komputasi paralel dan model pemrograman yang dikembangkan oleh NVIDIA untuk memungkinkan penggunaan *Graphics Processing Unit* (GPU) dalam komputasi umum. Dengan CUDA, pengguna dapat memanfaatkan ribuan inti prosesor GPU untuk mempercepat perhitungan yang kompleks, seperti pemrosesan gambar, simulasi ilmiah, dan *deep learning* (Kirk, 2010). Saat ini, CUDA sudah terintegrasi dengan berbagai *library deep learning* seperti TensorFlow dan PyTorch. Selain itu, Teknologi ini memungkinkan eksekusi paralel yang efisien melalui konsep *kernels*, *threads*, dan *blocks*, yang dirancang untuk meningkatkan kinerja aplikasi berbasis komputasi intensif.

3.4. OpenCV

OpenCV (*Open Source Computer Vision Library*) adalah pustaka *open-source* yang dirancang untuk pemrosesan citra dan visi komputer, digunakan secara luas dalam berbagai aplikasi seperti pengenalan wajah, deteksi objek, dan analisis video. Dikembangkan dalam bahasa C++ dengan antarmuka untuk Python, OpenCV menyediakan berbagai algoritma optimasi yang memungkinkan pemrosesan citra secara real-time dengan efisiensi tinggi (Howse, 2020). *Library* ini juga mendukung akselerasi perangkat keras menggunakan CUDA dan OpenCL, sehingga dapat meningkatkan kinerja komputasi pada GPU.

3.5. *Video Recognition*

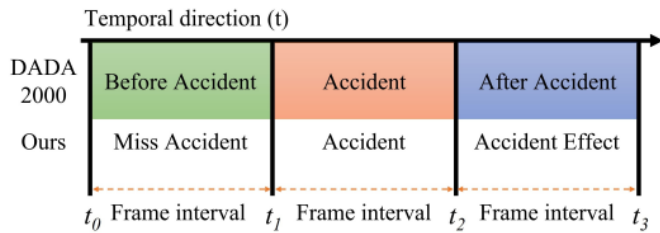
Video recognition adalah cabang dari *computer vision* yang berfokus pada analisis dan interpretasi konten video untuk mengenali objek, peristiwa, atau pola tertentu di dalamnya. Teknologi ini melibatkan ekstraksi fitur dari antar *frame* video dan penggunaan algoritma *deep learning* untuk mengenali aktivitas atau objek berdasarkan pola temporal dan spasial. Teknologi ini memiliki aplikasi luas, mulai dari pengawasan keamanan, analisis olahraga, hingga sistem rekomendasi konten multimedia (Ballas, 2016).

3.6. **DADA-2000**

Data *Driver Attention in Driving Accident* (DADA) merupakan data video yang terdiri dari 2.000 klip dengan total sekitar 658.476 frame, mencakup 54 jenis kecelakaan yang berbeda. Dataset ini dikumpulkan dari berbagai kondisi jalan (seperti jalan tol, perkotaan, pedesaan, dan terowongan), cuaca (cerah, hujan, dan salju), serta pencahayaan (siang dan malam). Setiap klip dilengkapi dengan anotasi perhatian pengemudi, termasuk peta fiksasi, *saccade path*, waktu fokus, hingga informasi detail tentang kecelakaan seperti kategori, durasi, dan lokasi objek tabrakan. Tujuan utama DADA-2000 adalah untuk menyediakan *benchmark* bagi penelitian prediksi perhatian pengemudi dalam skenario kecelakaan sehingga dapat membantu dalam pengembangan sistem mengemudi yang lebih aman (Fang, 2021).

Namun pada data DADA-2000, tidak ada perbedaan antara kejadian akan kecelakaan (*near-miss*) dan kejadian saat kecelakaan terjadi (*accident*). Sehingga pada penelitian yang dilakukan oleh Pradana et al. (2023) mendefinisikan dan memberi anotasi ulang pada data kecelakaan yang ada pada DADA-2000. Adapun metode

yang digunakan pada penelitian tersebut adalah memperpanjang dan memberikan label waktu awal dan durasi akhir kecelakaan. Melalui pendekatan metode tersebut, diperoleh hasil data yang konsisten dalam mengklasifikasikan semua kemungkinan risiko kecelakaan lalu lintas.



Gambar 3. 1 Pembagian Temporal Window dalam Anotasi Ulang Data DADA-2000

4 classes	7 classes	# 16 classes
1. Hitting pedestrian	1. Hitting pedestrian	1. Crossing pedestrian
2. Hitting vehicles	2. Hitting cyclist	2. Hitting pedestrian
3. Hitting obstacles	3. Hitting motorbike	3. Crossing cyclist
4. Normal	4. Hitting truck	4. Hitting cyclist
	5. Hitting car	5. Crossing motorbike
	6. Self-accident	6. Hitting motorbike
	7. Normal	7. Crossing truck
		8. Hitting truck
		9. Overtaking truck
		10. Crossing car
		11. Hitting car
		12. Overtaking car
		13. Hitting roadblocks
		14. Hitting road facilities
		15. Self-accident
		16. Normal

Gambar 3. 2 Perbedaan Jumlah Kelas Klasifikasi pada Anotasi Ulang Data DADA-2000

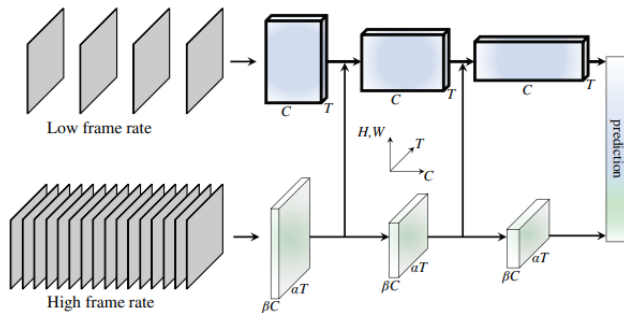
[Halaman ini sengaja dikosongkan]

BAB IV

IMPLEMENTASI SISTEM

4.1. Analisis Infrastruktur *SlowFast Network*

Arsitektur *SlowFast Network* memproses informasi dalam 2 jalur paralel, yakni jalur *slow* dan jalur *fast*. Jalur *slow* akan memproses informasi yang memiliki resolusi temporal rendah. Sementara itu, jalur *fast* akan memproses informasi yang memiliki resolusi temporal yang lebih tinggi. Meski berjalan pada 2 jalur yang berbeda, pada arsitektur ini terdapat proses memadukan informasi antar kedua jalur melalui proses *summation* atau *concatenation* informasi yang telah diproses dari jalur *fast* ke jalur *slow*. Adapun kedua jalur tersebut menggunakan model 3D ResNet-50 untuk menangkap berbagai *frame* dalam saat yang bersamaan dan melakukan operasi 3D *convolution* pada *frame-frame* tersebut (Feichtenhofer, 2019).



Gambar 4. 1 Arsitektur *SlowFast Network*

5.3.1. Jalur *Slow*

Konsep utama pada jalur *slow* adalah mendefinisikan langkah temporal τ yang besar pada

input frame. Berdasarkan konfigurasi *default* dari model *SlowFast Network*, didefinisikan nilai τ adalah 16 sehingga jika terdapat sebuah data video dengan resolusi 30-fps, maka akan terdapat 2 *frame* yang akan diambil sampel per detik. Dengan menyatakan jumlah *frame* yang diambil sampelnya sebagai T , maka panjang klip video *input* adalah $T \times \tau$ *frame* (Feichtenhofer, 2019).

5.3.2. Jalur *Fast*

Pada jalur ini akan mengambil jumlah *frame* lebih banyak daripada jalur *slow*. Jika jalur *slow* mengambil T sampel *frame*, maka jalur *fast* akan mengambil αT sampel *frame*. Untuk dapat memproses resolusi *input* yang tinggi, model *SlowFast Network* tidak menggunakan lapisan *temporal downsampling* hingga lapisan global *pooling* terakhir yang digunakan untuk klasifikasi. Hal ini membuat *frame* yang diproses tetap berjumlah αT (Feichtenhofer, 2019).

Selain itu, jalur *fast* memiliki jumlah *channel* yang lebih sedikit dibandingkan dengan jalur *slow*, yakni sebesar rasio $\frac{1}{\beta}$ (misal jumlah *channel* pada jalur *slow* adalah 64, maka jumlah *channel* pada jalur *fast* adalah $\frac{64}{\beta}$). Pengurangan jumlah *channel* ini dikarenakan jalur *fast* dirancang untuk menangkap informasi temporal dengan efisien. Teknik ini membuat komputasi model *SlowFast Network* menjadi lebih efektif (Feichtenhofer, 2019).

4.2. Tahap Pemahaman Data

Pada penelitian ini, digunakan data video kecelakaan kendaraan bermotor (DADA-2000) sebanyak 704 data dengan pembagian sebagai berikut: 489 data latih (70%), 67 data validasi (10%), dan 148 (20%) data uji. Selain itu,

pada penelitian ini akan menggunakan 4 klasifikasi kelas, yakni menabrak pejalan kaki (kelas 0), menabrak kendaraan (kelas 1), menabrak penghalang (kelas 2), dan tidak menabrak (kelas 3). Data video tersebut dikemas dalam format data Kinetics.

	./DADA_2000/00_00_00_00_0000.mp4	0
0	./DADA_2000/00_00_00_00_0001.mp4	0
1	./DADA_2000/00_00_00_00_0003.mp4	0
2	./DADA_2000/00_00_00_00_0005.mp4	0
3	./DADA_2000/00_00_00_00_0006.mp4	0
4	./DADA_2000/00_00_00_00_0007.mp4	0
...	...	...
483	./DADA_2000/00_03_06_15_0266.mp4	3
484	./DADA_2000/00_03_06_15_0267.mp4	3
485	./DADA_2000/00_03_06_15_0270.mp4	3
486	./DADA_2000/00_03_06_15_0271.mp4	3
487	./DADA_2000/00_03_06_15_0272.mp4	3

488 rows x 2 columns

Gambar 4. 2 Format Data Video Kinetics



Gambar 4. 3 Contoh Kelas Klasifikasi Menabrak Pejalan Kaki



Gambar 4. 4 Contoh Kelas Klasifikasi Menabrak Kendaraan



Gambar 4. 5 Contoh Kelas Klasifikasi Menabrak Penghalang



Gambar 4. 6 Contoh Kelas Klasifikasi Tidak Menabrak

4.3. Tahap Pemrosesan Data

Pertama, memanggil semua *library* yang diperlukan dan variabel konstan untuk memproses data.

```
import os
import random

import numpy as np
import pandas

import slowfast.utils.logging as logging
import torch
import torch.utils.data
from slowfast.utils.env import pathmgr
from torchvision import transforms
from slowfast.datasets import (
    decoder as decoder,
    transform as transform,
    utils as utils,
    video_container as container,
)
from slowfast.datasets.build import DATASET_REGISTRY
from slowfast.datasets.random_erasing import RandomErasing
from slowfast.datasets.transform import create_random_augment, MaskingGenerator,
MaskingGenerator3D
from fvcare.common.config import CfgNode

logger = logging.get_logger(__name__)

_C = CfgNode()
_C.DATA.NUM_FRAMES = 8
_C.DATA.MEAN = [0.45, 0.45, 0.45]
_C.DATA.STD = [0.225, 0.225, 0.225]
_C.DATA.TRAIN_JITTER_SCALES = [256, 320]
_C.TEST.NUM_SPATIAL_CROPS = 3
_C.DATA.TRAIN_CROP_SIZE = 224
_C.DATA.TEST_CROP_SIZE = 256
_C.DATA.TRAIN_CROP_NUM_TEMPORAL = 1
_C.DATA.TRAIN_CROP_NUM_SPATIAL = 1
_C.DATA.LOADER_CHUNK_SIZE = 0
_C.DATA.DECODING_BACKEND = "pyav"
_C.DATA.PATH_LABEL_SEPARATOR = " "
_C.DATA.PATH_PREFIX = ""
_C.AUG.AA_TYPE = "rand-m9-mstd0.5-inc1"
_C.AUG.INTERPOLATION = "bicubic"
_C.DATA.RANDOM_FLIP = True
_C.AUG.RE_PROB = 0.25
_C.AUG.RE_MODE = "pixel"
_C.AUG.RE_COUNT = 1
```

Gambar 4. 7 Import dan Deklarasi Variabel Konstan Untuk Pemrosesan Data

Selanjutnya, dilakukan proses memuat data video DADA-2000 dalam format Kinetics dengan membuat sebuah *loader*.

```
def _construct_loader(self):
    path_to_file = os.path.join(
        self.cfg.DATA.PATH_TO_DATA_DIR, "{}.csv".format(self.mode)
    )
    assert pathmgr.exists(path_to_file), "{} dir not found".format(path_to_file)

    self._path_to_videos = []
    self._labels = []
    self._spatial_temporal_idx = []
    self.cur_iter = 0
    self.chunk_epoch = 0
    self.epoch = 0.0
    self.skip_rows = 0

    with pathmgr.open(path_to_file, "r") as f:
        if self.use_chunk_loading:
            rows = self._get_chunk(f,
                self.cfg.DATA.LOADER_CHUNK_SIZE)
        else:
            rows = f.read().splitlines()
            for clip_idx, path_label in enumerate(rows):
                fetch_info =
                path_label.split(self.cfg.DATA.PATH_LABEL_SEPARATOR)
                if len(fetch_info) == 2:
                    path, label = fetch_info
                elif len(fetch_info) == 3:
                    path, fn, label = fetch_info
                elif len(fetch_info) == 1:
                    path, label = fetch_info[0], 0
                else:
                    raise RuntimeError(
                        "Failed to parse video fetch {} info {}".format(
                            retries, path_to_file, fetch_info
                        )
                    )
                for idx in range(self._num_clips):
                    self._path_to_videos.append(
                        os.path.join(self.cfg.DATA.PATH_PREFIX, path)
                    )
                    self._labels.append(int(label))
                    self._spatial_temporal_idx.append(idx)
                    self._video_meta[clip_idx * self._num_clips + idx] =
                {}
    assert (
        len(self._path_to_videos) > 0
    ), "Failed to load Kinetics split {} from {}".format(
        self._split_idx, path_to_file
    )
    logger.info(
        "Constructing kinetics dataloader (size: {} skip_rows {})
        from {}".format(
            len(self._path_to_videos), self.skip_rows, path_to_file
        )
    )
)
```

Gambar 4. 8 Pembuatan Loader Data

Setelah membuat sebuah *loader*, selanjutnya data video DADA-2000 tersebut akan dilakukan proses normalisasi dan augmentasi sebagai berikut.

```
def __getitem__(self, index):
    if self.mode in ["train", "val"]:
        # -1 indicates random sampling.
        temporal_sample_index = -1
        spatial_sample_index = -1
        min_scale = self.cfg.DATA.TRAIN_JITTER_SCALES[0]
        max_scale = self.cfg.DATA.TRAIN_JITTER_SCALES[1]
        crop_size = self.cfg.DATA.TRAIN_CROP_SIZE
    elif self.mode in ["test"]:
        temporal_sample_index = (
            self._spatial_temporal_idx[index] // self.cfg.TEST.NUM_SPATIAL_CROPS
        )
        # spatial_sample_index is in [0, 1, 2]. Corresponding to left,
        # center, or right if width is larger than height, and top, middle,
        # or bottom if height is larger than width.
        spatial_sample_index = (
            (self._spatial_temporal_idx[index] %
             self.cfg.TEST.NUM_SPATIAL_CROPS)
            if self.cfg.TEST.NUM_SPATIAL_CROPS > 1
            else 1
        )
        min_scale, max_scale, crop_size = (
            [self.cfg.DATA.TEST_CROP_SIZE] * 3
            if self.cfg.TEST.NUM_SPATIAL_CROPS > 1
            else [self.cfg.DATA.TRAIN_JITTER_SCALES[0]] * 2
            + [self.cfg.DATA.TEST_CROP_SIZE]
        )
        # min_scale, max_scale, and crop_size are expect to be the same.
        assert len({min_scale, max_scale}) == 1
        num_decode = (
            self.cfg.DATA.TRAIN_CROP_NUM_TEMPORAL if self.mode in ["train"] else 1
        )
        min_scale, max_scale, crop_size = [min_scale], [max_scale]
        if len(min_scale) < num_decode:
            min_scale += [self.cfg.DATA.TRAIN_JITTER_SCALES[0]] * (
                num_decode - len(min_scale)
            )
            max_scale += [self.cfg.DATA.TRAIN_JITTER_SCALES[1]] * (
                num_decode - len(max_scale)
            )
        assert self.mode in ["train", "val"]
```

Gambar 4. 9 Proses Normalisasi Data DADA-2000

```

for i in range(num_decode):
    for _ in range(num_aug):
        idx += 1
        f_out[idx] = frames_decoded[i].clone()
        time_idx_out[idx] = time_idx_decoded[i, :]

        f_out[idx] = f_out[idx].float()
        f_out[idx] = f_out[idx] / 255.0

        if self.aug and self.cfg.AUG.AA_TYPE:
            aug_transform = create_random_augment(
                input_size=(f_out[idx].size(1), f_out[idx].size(2)),
                auto_augment=self.cfg.AUG.AA_TYPE,
                interpolation=self.cfg.AUG.INTERPOLATION,
            )
            # T H W C -> T C H W.
            f_out[idx] = f_out[idx].permute(0, 3, 1, 2)
            list_img = self._frame_to_list_img(f_out[idx])
            list_img = aug_transform(list_img)
            f_out[idx] = self._list_img_to_frames(list_img)
            f_out[idx] = f_out[idx].permute(0, 2, 3, 1)

        # Perform color normalization.
        f_out[idx] = utils.tensor_normalize(
            f_out[idx], self.cfg.DATA.MEAN, self.cfg.DATA.STD
        )

        # T H W C -> C T H W.
        f_out[idx] = f_out[idx].permute(3, 0, 1, 2)

        f_out[idx] = utils.spatial_sampling(
            f_out[idx],
            spatial_idx=spatial_sample_index,
            min_scale=min_scale[i],
            max_scale=max_scale[i],
            crop_size=crop_size[i],
            random_horizontal_flip=self.cfg.DATA.RANDOM_FLIP
        )

        if self.rand_erase:
            erase_transform = RandomErasing(
                self.cfg.AUG.RE_PROB,
                mode=self.cfg.AUG.RE_MODE,
                max_count=self.cfg.AUG.RE_COUNT,
                num_splits=self.cfg.AUG.RE_COUNT,
                device="cpu",
            )
            f_out[idx] = erase_transform(
                f_out[idx].permute(1, 0, 2, 3)
            ).permute(1, 0, 2, 3)

        f_out[idx] = utils.pack_pathway_output(self.cfg, f_out[idx])

```

Gambar 4. 10 Proses Augmentasi Data DADA-2000

4.4. Tahap Pemodelan

Arsitektur *SlowFast Network* menerapkan ResNet-50 sebagai *backbone* utamanya. Berikut ini adalah potongan kode dari arsitektur *SlowFast Network*.

```
import torch.nn as nn
from slowfast.models import head_helper, resnet_helper, stem_helper # noqa

_MODEL_STAGE_DEPTH = {18: (2, 2, 2, 2), 50: (3, 4, 6, 3), 101: (3, 4, 23, 3)}
_TEMPORAL_KERNEL_BASIS = {
    "slowfast": [
        [[1], [5]], # conv1 temporal kernel for slow and fast pathway.
        [[1], [3]], # res2 temporal kernel for slow and fast pathway.
        [[1], [3]], # res3 temporal kernel for slow and fast pathway.
        [[3], [3]], # res4 temporal kernel for slow and fast pathway.
        [[3], [3]], # res5 temporal kernel for slow and fast pathway.
    ]
}

_POOL1 = {
    "slowfast": [[1, 1, 1], [1, 1, 1]]
}

def _construct_network(self, cfg):
    assert cfg.MODEL.ARCH in _POOL1.keys()
    pool_size = _POOL1[cfg.MODEL.ARCH]
    assert len({len(pool_size), self.num_pathways}) == 1
    assert cfg.RESNET.DEPTH in _MODEL_STAGE_DEPTH.keys()

    (d2, d3, d4, d5) = _MODEL_STAGE_DEPTH[cfg.RESNET.DEPTH]

    num_groups = cfg.RESNET.NUM_GROUPS
    width_per_group = cfg.RESNET.WIDTH_PER_GROUP
    dim_inner = num_groups * width_per_group
    out_dim_ratio = cfg.SLOWFAST.BETA_INV //
    cfg.SLOWFAST.FUSION_CONV_CHANNEL_RATIO

    temp_kernel = _TEMPORAL_KERNEL_BASIS[cfg.MODEL.ARCH]

    self.s1 = stem_helper.VideoModelStem(
        dim_in=cfg.DATA.INPUT_CHANNEL_NUM,
        dim_out=[width_per_group, width_per_group // cfg.SLOWFAST.BETA_INV],
        kernel=[temp_kernel[0][0] + [7, 7], temp_kernel[0][1] + [7, 7]],
        stride=[[1, 2, 2]] * 2,
        padding=[
            [temp_kernel[0][0][0] // 2, 3, 3],
            [temp_kernel[0][1][0] // 2, 3, 3],
        ],
        norm_module=self.norm_module,
    )

    self.s1_fuse = FuseFastToSlow(
        width_per_group // cfg.SLOWFAST.BETA_INV,
        cfg.SLOWFAST.FUSION_CONV_CHANNEL_RATIO,
        cfg.SLOWFAST.FUSION_KERNEL_SZ,
        cfg.SLOWFAST.ALPHA,
        norm_module=self.norm_module,
    )
```

```

self.s2 = resnet_helper.ResStage(
    dim_in=[
        width_per_group + width_per_group // out_dim_ratio,
        width_per_group // cfg.SLOWFAST.BETA_INV,
    ],
    dim_out=[
        width_per_group * 4,
        width_per_group * 4 // cfg.SLOWFAST.BETA_INV,
    ],
    dim_inner=[dim_inner, dim_inner // cfg.SLOWFAST.BETA_INV],
    temp_kernel_sizes=temp_kernel[1],
    stride=cfg.RESNET.SPATIAL_STRIDES[0],
    num_blocks=[d2] * 2,
    num_groups=[num_groups] * 2,
    num_block_temp_kernel=cfg.RESNET.NUM_BLOCK_TEMP_KERNEL[0],
    nonlocal_inds=cfg.NONLOCAL.LOCATION[0],
    nonlocal_group=cfg.NONLOCAL.GROUP[0],
    nonlocal_pool=cfg.NONLOCAL.POOL[0],
    instantiation=cfg.NONLOCAL.INSTANTIATION,
    trans_func_name=cfg.RESNET.TRANS_FUNC,
    dilation=cfg.RESNET.SPATIAL_DILATIONS[0],
    norm_module=self.norm_module,
)

self.s2_fuse = FuseFastToSlow(
    width_per_group * 4 // cfg.SLOWFAST.BETA_INV,
    cfg.SLOWFAST.FUSION_CONV_CHANNEL_RATIO,
    cfg.SLOWFAST.FUSION_KERNEL_SZ,
    cfg.SLOWFAST.ALPHA,
    norm_module=self.norm_module,
)

for pathway in range(self.num_pathways):
    pool = nn.MaxPool3d(
        kernel_size=pool_size[pathway],
        stride=pool_size[pathway],
        padding=[0, 0, 0],
    )
    self.add_module("pathway{}{}_pool".format(pathway, pool))

self.s3 = resnet_helper.ResStage(
    dim_in=[
        width_per_group * 4 + width_per_group * 4 // out_dim_ratio,
        width_per_group * 4 // cfg.SLOWFAST.BETA_INV,
    ],
    dim_out=[
        width_per_group * 8,
        width_per_group * 8 // cfg.SLOWFAST.BETA_INV,
    ],
    dim_inner=[dim_inner * 2, dim_inner * 2 // cfg.SLOWFAST.BETA_INV],
    temp_kernel_sizes=temp_kernel[2],
    stride=cfg.RESNET.SPATIAL_STRIDES[1],
    num_blocks=[d3] * 2,
    num_groups=[num_groups] * 2,
    num_block_temp_kernel=cfg.RESNET.NUM_BLOCK_TEMP_KERNEL[1],
    nonlocal_inds=cfg.NONLOCAL.LOCATION[1],
    nonlocal_group=cfg.NONLOCAL.GROUP[1],
    nonlocal_pool=cfg.NONLOCAL.POOL[1],
    instantiation=cfg.NONLOCAL.INSTANTIATION,
    trans_func_name=cfg.RESNET.TRANS_FUNC,
    dilation=cfg.RESNET.SPATIAL_DILATIONS[1],
    norm_module=self.norm_module,
)

self.s3_fuse = FuseFastToSlow(
    width_per_group * 8 // cfg.SLOWFAST.BETA_INV,
    cfg.SLOWFAST.FUSION_CONV_CHANNEL_RATIO,
    cfg.SLOWFAST.FUSION_KERNEL_SZ,
    cfg.SLOWFAST.ALPHA,
    norm_module=self.norm_module,
)

```

```

self.s4 = resnet_helper.ResStage(
    dim_in=[
        width_per_group * 8 + width_per_group * 8 // out_dim_ratio,
        width_per_group * 8 // cfg.SLOWFAST.BETA_INV,
    ],
    dim_out=[
        width_per_group * 16,
        width_per_group * 16 // cfg.SLOWFAST.BETA_INV,
    ],
    dim_inner=[dim_inner * 4, dim_inner * 4 // cfg.SLOWFAST.BETA_INV],
    temp_kernel_sizes=temp_kernel[3],
    stride=cfg.RESNET.SPATIAL_STRIDES[2],
    num_blocks=[d4] * 2,
    num_groups=[num_groups] * 2,
    num_block_temp_kernel=cfg.RESNET.NUM_BLOCK_TEMP_KERNEL[2],
    nonlocal_inds=cfg.NONLOCAL.LOCATION[2],
    nonlocal_group=cfg.NONLOCAL.GROUP[2],
    nonlocal_pool=cfg.NONLOCAL.POOL[2],
    instantiation=cfg.NONLOCAL.INSTANTIATION,
    trans_func_name=cfg.RESNET.TRANS_FUNC,
    dilation=cfg.RESNET.SPATIAL_DILATIONS[2],
    norm_module=self.norm_module,
)
self.s4_fuse = FuseFastToSlow(
    width_per_group * 16 // cfg.SLOWFAST.BETA_INV,
    cfg.SLOWFAST.FUSION_CONV_CHANNEL_RATIO,
    cfg.SLOWFAST.FUSION_KERNEL_SZ,
    cfg.SLOWFAST.ALPHA,
    norm_module=self.norm_module,
)
self.s5 = resnet_helper.ResStage(
    dim_in=[
        width_per_group * 16 + width_per_group * 16 // out_dim_ratio,
        width_per_group * 16 // cfg.SLOWFAST.BETA_INV,
    ],
    dim_out=[
        width_per_group * 32,
        width_per_group * 32 // cfg.SLOWFAST.BETA_INV,
    ],
    dim_inner=[dim_inner * 8, dim_inner * 8 // cfg.SLOWFAST.BETA_INV],
    temp_kernel_sizes=temp_kernel[4],
    stride=cfg.RESNET.SPATIAL_STRIDES[3],
    num_blocks=[d5] * 2,
    num_groups=[num_groups] * 2,
    num_block_temp_kernel=cfg.RESNET.NUM_BLOCK_TEMP_KERNEL[3],
    nonlocal_inds=cfg.NONLOCAL.LOCATION[3],
    nonlocal_group=cfg.NONLOCAL.GROUP[3],
    nonlocal_pool=cfg.NONLOCAL.POOL[3],
    instantiation=cfg.NONLOCAL.INSTANTIATION,
    trans_func_name=cfg.RESNET.TRANS_FUNC,
    dilation=cfg.RESNET.SPATIAL_DILATIONS[3],
    norm_module=self.norm_module,
)

```

```

self.head = head_helper.ResNetBasicHead(
    dim_in=[
        width_per_group * 32,
        width_per_group * 32 // cfg.SLOWFAST.BETA_INV,
    ],
    num_classes=cfg.MODEL.NUM_CLASSES,
    pool_size=(
        [None, None]
        if cfg.MULTIGRID.SHORT_CYCLE
        or cfg.MODEL.MODEL_NAME == "ContrastiveModel"
        else [
            [
                cfg.DATA.NUM_FRAMES
                // cfg.SLOWFAST.ALPHA
                // pool_size[0][0],
                cfg.DATA.TRAIN_CROP_SIZE // 32 // pool_size[0][1],
                cfg.DATA.TRAIN_CROP_SIZE // 32 // pool_size[0][2],
            ],
            [
                cfg.DATA.NUM_FRAMES // pool_size[1][0],
                cfg.DATA.TRAIN_CROP_SIZE // 32 // pool_size[1][1],
                cfg.DATA.TRAIN_CROP_SIZE // 32 // pool_size[1][2],
            ],
        ],
    ), # None for AdaptiveAvgPool3d((1, 1, 1))
    dropout_rate=cfg.MODEL.DROPOUT_RATE,
    act_func=cfg.MODEL.HEAD_ACT,
    detach_final_fc=cfg.MODEL.DETACH_FINAL_FC,
    cfg=cfg,
)

```

Gambar 4. 11 Tahap Pemodelan Arsitektur SlowFast Network

4.5. Tahap Evaluasi

Pada penelitian ini, model dievaluasi dengan metrik Top-1 *Accuracy*. Berikut ini potongan kode untuk menampilkan akurasi model.

```

import os
import pickle
import numpy as np
import slowfast.utils.distributed as du
import slowfast.utils.logging as logging
import torch
from slowfast.utils.env import pathmgr
logger = logging.get_logger(__name__)

@torch.no_grad()
def perform_test(test_loader, model, test_meter, cfg, writer=None):
    model.eval()
    test_meter.iter_tic()
    for cur_iter, (inputs, labels, video_idx, time, meta) in
        enumerate(test_loader):
        if cfg.NUM_GPUS:
            if isinstance(inputs, (list,)):
                for i in range(len(inputs)):
                    inputs[i] = inputs[i].cuda(non_blocking=True)
            else:
                inputs = inputs.cuda(non_blocking=True)
            labels = labels.cuda()
            video_idx = video_idx.cuda()
            for key, val in meta.items():
                if isinstance(val, (list,)):
                    for i in range(len(val)):
                        val[i] = val[i].cuda(non_blocking=True)
                else:
                    meta[key] = val.cuda(non_blocking=True)
            test_meter.data_toc()
            preds = model(inputs)
            if cfg.NUM_GPUS:
                preds = preds.cpu()
                labels = labels.cpu()
                video_idx = video_idx.cpu()
            test_meter.iter_toc()
            test_meter.log_iter_stats(cur_iter)
            test_meter.iter_tic()

        if not cfg.DETECTION.ENABLE:
            all_preds = test_meter.video_preds.clone().detach()
            all_labels = test_meter.video_labels
            if cfg.NUM_GPUS:
                all_preds = all_preds.cpu()
                all_labels = all_labels.cpu()
            if writer is not None:
                writer.plot_eval(preds=all_preds, labels=all_labels)

        if cfg.TEST.SAVE_RESULTS_PATH != "":
            save_path = os.path.join(cfg.OUTPUT_DIR, cfg.TEST.SAVE_RESULTS_PATH)
            if du.is_root_proc():
                with pathmgr.open(save_path, "wb") as f:
                    pickle.dump([all_preds, all_labels], f)
                logger.info("Successfully saved prediction results to
                    {}".format(save_path))

        test_meter.finalize_metrics()
    return test_meter

```

```

for view, test_meter in zip(cfg.TEST.NUM_TEMPORAL_CLIPS, test_meters):
    logger.info(
        "Finalized testing with {} temporal clips and {} spatial crops".format(
            view, cfg.TEST.NUM_SPATIAL_CROPS
        )
    )
    result_string_views += "{}a{}\n".format(view,
test_meter.stats["top1_acc"])
    result_string = (
        "_p{:.2f}_f{:.2f}_{}_a Top1 Acc: {} MEM: {:.2f} f: {:.4f}"
        "".format(
            params / 1e6,
            flops,
            view,
            test_meter.stats["top1_acc"],
            misc.gpu_mem_usage(),
            flops,
        )
    )
    logger.info("{}".format(result_string))
    logger.info("{}".format(result_string_views))
    return result_string + "\n" + result_string_views

```

Gambar 4. 12 Tahap Evaluasi Model SlowFast Network

[Halaman ini sengaja dikosongkan]

BAB V

PENGUJIAN DAN EVALUASI

Bab ini akan menjelaskan tahapan pengujian dan evaluasi model *SlowFast Network* terhadap data DADA-2000.

5.1. Tujuan Pengujian

Pada penelitian ini bertujuan untuk menguji keakuratan 5 konfigurasi arsitektur *SlowFast Network* terhadap klasifikasi kejadian akan kecelakaan pada data DADA-2000. Adapun kelas klasifikasi yang digunakan adalah menabrak pejalan kaki (kelas 0), menabrak kendaraan (kelas 1), menabrak penghalang (kelas 2), dan tidak menabrak (kelas 3).

5.2. Kriteria Pengujian

Pencapaian tujuan pengujian dinilai berdasarkan akurasi model dalam melakukan prediksi. Akurasi model akan menggunakan metrik evaluasi Top-1 *Accuracy*. Tidak ada target spesifik untuk Top-1 *Accuracy* tersebut. Selain itu, kelima konfigurasi tersebut dilatih pada jumlah *epoch* yang sama, yakni 100 *epoch*.

5.3. Skenario Pengujian

Pengujian dilakukan pada 5 konfigurasi arsitektur *SlowFast Network* sebagai berikut.

5.3.1. *SlowFast* 4x16

Konfigurasi ini mendefinisikan nilai α sebesar 8 sehingga jalur *slow* akan mengambil sampel *frame* sebanyak $\frac{\text{num_frame}}{\alpha} = \frac{32}{8} = 4 \text{ frame}$. Sementara pada jalur

fast akan mengambil sampel *frame* sebanyak $\alpha T = 8 \cdot 2 = 16 \text{ frame}$.

5.3.2. *SlowFast 8x8*

Konfigurasi ini mendefinisikan nilai α sebesar 4 sehingga jalur *slow* akan mengambil sampel *frame* sebanyak $\frac{\text{num_frame}}{\alpha} = \frac{32}{4} = 8 \text{ frame}$. Sementara pada jalur *fast* akan mengambil sampel *frame* sebanyak $\alpha T = 4 \cdot 2 = 8 \text{ frame}$. Selain itu, jumlah *fusion kernel size* pada konfigurasi ini sebanyak 7 kernel. Hal ini menyebabkan lebih banyak informasi temporal yang diproses dibandingkan dengan konfigurasi *SlowFast 4x16* yang hanya memiliki *fusion kernel size* sebesar 5 kernel.

5.3.3. *SlowFast 8x8 stepwise*

Konfigurasi ini mendefinisikan *learning rate* yang terus menurun secara diskrit pada *epoch* tertentu. Tujuannya adalah untuk mendapatkan *local optimum* lebih cepat. Selain variabel *learning rate* tersebut, semua variabelnya mirip dengan konfigurasi *SlowFast 8x8*.

5.3.4. *SlowFast NLN 4x16*

Konfigurasi ini mendefinisikan adanya *Non-Local Network* (NLN) pada beberapa *stage* dari ResNet. Pada *stage* pertama dan keempat, tidak terdapat *non-local module*. Pada *stage* kedua, terdapat *non-local module* pada blok ke-1 dan ke-3. Sementara pada *stage* ketiga, terdapat *non-local module* pada blok ke-1, ke-3, dan ke-5. *Non-local module* menerapkan *self-attention mechanism* untuk menangkap dependensi jarak jauh sehingga mampu memproses informasi spasial dan temporal pada *frame-frame* video.

5.3.5. *SlowFast* NLN 8x8

Konfigurasi ini merupakan gabungan *SlowFast* 8x8 dengan *SlowFast* NLN 4x16, yakni dengan jumlah α yang sama dengan *SlowFast* 8x8 dan adanya *Non-local module* yang sama dengan *SlowFast* NLN 4x16.

5.4. Evaluasi Pengujian

Berdasarkan hasil pengujian dan evaluasi, didapatkan Top-1 *Accuracy* dari kelima konfigurasi model *SlowFast Network* sebagai berikut:

Konfigurasi Model	Top-1 <i>Accuracy</i>
<i>SlowFast</i> 4x16	68.24
<i>SlowFast</i> 8x8	54.05
<i>SlowFast</i> 8x8 stepwise	66.89
<i>SlowFast</i> NLN 4x16	35.14
<i>SlowFast</i> NLN 8x8	60.81

Tabel 5. 1 Hasil Pengujian 5 Konfigurasi *SlowFast Network*

Berdasarkan tabel tersebut, didapatkan hasil akurasi tertinggi adalah pada konfigurasi *SlowFast* 4x16.

[Halaman ini sengaja dikosongkan]

BAB VI

KESIMPULAN DAN SARAN

6.1. Kesimpulan

Kesimpulan yang didapat setelah melakukan pengujian 5 konfigurasi model *SlowFast Network* pada data video kecelakaan DADA-2000 adalah sebagai berikut:

- a. Konfigurasi *SlowFast* 4x16 mencapai akurasi tertinggi dengan Top-1 *Accuracy* sebesar 68.24%, menunjukkan bahwa konfigurasi ini paling efektif dalam mengenali kejadian kecelakaan dari data DADA-2000.
- b. *SlowFast* 8x8 *stepwise* memiliki performa mendekati *SlowFast* 4x16, dengan Top-1 *Accuracy* 66.89%, yang menunjukkan bahwa metode *stepwise* dapat meningkatkan akurasi dibandingkan dengan konfigurasi standar *SlowFast* 8x8.
- c. Konfigurasi *SlowFast* NLN 8x8 memiliki akurasi lebih baik dibandingkan *SlowFast* NLN 4x16, dengan 60.81%. Hal ini yang menunjukkan bahwa NLN lebih efektif pada konfigurasi dengan temporal *resolution* yang lebih pendek.

6.2. Saran

Saran untuk penelitian lebih lanjut dari model *SlowFast Network* terhadap klasifikasi kejadian akan kecelakaan adalah sebagai berikut:

- a. Melakukan *hyperparameter tuning* terhadap tiap konfigurasi model *SlowFast Network*.
- b. Melakukan teknik *cross-validation* untuk mengevaluasi model secara lebih *robust*.
- c. Mengimplementasikan model *SlowFast Network* terbaik pada kendaraan swakemudi.

[Halaman ini sengaja dikosongkan]

DAFTAR PUSTAKA

- Ballas, N., Yao, L., Pal, C., & Courville, A. (2016). *Delving deeper into convolutional networks for learning video representations*. International Conference on Learning Representations.
- Bocca, Alberto & Baek, Donkyu. (2019). *Automated Driving Systems: Key Advantages, Limitations and Risks*. 1-6. 10.23919/EETA.2019.8804564.
- Daily, M., Medasani, S., Behringer, R., & Trivedi, M. (2017). *Self-driving cars*. Computer, 50(12), 18-23. <https://doi.org/10.1109/MC.2017.4451204>
- Fang, Jianwu & Yan, Dingxin & Qiao, Jiahuan & Xue, Jianru & Yu, Hongkai. (2021). *DADA: Driver Attention Prediction in Driving Accident Scenarios*. IEEE Transactions on Intelligent Transportation Systems. PP. 1-13. 10.1109/TITS.2020.3044678.
- Feichtenhofer, C., Fan, H., Malik, J., & He, K. (2019). *SlowFast networks for video recognition*. 2021 IEEE/CVF International Conference on Computer Vision (ICCV). <https://doi.org/10.1109/iccv.2019.00630>
- Howse, J., & Minichino, J. (2020). *Learning OpenCV 4 computer vision with Python 3: Get to grips with tools, techniques, and algorithms for computer vision and machine learning*. Packt Publishing
- Kirk, D. B., & Hwu, W.-m. W. (2010). *Programming massively parallel processors: A hands-on approach*. Morgan Kaufmann Publishers Inc.

- Lutz, M. (2013). *Learning Python*. O'Reilly Media, Inc.
- M, Karuppasamy & Sainath, Vayara & S, Sreenatha. (2021). *Deep Learning for Autonomous Driving System*. 1744-1749. 10.1109/ICESC51422.2021.9532819.
- Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E. Z., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., . . . Chintala, S. (2019). *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. arXiv (Cornell University), 32, 8026–8037. <https://arxiv.org/pdf/1912.01703.pdf>
- Pradana, Muhamad Hilmi Muchtar & Dao, Minh & Zettsu, Koji. (2023). *Augmenting Ego-Vehicle for Traffic Near-Miss and Accident Classification Dataset using Manipulating Conditional Style Translation*. 10.48550/arXiv.2301.02726.
- Swief, A., & El-Habrouk, M. (2018). *A survey of Automotive Driving Assistance Systems technologies*. 2018 International Conference on Artificial Intelligence and Data Processing (IDAP), 1-12.

[Halaman ini sengaja dikosongkan]

LAMPIRAN

Lampiran 1. Konfigurasi *SlowFast* 4x16

```
TRAIN:
  ENABLE: True
  DATASET: kinetics
  BATCH_SIZE: 8
  EVAL_PERIOD: 10
  CHECKPOINT_PERIOD: 1
  AUTO_RESUME: True
DATA:
  NUM_FRAMES: 32
  SAMPLING_RATE: 2
  TRAIN_JITTER_SCALES: [256, 320]
  TRAIN_CROP_SIZE: 224
  TEST_CROP_SIZE: 256
  INPUT_CHANNEL_NUM: [3, 3]
  PATH_TO_DATA_DIR: .
SLOWFAST:
  ALPHA: 8
  BETA_INV: 8
  FUSION_CONV_CHANNEL_RATIO: 2
  FUSION_KERNEL_SZ: 5
RESNET:
  ZERO_INIT_FINAL_BN: True
  WIDTH_PER_GROUP: 64
  NUM_GROUPS: 1
  DEPTH: 50
  TRANS_FUNC: bottleneck_transform
  STRIDE_1X1: False
  NUM_BLOCK_TEMP_KERNEL: [[3, 3], [4, 4], [6, 6], [3, 3]]
  SPATIAL_STRIDES: [[1, 1], [2, 2], [2, 2], [2, 2]]
  SPATIAL_DILATIONS: [[1, 1], [1, 1], [1, 1], [1, 1]]
NONLOCAL:
  LOCATION: [[[[]], [[]], [[]], [[]], [[]], [[]], [[]], [[]]]
  GROUP: [[1, 1], [1, 1], [1, 1], [1, 1], [1, 1], [1, 1]]
  INSTANTIATION: dot_product
BN:
  USE_PRECISE_STATS: True
  NUM_BATCHES_PRECISE: 200
SOLVER:
  BASE_LR: 0.1
  LR_POLICY: cosine
  MAX_EPOCH: 100 #196
  MOMENTUM: 0.9
  WEIGHT_DECAY: 1e-4
  WARMUP_EPOCHS: 34.0
  WARMUP_START_LR: 0.01
  OPTIMIZING_METHOD: sgd
MODEL:
  NUM_CLASSES: 4
  ARCH: slowfast
  MODEL_NAME: SlowFast
  LOSS_FUNC: cross_entropy
  DROPOUT_RATE: 0.5
TEST:
  ENABLE: True
  DATASET: kinetics
  BATCH_SIZE: 8
DATA_LOADER:
  NUM_WORKERS: 0
  PIN_MEMORY: True
NUM_GPUS: 1
NUM_SHARDS: 1
RNG_SEED: 0
OUTPUT_DIR: .
```

Lampiran 2. Konfigurasi *SlowFast* 8x8

```
TRAIN:
  ENABLE: True
  DATASET: kinetics
  BATCH_SIZE: 8 # 64
  EVAL_PERIOD: 10
  CHECKPOINT_PERIOD: 1
  AUTO_RESUME: True
DATA:
  NUM_FRAMES: 32
  SAMPLING_RATE: 2
  TRAIN_JITTER_SCALES: [256, 320]
  TRAIN_CROP_SIZE: 224
  TEST_CROP_SIZE: 256
  INPUT_CHANNEL_NUM: [3, 3]
  PATH_TO_DATA_DIR: .
SLOWFAST:
  ALPHA: 4
  BETA_INV: 8
  FUSION_CONV_CHANNEL_RATIO: 2
  FUSION_KERNEL_SZ: 7
RESNET:
  ZERO_INIT_FINAL_BN: True
  WIDTH_PER_GROUP: 64
  NUM_GROUPS: 1
  DEPTH: 50
  TRANS_FUNC: bottleneck_transform
  STRIDE_1X1: False
  NUM_BLOCK_TEMP_KERNEL: [[3, 3], [4, 4], [6, 6], [3, 3]]
  SPATIAL_STRIDES: [[1, 1], [2, 2], [2, 2], [2, 2]]
  SPATIAL_DILATIONS: [[1, 1], [1, 1], [1, 1], [1, 1]]
NONLOCAL:
  LOCATION: [[[]], [[]], [[]], [[]], [[]], [[]], [[]], [[]]]
  GROUP: [[1, 1], [1, 1], [1, 1], [1, 1], [1, 1]]
  INSTANTIATION: dot_product
BN:
  USE_PRECISE_STATS: True
  NUM_BATCHES_PRECISE: 200
SOLVER:
  BASE_LR: 0.1
  LR_POLICY: cosine
  MAX_EPOCH: 100 #196
  MOMENTUM: 0.9
  WEIGHT_DECAY: 1e-4
  WARMUP_EPOCHS: 34.0
  WARMUP_START_LR: 0.01
  OPTIMIZING_METHOD: sgd
MODEL:
  NUM_CLASSES: 4
  ARCH: slowfast
  MODEL_NAME: SlowFast
  LOSS_FUNC: cross_entropy
  DROPOUT_RATE: 0.5
TEST:
  ENABLE: True
  DATASET: kinetics
  BATCH_SIZE: 8 # 64
DATA_LOADER:
  NUM_WORKERS: 0 # 8
  PIN_MEMORY: True
NUM_GPUS: 1 # 8
NUM_SHARDS: 1
RNG_SEED: 0
OUTPUT_DIR: .
```

Lampiran 3. Konfigurasi *SlowFast* 8x8 *stepwise*

```
TRAIN:
  ENABLE: True
  DATASET: kinetics
  BATCH_SIZE: 8
  EVAL_PERIOD: 10
  CHECKPOINT_PERIOD: 1
  AUTO_RESUME: True
DATA:
  NUM_FRAMES: 32
  SAMPLING_RATE: 2
  TRAIN_JITTER_SCALES: [256, 320]
  TRAIN_CROP_SIZE: 224
  TEST_CROP_SIZE: 256
  INPUT_CHANNEL_NUM: [3, 3]
SLOWFAST:
  ALPHA: 4
  BETA_INV: 8
  FUSION_CONV_CHANNEL_RATIO: 2
  FUSION_KERNEL_SZ: 7
RESNET:
  ZERO_INIT_FINAL_BN: True
  WIDTH_PER_GROUP: 64
  NUM_GROUPS: 1
  DEPTH: 50
  TRANS_FUNC: bottleneck_transform
  STRIDE_1X1: False
  NUM_BLOCK_TEMP_KERNEL: [[3, 3], [4, 4], [6, 6], [3, 3]]
  SPATIAL_STRIDES: [[1, 1], [2, 2], [2, 2], [2, 2]]
  SPATIAL_DILATIONS: [[1, 1], [1, 1], [1, 1], [1, 1]]
NONLOCAL:
  LOCATION: [[[]], [[]], [[]], [[]], [[]], [[]], [[]]]
  GROUP: [[1, 1], [1, 1], [1, 1], [1, 1]]
  INSTANTIATION: dot_product
BN:
  USE_PRECISE_STATS: True
  NUM_BATCHES_PRECISE: 200
SOLVER:
  BASE_LR: 0.1
  LR_POLICY: steps_with_relative_lrs
  LRS: [1, 0.1, 0.01, 0.001, 0.0001, 0.00001]
  STEPS: [0, 94, 154, 196]
  MAX_EPOCH: 100 #239
  MOMENTUM: 0.9
  WEIGHT_DECAY: 1e-4
  WARMUP_EPOCHS: 34.0
  WARMUP_START_LR: 0.01
  OPTIMIZING_METHOD: sgd
MODEL:
  NUM_CLASSES: 4
  ARCH: slowfast
  MODEL_NAME: SlowFast
  LOSS_FUNC: cross_entropy
  DROPOUT_RATE: 0.5
TEST:
  ENABLE: True
  DATASET: kinetics
  BATCH_SIZE: 8
DATA_LOADER:
  NUM_WORKERS: 0
  PIN_MEMORY: True
NUM_GPUS: 1
NUM_SHARDS: 1
RNG_SEED: 0
OUTPUT_DIR: .
```

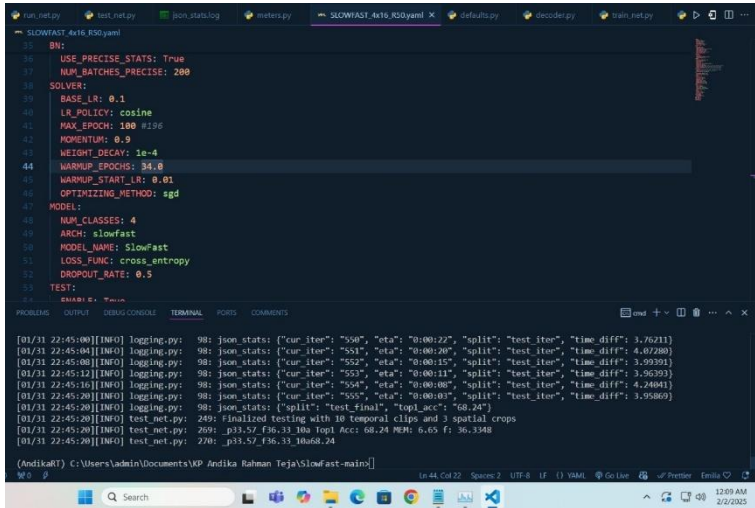
Lampiran 4. Konfigurasi *SlowFast* NLN 4x16

```
TRAIN:
  ENABLE: True
  DATASET: kinetics
  BATCH_SIZE: 8
  EVAL_PERIOD: 10
  CHECKPOINT_PERIOD: 1
  AUTO_RESUME: True
DATA:
  NUM_FRAMES: 32
  SAMPLING_RATE: 2
  TRAIN_JITTER_SCALES: [256, 320]
  TRAIN_CROP_SIZE: 224
  TEST_CROP_SIZE: 256
  INPUT_CHANNEL_NUM: [3, 3]
  PATH_TO_DATA_DIR: .
SLOWFAST:
  ALPHA: 8
  BETA_INV: 8
  FUSION_CONV_CHANNEL_RATIO: 2
  FUSION_KERNEL_SZ: 5
RESNET:
  ZERO_INIT_FINAL_BN: True
  WIDTH_PER_GROUP: 64
  NUM_GROUPS: 1
  DEPTH: 50
  TRANS_FUNC: bottleneck_transform
  STRIDE_1X1: False
  NUM_BLOCK_TEMP_KERNEL: [[3, 3], [4, 4], [6, 6], [3, 3]]
  SPATIAL_STRIDES: [[1, 1], [2, 2], [2, 2], [2, 2]]
  SPATIAL_DILATIONS: [[1, 1], [1, 1], [1, 1], [1, 1]]
NONLOCAL:
  LOCATION: [[[], []], [[1, 3], []], [[1, 3, 5], []], [[], []]]
  GROUP: [[1, 1], [1, 1], [1, 1], [1, 1]]
  INSTANTIATION: dot_product
BN:
  USE_PRECISE_STATS: True
  NUM_BATCHES_PRECISE: 200
SOLVER:
  BASE_LR: 0.1
  LR_POLICY: cosine
  MAX_EPOCH: 100 #196
  MOMENTUM: 0.9
  WEIGHT_DECAY: 1e-4
  WARMUP_EPOCHS: 34.0
  WARMUP_START_LR: 0.01
  OPTIMIZING_METHOD: sgd
MODEL:
  NUM_CLASSES: 4
  ARCH: slowfast
  MODEL_NAME: SlowFast
  LOSS_FUNC: cross_entropy
  DROPOUT_RATE: 0.5
TEST:
  ENABLE: True
  DATASET: kinetics
  BATCH_SIZE: 8
DATA_LOADER:
  NUM_WORKERS: 0
  PIN_MEMORY: True
NUM_GPUS: 1
NUM_SHARDS: 1
RNG_SEED: 0
OUTPUT_DIR: .
```

Lampiran 5. Konfigurasi *SlowFast* NLN 8x8

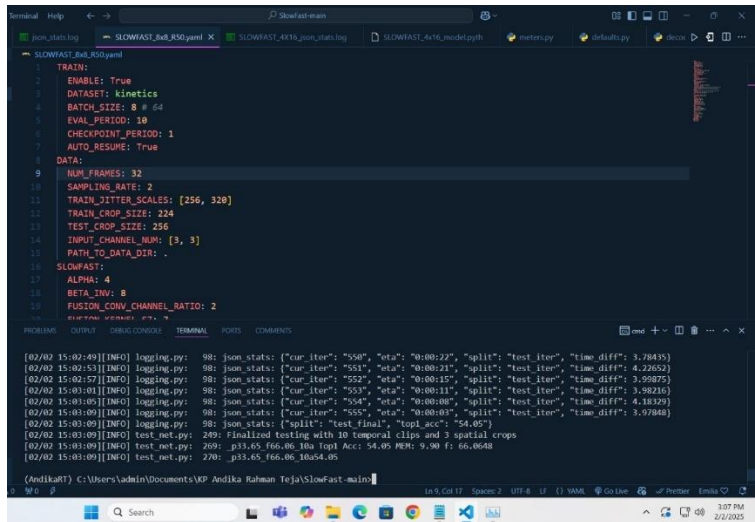
```
TRAIN:
  ENABLE: True
  DATASET: kinetics
  BATCH_SIZE: 8
  EVAL_PERIOD: 10
  CHECKPOINT_PERIOD: 1
  AUTO_RESUME: True
DATA:
  NUM_FRAMES: 32
  SAMPLING_RATE: 2
  TRAIN_JITTER_SCALES: [256, 320]
  TRAIN_CROP_SIZE: 224
  TEST_CROP_SIZE: 256
  INPUT_CHANNEL_NUM: [3, 3]
  PATH_TO_DATA_DIR: .
SLOWFAST:
  ALPHA: 4
  BETA_INV: 8
  FUSION_CONV_CHANNEL_RATIO: 2
  FUSION_KERNEL_SZ: 5
RESNET:
  ZERO_INIT_FINAL_BN: True
  WIDTH_PER_GROUP: 64
  NUM_GROUPS: 1
  DEPTH: 50
  TRANS_FUNC: bottleneck_transform
  STRIDE_1X1: False
  NUM_BLOCK_TEMP_KERNEL: [[3, 3], [4, 4], [6, 6], [3, 3]]
  SPATIAL_STRIDES: [[1, 1], [2, 2], [2, 2], [2, 2]]
  SPATIAL_DILATIONS: [[1, 1], [1, 1], [1, 1], [1, 1]]
NONLOCAL:
  LOCATION: [[[]], [[]], [[1, 3], [[]]], [[1, 3, 5], [[]], [[]], [[]]]
  GROUP: [[1, 1], [1, 1], [1, 1], [1, 1]]
  INSTANTIATION: dot_product
BN:
  USE_PRECISE_STATS: True
  NUM_BATCHES_PRECISE: 200
SOLVER:
  BASE_LR: 0.1
  LR_POLICY: cosine
  MAX_EPOCH: 100 #196
  MOMENTUM: 0.9
  WEIGHT_DECAY: 1e-4
  WARMUP_EPOCHS: 34.0
  WARMUP_START_LR: 0.01
  OPTIMIZING_METHOD: sgd
MODEL:
  NUM_CLASSES: 4
  ARCH: slowfast
  MODEL_NAME: SlowFast
  LOSS_FUNC: cross_entropy
  DROPOUT_RATE: 0.5
TEST:
  ENABLE: True
  DATASET: kinetics
  BATCH_SIZE: 8
DATA_LOADER:
  NUM_WORKERS: 0
  PIN_MEMORY: True
NUM_GPUS: 1
NUM_SHARDS: 1
RNG_SEED: 0
OUTPUT_DIR: .
```

Lampiran 6. Hasil Akurasi *SlowFast* 4x16



```
36 USE_PRECISE_STATS: True
37 NUM_BATCHES_PRECISE: 200
38
39 SOLVER:
40   BASE_LR: 0.1
41   LR_POLICY: cosine
42   MAX_EPOCH: 100 #196
43   MOMENTUM: 0.9
44   WEIGHT_DECAY: 1e-4
45
46 WARMUP_EPOCHS: 34.0
47 WARMUP_START_LR: 0.01
48 OPTIMIZING_METHOD: sgd
49
50 MODEL:
51   NUM_CLASSES: 4
52   ARCH: slowfast
53   MODEL_NAME: SlowFast
54   LOSS_FUNC: cross_entropy
55   DROPOUT_RATE: 0.5
56
57 TESTS
58   ENABLED: True
59
60 PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS
61
62 [01/31 22:45:00][INFO] logging.py: 98: json_stats: {"cur_iter": "590", "eta": "0:00:22", "split": "test_iter", "time_diff": 1.76211}
63 [01/31 22:45:04][INFO] logging.py: 98: json_stats: {"cur_iter": "551", "eta": "0:00:20", "split": "test_iter", "time_diff": 4.02200}
64 [01/31 22:45:08][INFO] logging.py: 98: json_stats: {"cur_iter": "552", "eta": "0:00:15", "split": "test_iter", "time_diff": 3.99393}
65 [01/31 22:45:12][INFO] logging.py: 98: json_stats: {"cur_iter": "553", "eta": "0:00:11", "split": "test_iter", "time_diff": 3.96393}
66 [01/31 22:45:16][INFO] logging.py: 98: json_stats: {"cur_iter": "554", "eta": "0:00:08", "split": "test_iter", "time_diff": 4.24041}
67 [01/31 22:45:20][INFO] logging.py: 98: json_stats: {"cur_iter": "555", "eta": "0:00:04", "split": "test_iter", "time_diff": 3.95060}
68 [01/31 22:45:20][INFO] logging.py: 98: json_stats: {"split": "test_final", "top1_acc": "68.24"}
69 [01/31 22:45:20][INFO] test_net.py: 249: Finalized testing with 10 temporal clips and 3 spatial crops
70 [01/31 22:45:20][INFO] test_net.py: 249: p33.52_f36.33_10a top1 Acc: 68.24 PER: 6.65 f: 36.3348
71 [01/31 22:45:20][INFO] test_net.py: 270: p33.52_f36.33_10a68.24
72
73 (AndikaRT) C:\Users\Admin\Documents\VP Andika Rahman Teja\SlowFast-main\
74
75 1648 Col 22 Space 2 UTF-8 LF (1) NAME Go Live Preview Emits Ctrl
```

Lampiran 7. Hasil Akurasi *SlowFast* 8x8



```
1 TRAIN:
2   ENABLE: True
3   DATASET: kinetics
4   BATCH_SIZE: 8 # 64
5   EVAL_PERIOD: 10
6   CHECKPOINT_PERIOD: 1
7   AUTO_RESUME: True
8
9 DATA:
10   NUM_FRAMES: 32
11   SAMPLING_RATE: 2
12   TRAIN_JITTER_SCALES: [256, 310]
13   TRAIN_CROP_SIZE: 224
14   TEST_CROP_SIZE: 256
15   INPUT_CHANNEL_NUM: [3, 3]
16   PATH_TO_DATA_DIR: .
17
18 SLOWFAST:
19   ALPHA: 4
20   BETA_INV: 8
21   FUSION_CONV_CHANNEL_RATIO: 2
22
23 PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS
24
25 [02/02 15:02:49][INFO] logging.py: 98: json_stats: {"cur_iter": "590", "eta": "0:00:22", "split": "test_iter", "time_diff": 3.78435}
26 [02/02 15:02:53][INFO] logging.py: 98: json_stats: {"cur_iter": "551", "eta": "0:00:21", "split": "test_iter", "time_diff": 4.22652}
27 [02/02 15:02:57][INFO] logging.py: 98: json_stats: {"cur_iter": "552", "eta": "0:00:15", "split": "test_iter", "time_diff": 3.99075}
28 [02/02 15:03:01][INFO] logging.py: 98: json_stats: {"cur_iter": "553", "eta": "0:00:11", "split": "test_iter", "time_diff": 3.98216}
29 [02/02 15:03:05][INFO] logging.py: 98: json_stats: {"cur_iter": "554", "eta": "0:00:08", "split": "test_iter", "time_diff": 4.18329}
30 [02/02 15:03:09][INFO] logging.py: 98: json_stats: {"cur_iter": "555", "eta": "0:00:04", "split": "test_iter", "time_diff": 3.97948}
31 [02/02 15:03:09][INFO] logging.py: 98: json_stats: {"split": "test_final", "top1_acc": "54.09"}
32 [02/02 15:03:09][INFO] test_net.py: 249: Finalized testing with 10 temporal clips and 3 spatial crops
33 [02/02 15:03:09][INFO] test_net.py: 249: p33.65_f66.06_10a top1 Acc: 54.09 PER: 9.90 f: 66.0648
34 [02/02 15:03:09][INFO] test_net.py: 270: p33.65_f66.06_10a54.09
35
36 (AndikaRT) C:\Users\Admin\Documents\VP Andika Rahman Teja\SlowFast-main\
37
38 1648 Col 17 Space 2 UTF-8 LF (1) NAME Go Live Preview Emits Ctrl
```

Lampiran 8. Hasil Akurasi *SlowFast* 8x8 *stepwise*

[illegible]

Lampiran 9. Hasil Akurasi *SlowFast* NLN 4x16

```
slowfast_main_4x16_K60.py
47: MODEL:
48:   ARCH: slowfast
49:   MODEL_NAME: Slowfast
50:   LOSS_FUNC: cross_entropy
51:   DROPOUT_RATE: 0.5
52: TEST:
53:   ENABLE: True
54:   DATASET: kinetics
55:   BATCH_SIZE: 8
56: DATA_LOADER:
57:   NUM_WORKERS: 0
58:   PIN_MEMORY: True
59:   NUM_GPUS: 1
60:   NUM_SHARDS: 1
61:   RNG_SEED: 0
62:   OUTPUT_DIR: .
63:
64:
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS

```
[02/03 06:02:44][INFO] logging.py: 98: json_stats: {"cur_iter": "549", "eta": "0:00:26", "split": "test_iter", "time_diff": "3.79136"}
[02/03 06:02:47][INFO] logging.py: 98: json_stats: {"cur_iter": "550", "eta": "0:00:22", "split": "test_iter", "time_diff": "3.77712"}
[02/03 06:02:52][INFO] logging.py: 98: json_stats: {"cur_iter": "551", "eta": "0:00:20", "split": "test_iter", "time_diff": "4.06875"}
[02/03 06:02:56][INFO] logging.py: 98: json_stats: {"cur_iter": "552", "eta": "0:00:15", "split": "test_iter", "time_diff": "3.99902"}
[02/03 06:02:59][INFO] logging.py: 98: json_stats: {"cur_iter": "553", "eta": "0:00:11", "split": "test_iter", "time_diff": "3.97905"}
[02/03 06:03:01][INFO] logging.py: 98: json_stats: {"cur_iter": "554", "eta": "0:00:08", "split": "test_iter", "time_diff": "4.23037"}
[02/03 06:03:06][INFO] logging.py: 98: json_stats: {"cur_iter": "555", "eta": "0:00:03", "split": "test_iter", "time_diff": "3.95772"}
[02/03 06:03:08][INFO] logging.py: 98: json_stats: {"split": "test_final", "top_acc": "75.14"}
[02/03 06:03:08][INFO] test_net.py: 249: Finalized testing with 18 temporal clips and 3 spatial crops
[02/03 06:03:08][INFO] test_net.py: 249: p40.93 f48.16 th0 top1 Acc: 35.14 mIoU: 7.50 f: 48.16IoU
[02/03 06:03:08][INFO] test_net.py: 270: p40.93 f48.16 th0v.14

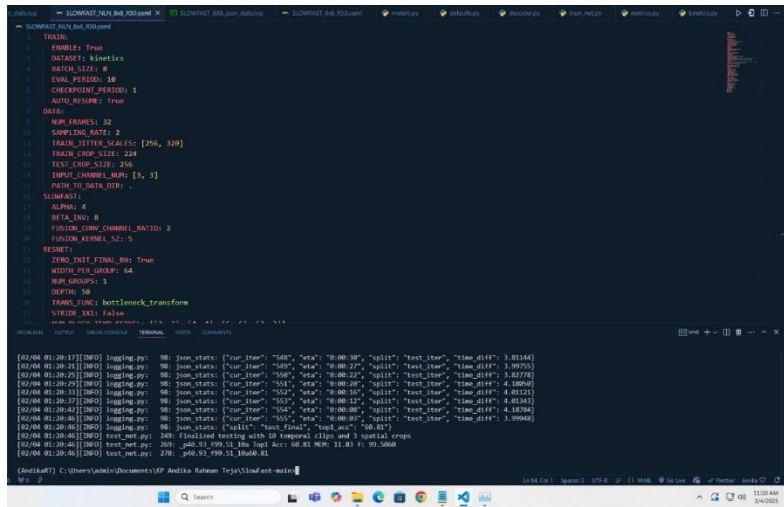
(AndikaRT) C:\Users\adhin\Documents\XP Andika Rahman Teja\Slowfast-main\]
```

WS P

La64, Col 1, Spans 2, UTF-8, LF, 1 VM, Go-Use, 1 of Printer, Endika VM

10:13 AM 2/2/2021

Lampiran 10. Hasil Akurasi *SlowFast* NLN 8x8



```

1 # SLOWFAST_NLN_8x8_Skipnet
2
3 TRAIN:
4   ENABLE: True
5   DATASET: kinetics
6   BATCH_SIZE: 8
7   EVAL_PERIOD: 10
8   CHECKPOINT_PERIOD: 1
9   AUTO_RESUME: True
10
11 DATA:
12   NUM_FRAMES: 32
13   SAMPLING_RATE: 2
14   TRAIN_ITER_SCALE: [256, 100]
15   TRAIN_CROP_SIZE: 224
16   TEST_CROP_SIZE: 224
17   INPUT_CHANNEL_NUM: [3, 3]
18   PATH_TO_DATA_DIR: .
19
20 SLOWFAST:
21   ALPHA: 4
22   BETA_INV: 8
23   FUSION_CONV_CHANNEL_RATIO: 2
24   FUSION_KERNEL_SZ: 5
25
26 RESET:
27   ZERO_INIT_FINAL_BIAS: True
28   WIDTH_FIX_GROUP: 64
29   NUM_GROUPS: 1
30   DEPTH: 50
31   TRANS_FUNC: nonlineack_transform
32   STRIDE_X3: False
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623

```

[Halaman ini sengaja dikosongkan]

BIODATA PENULIS

Nama : Andika Rahman Teja
Tempat, Tanggal Lahir : Surabaya, 4 Desember 2003
Jenis Kelamin : Laki-laki
Telepon : +6282143885331
Email : trahmanandika314@gmail.com

PENDIDIKAN FORMAL

2022 – sekarang : S1 Teknik Informatika ITS
2019 – 2022 : SMA Negeri 17 Surabaya
2016 – 2019 : SMP Negeri 35 Surabaya
2010 – 2016 : SD Negeri Wonorejo 274 Surabaya

AKADEMIS

Kuliah : Departemen Teknik Informatika –
FTEIC , ITS
Angkatan : 2022
Semester : 6 (Enam)
IPK : 3.87