

Smili

Sunni Widiarane

**OPTIMALISASI *HYPERPARAMETER* ARSITEKTUR *TEMPORAL*
CONVOLUTIONAL NETWORK MENGGUNAKAN METODE
HYPERBAND UNTUK MENGLASIFIKASI GERAKAN MATA
MANUSIA**

11/7
2023

SKRIPSI

Syukron A.I.A

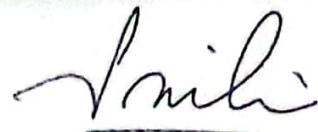
11/7 2023



Disusun oleh:


DAFFA BIL NADZARY
19/439811/TK/48541

**PROGRAM STUDI TEKNOLOGI INFORMASI
DEPARTEMEN TEKNIK ELEKTRO DAN TEKNOLOGI INFORMASI
FAKULTAS TEKNIK UNIVERSITAS GADJAH MADA
YOGYAKARTA
2023**


Sunu Wibirama.
11/7
2023.

HALAMAN PENGESAHAN

OPTIMALISASI *HYPERPARAMETER* ARSITEKTUR *TEMPORAL CONVOLUTIONAL NETWORK* MENGGUNAKAN METODE *HYPERBAND* UNTUK MENGLASIFIKASI GERAKAN MATA MANUSIA


Syukron A. I. A.
11/7 2023

SKRIPSI

Diajukan Sebagai Salah Satu Syarat untuk Memperoleh
Gelar Sarjana Teknik
pada Departemen Teknik Elektro dan Teknologi Informasi
Fakultas Teknik
Universitas Gadjah Mada

Disusun oleh:

DAFFA BIL NADZARY
19/439811/TK/48541

Telah disetujui dan disahkan

Pada tanggal 12 Juli 2023

Dosen Pembimbing I

Dosen Pembimbing II

Sunu Wibirama, Dr.Eng. Ir., S.T., M.Eng., IPM.
NIP 198510262015041003

Syukron Abu Ishaq Alfarozi, S.T., Ph.D.
NIP 111199205202101102

PERNYATAAN BEBAS PLAGIASI

Saya yang bertanda tangan di bawah ini :

Nama : Daffa Bil Nadzary
NIM : 19/439811/TK/48541
Tahun terdaftar : 2019
Program Studi : Teknologi Informasi
Fakultas : Teknik Universitas Gadjah Mada

Menyatakan bahwa dalam dokumen ilmiah Skripsi ini tidak terdapat bagian dari karya ilmiah lain yang telah diajukan untuk memperoleh gelar akademik di suatu lembaga Pendidikan Tinggi, dan juga tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang/lembaga lain, kecuali yang secara tertulis disitasi dalam dokumen ini dan disebutkan sumbernya secara lengkap dalam daftar pustaka.

Dengan demikian saya menyatakan bahwa dokumen ilmiah ini bebas dari unsur-unsur plagiasi dan apabila dokumen ilmiah Skripsi ini di kemudian hari terbukti merupakan plagiasi dari hasil karya penulis lain dan/atau dengan sengaja mengajukan karya atau pendapat yang merupakan hasil karya penulis lain, maka penulis bersedia menerima sanksi akademik dan/atau sanksi hukum yang berlaku.

Yogyakarta, 12 Juli 2023



Nama Mahasiswa
NIM

HALAMAN PERSEMBAHAN

Tugas akhir ini saya persembahkan kepada Almarhum Ayah beserta Ibu yang selalu mendukung dan mendoakan saya. Saya persembahkan pula tugas akhir ini kepada keluarga dan teman-teman semua, serta untuk bangsa, negara, dan agama yang saya abdi.

KATA PENGANTAR

Puji syukur ke hadirat Allah SWT atas limpahan rahmat, karunia, serta petunjuk-Nya sehingga tugas akhir berupa penyusunan skripsi ini telah terselesaikan dengan baik. Dalam hal 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. Bapak Ir. Hanung Adi Nugroho, S.T., M.Eng., Ph.D., IPM., selaku Kepala Departemen Teknik Elektro dan Teknologi Informasi Fakultas Teknik Universitas Gadjah Mada,
2. Bapak Ir. Lesnanto Multa Putranto, S.T., M.Eng., Ph.D., IPM. selaku Sekretaris Departemen Teknik Elektro dan Teknologi Informasi Fakultas Teknik Universitas Gadjah Mada,
3. Bapak Sunu Wibirama, Dr.Eng. Ir., S.T., M.Eng., IPM. selaku dosen pembimbing pertama dan Bapak Syukron Abu Ishaq Alfarozi, S.T., Ph.D. selaku dosen pembimbing kedua yang telah memberikan bantuan, bimbingan, arahan, serta ilmu kepada saya yang sangat bermanfaat pada tugas akhir ini,
4. Seluruh dosen dan tenaga pendidik Departemen Teknik Elektro dan Teknologi Informasi Fakultas Teknik Universitas Gadjah Mada yang menjadi bagian penting dari pengalaman menuntut ilmu dan pengetahuan saya di berbagai bidang selama delapan semester ini,
5. Almarhum Ayah saya, Andrea Zulfan, serta Ibu saya yang tercinta, Mariawati Cokogunanto, yang senantiasa mendidik, mendukung, membimbing, serta menyayangi saya sedari saya lahir hingga saat ini,
6. Rekan-rekan terdekat saya yang senantiasa membuat perkuliahan serta kehidupan saya menjadi lebih menyenangkan,
7. Serta seluruh pihak yang penulis tidak bisa sebutkan satu per satu yang telah memberikan bantuan dan dukungan selama pengerjaan tugas akhir ini.

Penulis menyadari secara penuh bahwa tugas akhir ini masih jauh dari kata sempurna. Untuk itu segala jenis saran, kritik serta masukan yang bersifat membangun pada tugas akhir ini sangat diharapkan. Akhir kata, penulis berharap bahwa skripsi ini dapat bermanfaat bagi banyak pihak, baik secara langsung maupun tidak langsung dalam menambah wawasan bagi para pembaca dan khususnya bagi penulis sendiri.

DAFTAR ISI

HALAMAN PENGESAHAN	ii
PERNYATAAN BEBAS PLAGIASI	iii
HALAMAN PERSEMBAHAN	iv
KATA PENGANTAR	v
DAFTAR ISI	vi
DAFTAR TABEL	ix
DAFTAR GAMBAR	x
DAFTAR SINGKATAN.....	xii
INTISARI.....	xiii
ABSTRACT	xiv
BAB I Pendahuluan	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	3
1.3 Tujuan Penelitian	4
1.4 Batasan Penelitian	4
1.5 Manfaat Penelitian	4
1.6 Sistematika Penulisan.....	5
BAB II Tinjauan Pustaka dan Dasar Teori	6
2.1 Tinjauan Pustaka	6
2.1.1 Penelitian Mengenai <i>Eye Movement Classification</i>	6
2.1.1.1 Pengklasifikasian Gerakan Mata Berbasis Ambang Ba- tas (<i>Threshold-based Algorithms</i>).....	6
2.1.1.2 Pengklasifikasian Gerakan Mata Berbasis Probabilitas (<i>Probability-based Algorithms</i>)	10
2.1.1.3 Pengklasifikasian Gerakan Mata Berbasis Data (<i>Data- driven algorithms</i>)	11
2.1.2 Penelitian Mengenai <i>Hyperparameter Optimization</i>	12
2.2 Dasar Teori	13
2.2.1 Fisiologi Gerakan Mata Manusia.....	13
2.2.2 Pelacakan Mata dan Akuisisi Data Gerakan Mata.....	13
2.2.2.1 Prinsip Dasar Pelacakan Mata	13
2.2.3 Ekstraksi Fitur Gerakan Mata	14
2.2.3.1 Kecepatan (<i>Speed</i>)	15
2.2.3.2 Arah (<i>Direction</i>)	15
2.2.3.3 Percepatan (<i>Acceleration</i>)	16
2.2.3.4 Perpindahan (<i>Displacement</i>)	16

2.2.3.5	Deviasi Standar (<i>Standard Deviation</i>)	16
2.2.4	Pembelajaran Dalam (<i>Deep Learning</i>)	17
2.2.4.1	<i>Artificial Neural Network</i>	18
2.2.4.2	Jaringan Saraf Umpan Maju dan Jaringan Saraf Maju Umpan Mundur (<i>Forward Propagation and Backpro- pagation</i>)	19
2.2.4.3	<i>Loss Function</i>	22
2.2.4.4	<i>Optimizer Algorithm</i>	23
2.2.4.5	<i>1 Dimensional-Convolutional Neural Network</i>	25
2.2.4.6	<i>Temporal Convolutional Network (TCN)</i>	27
2.2.5	Hyperparameter pada <i>Deep Learning</i>	28
2.2.5.1	<i>Dropout Rate</i>	28
2.2.5.2	<i>Filter Number</i>	29
2.2.5.3	<i>Kernel Size</i>	29
2.2.5.4	<i>Dilation Rate</i>	30
2.2.5.5	<i>Stacks Number</i>	30
2.2.5.6	<i>Activation Function</i>	31
2.2.5.7	<i>Batch Size</i>	32
2.2.5.8	<i>Epoch</i>	32
2.2.6	Hyperparameter Optimization (HPO)	32
2.2.6.1	<i>Hyperband</i>	34
2.2.7	Evaluasi Model	36
2.2.7.1	Metode Evaluasi	36
2.2.7.2	Metrik Evaluasi	37
2.3	Analisis Perbandingan Metode	39
BAB III Metode Penelitian		41
3.1	Alat dan Bahan Tugas Akhir	41
3.1.1	Alat Tugas Akhir	41
3.1.2	Bahan Tugas akhir	42
3.2	Metode yang Digunakan	43
3.2.1	Tahapan Penerapan Metode	44
3.3	Alur Tugas Akhir	44
3.3.1	Perolehan Data	45
3.3.2	Praproses Data	45
3.3.3	<i>Feature Engineering</i>	46
3.3.4	Agregasi Data	47
3.3.5	<i>Feature Extraction</i>	49
3.3.6	Pembuatan Arsitektur Model TCN	50
3.3.7	Proses <i>Hyperparameter Tuning</i>	51

3.3.8	Analisis <i>Hyperparameter</i> TCN.....	51
3.3.9	Analisis Hasil Perbandingan Metode	52
BAB IV Hasil dan Pembahasan.....		53
4.1	<i>Hyperparameter Tuning</i> dengan metode <i>Hyperband</i>	53
4.1.0.1	Evaluasi dengan Teknik LOVO	53
4.1.0.2	Evaluasi dengan Teknik <i>K-Fold Cross Validation</i>	58
4.2	Analisis <i>Hyperparameter</i> pada Model TCN.....	59
4.2.1	<i>Batch Size</i>	60
4.2.2	<i>Dropout Rate</i>	60
4.2.3	<i>Nb Filters</i>	61
4.2.4	<i>Nb Stacks</i>	62
4.2.5	<i>Kernel Size</i>	64
4.2.6	<i>Dilation Rate</i>	65
4.3	Perbandingan dengan Penelitian Sebelumnya	67
BAB V Kesimpulan dan Saran.....		70
5.1	Kesimpulan.....	70
5.2	Saran.....	70
DAFTAR PUSTAKA.....		71

DAFTAR TABEL

Tabel 2.1	Nilai-nilai umum dari gerakan-gerakan mata manusia. [1].....	6
Tabel 2.2	Tabel perbandingan metode riset <i>eye movement classification</i> dengan menggunakan <i>F1 Score</i>	39
Tabel 3.1	Deskripsi fitur pada dataset GazeCom [2].	43
Tabel 3.2	Deskripsi dari fitur-fitur hasil proses <i>feature engineering</i> pada dataset GazeCom.	47
Tabel 3.3	<i>Hyperparameter space</i> yang digunakan pada proses <i>hyperparameter tuning</i> menggunakan <i>Hyperband</i>	51
Tabel 4.1	Perbandingan nilai <i>hyperparameter</i> sebelum dan sesudah di- <i>tuning</i> menggunakan metode <i>Hyperband</i>	53
Tabel 4.2	Hasil evaluasi pada seluruh model hasil <i>tuning</i> metode <i>hyperband</i> dengan menggunakan teknik evaluasi LOVO.	54
Tabel 4.3	Hasil evaluasi pada seluruh model hasil <i>grid search</i> dengan menggunakan teknik evaluasi LOVO.	55
Tabel 4.4	Hasil evaluasi pada seluruh model dengan menggunakan teknik evaluasi <i>K-Fold Cross Validation</i>	59
Tabel 4.5	Perbandingan hasil evaluasi dari nilai-nilai pada <i>hyperparameter batch size</i> dalam performa TCN.	60
Tabel 4.6	Perbandingan hasil evaluasi dari nilai-nilai pada <i>hyperparameter dropout rate</i> dalam performa TCN.	61
Tabel 4.7	Perbandingan hasil evaluasi dari nilai-nilai pada <i>hyperparameter number of filters</i> dalam performa TCN.	62
Tabel 4.8	Perbandingan hasil evaluasi dari nilai-nilai pada <i>hyperparameter number of stacks</i> dalam performa TCN.....	63
Tabel 4.9	Perbandingan hasil evaluasi dari nilai-nilai pada <i>hyperparameter kernel size</i> dalam performa TCN.	65
Tabel 4.10	Perbandingan hasil evaluasi dari nilai-nilai pada <i>hyperparameter dilation rate</i> dalam performa TCN.....	66
Tabel 4.11	Perbandingan hasil evaluasi dari kedua model pada metode evaluasi LOVO dan <i>K-Fold Cross Validation</i>	68

DAFTAR GAMBAR

Gambar 2.1	<i>Pseudocode</i> dari algoritme I-VT. [3].....	8
Gambar 2.2	<i>Pseudocode</i> dari algoritme I-DT. [3].....	9
Gambar 2.3	<i>Pseudocode</i> dari algoritme I-VMP. [3]	10
Gambar 2.4	Representasi secara grafis dari ketiga gerakan mata <i>fixation</i> , <i>saccade</i> , dan <i>smooth pursuit</i> . Warna biru menandakan gerakan <i>fixation</i> , hijau menandakan <i>saccade</i> , kuning menandakan <i>smooth pursuit</i> , dan merah menandakan <i>noise</i> pada data.	13
Gambar 2.5	Contoh perangkat <i>eye tracker</i> jenis <i>head mounted</i> [4].	14
Gambar 2.6	<i>Layer</i> pada <i>deep learning</i> [5].	18
Gambar 2.7	Struktur dari satu unit <i>perceptron</i> dasar. [6].	18
Gambar 2.8	Contoh jaringan <i>feed-forward multilayer</i> dengan satu lapisan masukan, dua lapisan tersembunyi, serta satu lapisan luaran. [6]..	20
Gambar 2.9	Contoh jaringan ANN sederhana pada proses <i>backpropagation</i>	21
Gambar 2.10	Ilustrasi permasalahan <i>gradient descent</i> yang berada di <i>local minima</i>	24
Gambar 2.11	<i>Pseudocode</i> dari algoritme <i>optimizer</i> Adam. [7]	25
Gambar 2.12	Struktur arsitektur CNN. [8].	26
Gambar 2.13	Struktur arsitektur 1D CNN.	26
Gambar 2.14	Arsitektur <i>dilated convolution</i> kausal dengan faktor dilasi 1, 2, dan 4 dan ukuran kernel 3. [9].....	28
Gambar 2.15	Contoh ilustrasi penggunaan <i>dropout</i> pada <i>neural network</i> sebesar 50%.	29
Gambar 2.16	Struktur <i>residual block</i> pada TCN [10]	31
Gambar 2.17	<i>Pseudocode</i> dari algoritme <i>Bayesian optimization</i>	34
Gambar 2.18	<i>Pseudocode</i> dari algoritme <i>Hyperband optimization</i>	35
Gambar 2.19	Ilustrasi diagram dari metode evaluasi <i>K-Fold Cross Validation</i> dengan jumlah sebesar 10. [11]	37
Gambar 2.20	Ilustrasi dari <i>confusion matrix</i> . [12].....	38
Gambar 3.1	Contoh cuplikan dataset GazeCom pada salah satu <i>file</i>	43
Gambar 3.2	Alur dari tugas akhir.	45
Gambar 3.3	Contoh dari dataset hasil ekstraksi fitur baru.	47
Gambar 3.4	<i>Pseudocode</i> dari proses perhitungan <i>pixel per degree</i> (PPD).	48
Gambar 3.5	Proses ekstraksi fitur dengan menggunakan <i>temporal window</i> berukuran 257 dengan <i>overlap</i> sebesar 192. Kode F menandakan gerakan <i>fixation</i> , S menandakan <i>saccade</i> , dan SP menandakan <i>smooth pursuit</i>	49
Gambar 3.6	Contoh kerangka arsitektur model TCN yang dibuat dengan menggunakan 385 <i>temporal window</i> dengan jumlah fitur sebesar 28. ...	50
Gambar 4.1	Plot <i>confusion matrix</i> pada hasil evaluasi model TCN setelah proses <i>tuning</i> dengan <i>hyperband</i>	56
Gambar 4.2	Plot <i>confusion matrix</i> pada hasil evaluasi model TCN dengan menggunakan metode <i>grid search</i>	57
Gambar 4.3	Perbandingan kedua plot <i>history</i> proses <i>training</i> pada iterasi video <i>ducks_children</i>	58

Gambar 4.4	Perbandingan kedua plot <i>history</i> proses <i>training</i> pada iterasi video <i>St_petri_market</i>	58
Gambar 4.5	Perbandingan ketiga plot <i>history</i> pada proses <i>training</i> dengan ukuran <i>batch size</i> yang berbeda-beda.	60
Gambar 4.6	Perbandingan ketiga plot <i>history</i> pada proses <i>training</i> dengan ukuran <i>dropout rate</i> yang berbeda-beda.	61
Gambar 4.7	Perbandingan ketiga plot <i>history</i> pada proses <i>training</i> dengan jumlah filter yang berbeda-beda.	62
Gambar 4.8	Perbandingan kelima plot <i>history</i> pada proses <i>training</i> dengan ukuran jumlah <i>stacks</i> yang berbeda-beda.	64
Gambar 4.9	Perbandingan keenam plot <i>history</i> pada proses <i>training</i> dengan ukuran <i>kernel</i> yang berbeda-beda.	65
Gambar 4.10	Perbandingan keempat plot <i>history</i> pada proses <i>training</i> dengan <i>dilation rate</i> yang berbeda-beda.	67

DAFTAR SINGKATAN

TCN	=	<i>Temporal Convolutional Network</i>
1D-CNN-BLSTM	=	<i>1 Dimensional Convolutional Neural Network Bi-Directional Long Short-Term Memory</i>
RNN	=	<i>Recurrent Neural Network</i>
IMK	=	Interaksi Manusia dan Komputer
x dan y	=	Posisi koordinat gerakan mata dalam sumbu horizontal dan vertikal
I-VT	=	<i>Velocity-Threshold Identification</i>
I-DT	=	<i>Dispersion-Threshold Identification</i>
HMM	=	<i>Hidden Markov Model</i>
E	=	Semesta dari kelas pada model
FCN	=	<i>Fully Connected Network</i>
ReLU	=	<i>Rectified Linear Unit</i>
LOVO	=	<i>Leave One Video Out</i>
HPO	=	<i>Hyperparameter Optimization</i>
GPU	=	<i>Graphical Processing Unit</i>

INTISARI

Interaksi berbasis pandangan mata merupakan salah satu bentuk implementasi dari Industri 5.0. Sistem ini sangat bermanfaat untuk berbagai kondisi, terutama pada orang yang mengalami gangguan gerak. Implementasi dari sistem interaksi berbasis mata memerlukan sebuah sistem yang dapat mengenali dan mengklasifikasikan gerakan-gerakan mata yang berbeda pada manusia. Model *Temporal Convolutional Network* (TCN) merupakan model *state-of-the-art* dari klasifikasi gerakan mata ini. Pada tugas akhir ini, dilakukan peningkatan performa pada model TCN yang sudah ada dengan melakukan *hyperparameter optimization* menggunakan metode *Hyperband*. Penulis juga melakukan analisis dari keseluruhan *hyperparameter* yang ada secara individu untuk melihat pengaruh dari masing-masing *hyperparameter* terhadap model TCN. Hasil dari penelitian menunjukkan bahwa model TCN hasil proses *hyperparameter optimization* dapat mengalami peningkatan performa dengan menggunakan dua metode evaluasi yang berbeda. Evaluasi dengan metode LOVO menunjukkan peningkatan paling tinggi pada kelas gerakan *saccade* sebesar 1%, sedangkan evaluasi dengan metode *K-Fold Cross Validation* mengalami peningkatan yang cukup besar yakni 0,9% pada *fixation*, 1,3% pada *saccade*, 3,2% pada *noise* serta 6,6% pada *smooth pursuit*.

Kata kunci : *Eye movement classification, Temporal Convolutional Network, Hyperparameter Optimization, Hyperband, K-Fold Cross Validation*

ABSTRACT

Gaze-based interaction is one of many implementations of Industry 5.0. The system is highly beneficial in various conditions, especially for individuals with motion-impaired people. The implementation of an eye-based interaction system requires a system that can recognize and classify different types of eye movements in humans. In the eye movement classification field, Temporal Convolutional Network (TCN) model is proven to be a state-of-the-art model. In this research, performance improvement is conducted on the existing TCN model by optimizing hyperparameters using the Hyperband method. We also analyzed the individual hyperparameters to examine the influence of each hyperparameter on the TCN model. The research results show that the TCN model, after undergoing hyperparameter optimization, experiences performance improvement using two different evaluation methods. The LOVO method evaluation shows the highest increase in the saccade movement class by 1%, while the K-Fold Cross Validation evaluation experiences significant improvements, namely 0.9% in fixation, 1.3% in saccade, 3.2% in noise, and 6.6% in smooth pursuit.

Keywords : *Eye movement classification, Temporal Convolutional Network, Hyperparameter Optimization, Hyperband, K-Fold Cross Validation*

BAB I

PENDAHULUAN

1.1 Latar Belakang

Interaksi berbasis pandangan mata (*gaze-based Interaction*) merupakan salah satu pengaplikasian dari bidang ilmu Interaksi Manusia dan Komputer (IMK). Dengan perkembangan teknologi yang sangat pesat dalam beberapa tahun terakhir, IMK berperan sebagai jembatan pelengkap bagi terjadinya transisi dari Industri 4.0 ke Industri 5.0 [13]. Industri 5.0 lebih menekankan ke pada aspek sosial dari teknologi, yang berfokus pada peningkatan kolaborasi antar manusia dan mesin, hal yang tidak ditemukan pada Industri 4.0 [14].

Adanya penerapan *gaze-based interaction* merupakan salah satu bentuk dari implementasi Industri 5.0. Berbeda dengan perangkat masukan yang lain seperti papan tombol (*keyboard*), tetikus, hingga layar sentuh yang memerlukan gerakan fisik untuk dapat berinteraksi dengan mesin, interaksi berbasis pandangan mata ini sangat bermanfaat terutama untuk penderita gangguan gerak [15]. Sebagai tambahan, interaksi dengan *gaze-based input* juga lebih cepat dikarenakan gerakan mata umumnya merupakan gerakan utama yang dilakukan oleh manusia sebelum melakukan aksi lainnya dengan bagian tubuh yang lain [16].

Untuk dapat membuat sebuah sistem interaksi berbasis pandangan mata, salah satu hal yang diperlukan adalah algoritme yang dapat mengklasifikasikan gerakan-gerakan mata yang berbeda. Algoritme ini dikenal sebagai *eye movement classification* atau *events detection* [17]. *Eye movement classification* menjadi tugas yang tidak mudah dikarenakan berbagai hal, seperti data yang mengandung derau/*noise*, jenis-jenis pada gerakan mata yang beragam, hingga algoritme yang tidak mampu memproses data tersebut [18].

Seiring dengan perkembangan teknologi pembelajaran mesin (*machine learning*), berbagai penelitian terkait dengan *eye movement classification* mulai mengarah pada penerapan *machine learning* untuk mengklasifikasi gerakan-gerakan mata tersebut. Algoritme ini lebih dikenal sebagai *data-driven algorithms*. Pendekatan mula-mula masih berfokus pada penerapan metode *machine learning* tradisional. Vidal *et al.* [19] memperkenalkan sebuah model *K-Nearest Neighbor* untuk melakukan klasifikasi pada gerakan mata. Contoh lainnya terdapat pada Zemblys *et al.* [20], yang merancang model berbasis *random forest* untuk mendeteksi baik *fixation*, *saccade*, dan gerakan mata yang lain.

Pendekatan lebih lanjut mulai terlihat dengan adanya penerapan pembelajaran dalam (*deep learning*) untuk mengklasifikasi gerakan mata manusia. Dengan berbagai kelebihan yang ditawarkan oleh *deep learning* dibandingkan dengan metode *machine lear-*

ning tradisional membuat penelitian saat ini lebih berfokus pada metode *deep learning*. Hoppe dan Bulling [21] merancang arsitektur CNN untuk mengklasifikasikan gerakan *fixation*, *saccade*, dan *smooth pursuit*. Lanjut, Startsev *et al.* [22] memperkenalkan arsitektur 1D-CNN-BLSTM pada kasus ini. Beberapa metode *deep learning* yang ada dapat mengungguli hampir seluruh metode *machine learning* tradisional sebelumnya. Namun, hampir keseluruhan metode baik *machine learning* maupun *deep learning* memiliki kelemahan yang sama, yakni ketidakmampuan dalam mengenali gerakan *smooth pursuit*. Rancangan Startsev *et al.* dapat mengatasi permasalahan ini dengan mengekstrak dua fitur tambahan pada gerakan mata, yakni kecepatan (*speed*) dan arah (*direction*) pada model 1D-CNN-BLSTM yang dirancang. Alhasil, metode 1D-CNN-BLSTM mengalami peningkatan yang cukup signifikan terutama pada gerakan *smooth pursuit*. Pengembangan lebih lanjut pada model ini dilakukan oleh Elmadjian *et al.* [23], yang memperkenalkan arsitektur *Temporal Convolutional Network* (TCN). TCN dalam pengaplikasiannya, dapat mengalahkan model 1D-CNN-BLSTM terutama pada gerakan *smooth pursuit*, sehingga secara efektif menjadi *state-of-the art* saat ini. TCN, sebagaimana metode *deep learning* lainnya, dapat menghasilkan akurasi yang sangat tinggi. Selain itu, model *deep learning* lebih bersifat adaptif dibandingkan algoritme-algoritme sebelumnya.

Baik pada *machine learning* maupun *deep learning* memiliki sebuah parameter yang dapat diubah-ubah sebelum proses *training* dilakukan. Parameter ini akan secara langsung mempengaruhi baik struktur maupun konfigurasi dari arsitektur yang digunakan. Parameter ini dikenal dengan sebutan *hyperparameter* [24]. Agar dapat menemukan nilai-nilai *hyperparameter* yang optimal untuk sebuah model yang digunakan pada sebuah kasus, diperlukan sebuah metode optimalisasi yang disebut sebagai *hyperparameter optimization*. Pada implementasinya, metode *hyperparameter optimization* akan melakukan proses *training* pada model dengan mengubah-ubah nilai *hyperparameter* yang ditentukan pada tiap iterasi. Elmadjian *et al.* dalam penelitiannya merancang arsitektur TCN dengan menggunakan nilai-nilai *hyperparameter* yang dicari dengan menggunakan metode *grid search* [25]. *Grid search* dalam penerapannya merupakan algoritme yang bersifat *brute force*, yang berarti algoritme akan mencoba keseluruhan kombinasi dari *hyperparameter* yang ditentukan dalam sebuah *hyperparameter space*. Namun, *grid search* memiliki banyak kelemahan. Algoritme ini sangat memakan banyak sumber daya dan memerlukan daya komputasi yang sangat tinggi untuk dijalankan, terutama seiring bertambah luasnya *hyperparameter space* yang ditentukan.

Oleh karena itu, diperlukan metode *hyperparameter optimization* yang lain untuk mengoptimalkan *hyperparameter* yang terdapat pada model TCN yang sudah ada. Salah satu metode *hyperparameter optimization* yang umum digunakan dikarenakan performanya yang baik yaitu metode *Bayesian Optimization* [26]. *Bayesian Optimization* umum digunakan dikarenakan kemampuannya yang dapat konvergen ke nilai-nilai *hyperpara-*

meter yang optimal dengan jumlah iterasi yang jauh lebih sedikit dibandingkan dengan metode *grid search*. Meskipun *Bayesian Optimization* melampaui performa *grid search* di kebanyakan kondisi dengan menghasilkan nilai yang optimal, metode ini masih memiliki beberapa kelemahan, yakni penggunaan *objective function* yang identik digunakan dalam penerapan metode ini. Penggunaan *objective function* tersebut masih memakan sumber daya komputasi yang cukup besar. Meskipun jauh lebih sedikit dibandingkan dengan *grid search*, untuk kasus di mana dataset yang digunakan berukuran sangat besar, penggunaan *Bayesian Optimization* tetap berdampak besar pada penggunaan sumber daya komputasi yang besar, terutama jika terdapat keterbatasan dalam sumber daya komputasi yang digunakan.

Kelemahan yang terdapat pada metode *Bayesian Optimization* dapat ditanggulangi dengan menggunakan metode lain yang disebut sebagai *Hyperband* [27]. *Hyperband* dalam implementasinya jauh lebih efisien dibandingkan dengan *Bayesian Optimization*, dan sangat cocok digunakan pada kondisi di mana ukuran data yang digunakan sangat besar. Oleh karena itu, penelitian ini berfokus pada penggunaan metode *Hyperband*, yang merupakan metode *hyperparameter optimization* yang lebih optimal dengan penggunaan sumber daya komputasi yang jauh lebih sedikit.

Penelitian ini dilakukan untuk menganalisis perbedaan pada luaran model arsitektur TCN yang sudah ada setelah dilakukan *hyperparameter optimization* dengan menggunakan metode *Hyperband* yang bersifat lebih optimal dibandingkan metode *grid search* yang digunakan sebelumnya. Selain itu, penelitian ini juga menganalisis lebih dalam mengenai tiap *hyperparameter* yang digunakan pada model TCN.

1.2 Rumusan Masalah

TCN merupakan model arsitektur *deep learning* yang menjadi *state-of-the-art* pada kasus *eye movement classification* ini. Berdasarkan uraian pada subbab sebelumnya, maka rumusan masalah dari penelitian ini ialah:

- Arsitektur model TCN merupakan *state-of-the-art* untuk permasalahan *eye movement classification*. Namun, kinerja dari model masih dapat ditingkatkan dikarenakan belum dilakukannya *hyperparameter optimization* yang lebih optimal. Pada penelitian yang dilakukan oleh Elmadjian *et al.*, penggunaan *hyperparameter optimization* masih sebatas pada penggunaan metode *grid search* dengan rentang nilai *hyperparameter* yang kecil.
- Secara khusus, belum ada penelitian yang berfokus pada investigasi pencarian *hyperparameter* terutama pada model TCN, baik itu nilai *hyperparameter* yang optimal dalam sebuah permasalahan, maupun pengaruh dari tiap *hyperparameter* yang ada.

1.3 Tujuan Penelitian

Sasaran yang diharapkan untuk dipenuhi setelah penelitian dinyatakan sebagai berikut.

- Menganalisis kinerja model TCN sebelum dan sesudah dilakukan *hyperparameter optimization* dengan menggunakan metode *hyperband*. Beberapa aspek yang dianalisis yaitu hasil evaluasi model, perbandingan nilai *hyperparameter*, serta kinerja kedua metode *hyperband* serta *grid search*.
- Menganalisis tiap *hyperparameter* yang digunakan pada model TCN beserta dengan pengaruhnya.

1.4 Batasan Penelitian

1. Objek penelitian: Analisis kinerja dari model arsitektur TCN serta pengaruh *hyperparameter optimization*.
2. Metode penelitian: Penelitian yang bersifat eksperimental menggunakan *platform* perancangan model berbasis kode pemrograman.
3. Waktu dan tempat penelitian: Januari hingga Juni 2023 di Fasilitas Lab Instrumentasi dan Kendali Departemen Teknik Elektro dan Teknologi Informasi Universitas Gadjah Mada.
4. Hipotesis: Hipotesis yang ingin dibuktikan pada penelitian ini adalah sebagai berikut.
 - (a) Optimalisasi *hyperparameter* menggunakan metode *hyperband* berpengaruh pada hasil evaluasi model arsitektur TCN secara keseluruhan.
 - (b) Metode *hyperband* bersifat lebih optimal baik dari proses pencarian nilai-nilai *hyperparameter* maupun dari penggunaan sumber daya komputasi dibandingkan dengan metode *grid search* pada model TCN.
5. Keterbatasan Penelitian: Keterbatasan dari penelitian adalah sebagai berikut.
 - Keseluruhan proses *tuning*, *training*, serta evaluasi model dilakukan pada *platform* berbasis *cloud* yang tidak bersifat gratis untuk digunakan.
 - Sistem klasifikasi yang dirancang merupakan sistem non kausal, sehingga tidak bersifat *real-time*.

1.5 Manfaat Penelitian

Setelah penelitian selesai dilakukan, diharapkan dapat memberikan manfaat ke pada dunia akademik serta bagi masyarakat sebagai berikut.

1. Bagi dunia akademik, hasil penelitian dapat digunakan sebagai dasar untuk melakukan penelitian lebih lanjut baik pada tiap *hyperparameter* maupun pada arsitektur model TCN itu sendiri secara mendalam.
2. Bagi masyarakat, hasil penelitian dapat digunakan sebagai dasar untuk mengembangkan aplikasi interaktif berbasis pandangan mata (*gaze-based interactive application*) dengan kinerja sistem pendeteksi gerakan mata yang lebih baik.

1.6 Sistematika Penulisan

Secara garis besar, penulisan tersusun dari lima bab utama sebagai berikut.

Bab I Pendahuluan memaparkan latar belakang dari penelitian mulai dari algoritme klasifikasi gerakan mata sebagai implementasi dari *gaze-based interaction* maupun perkembangan pada metode *hyperparameter optimization* yang ada, untuk kemudian ditarik permasalahan masalah darinya. Pada bab ini juga dijelaskan tujuan dari penelitian, batasan-batasan yang ada, serta manfaat dari penelitian.

Bab II Tinjauan Pustaka dan Dasar Teori menjabarkan secara spesifik tinjauan pustaka yang membahas dan menganalisis publikasi serta penelitian-penelitian terdahulu terkait dengan penelitian yang dilakukan, terutama mengenai *eye movement classification*. Selain itu, dijabarkan juga dasar-dasar teori yang mendasari penelitian terkait, seperti gerakan mata manusia, *deep learning*, *hyperparameter optimization*, hingga *Hyperband* sebagai metode yang menjadi fokus pada penelitian.

Bab III Metode Penelitian menjelaskan alat dan bahan yang digunakan pada penelitian, baik itu *hardware*, *software*, maupun dataset. Rancangan dan alur penelitian yang dilakukan juga akan dibahas pada bab ini.

Bab IV Hasil dan Pembahasan membahas tentang hasil penelitian yang dilakukan. Hasil penelitian akan ditampilkan pada bab ini untuk dibahas. Hasil yang dibahas meliputi hasil evaluasi model TCN dengan menggunakan dua metode evaluasi, serta analisis dari masing-masing *hyperparameter* yang terdapat pada model TCN.

Bab V Kesimpulan dan Saran merupakan bab penutup yang berisi kesimpulan dari penelitian serta temuan yang ditemukan, serta saran pengembangan untuk penelitian ke depan.

BAB II

TINJAUAN PUSTAKA DAN DASAR TEORI

2.1 Tinjauan Pustaka

2.1.1 Penelitian Mengenai *Eye Movement Classification*

Penelitian mengenai pengklasifikasian gerakan mata sudah dilakukan sejak lama [22]. Meskipun terdapat banyak sekali gerakan mata yang dapat dilakukan oleh manusia [28], *fixation*, *saccade*, dan *smooth pursuit* merupakan tiga jenis gerakan yang paling umum digunakan terutama pada riset pengklasifikasian gerakan mata [3]. *Fixation* merupakan jenis gerakan mata di mana kondisi mata terfokus ke suatu titik untuk jangka waktu tertentu. *Saccade* merupakan gerakan mata yang bersifat cepat secara instan berpindah dari satu titik ke titik yang lain. Sedangkan *smooth pursuit* merupakan gerakan mata yang bergerak dengan kecepatan lebih lambat dibandingkan *saccade*, yang mensimulasikan gerakan yang mengikuti suatu titik yang bergerak secara perlahan [1]. Tabel 2.1 menunjukkan perbedaan nilai pada masing-masing gerakan mata manusia, baik pada durasi, amplitudo, maupun kecepatan dari gerakan.

Tabel 2.1. Nilai-nilai umum dari gerakan-gerakan mata manusia. [1]

Jenis gerakan	Durasi (ms)	Amplitudo	Kecepatan
<i>Fixation</i>	200-300	-	-
<i>Saccade</i>	30-80	4-20°	30-500°/detik
<i>Smooth pursuit</i>	-	-	10-30°/detik

Beberapa algoritme sudah diperkenalkan untuk dapat mendeteksi dan mengklasifikasikan ketiga gerakan mata ini. Untuk jangka waktu yang lama, terdapat dua kelas algoritme yang digunakan: algoritme berbasis batas (*threshold-based algorithms*), dan algoritme berbasis probabilitas (*probability-based algorithms*) [29]. Kemudian dalam beberapa tahun terakhir, algoritme dengan pendekatan berbasis data pun muncul, sehingga saat ini terdapat tiga pendekatan utama dalam pengklasifikasian gerakan mata ini.

2.1.1.1 Pengklasifikasian Gerakan Mata Berbasis Ambang Batas (*Threshold-based Algorithms*)

Karakteristik utama dari metode berbasis algoritme ini adalah adanya penggunaan ambang batas (*threshold*) untuk mendefinisikan suatu gerakan pada mata. Ambang batas pada metode ini umumnya ditentukan pada sebuah fitur spesifik, seperti kecepatan dari gerakan mata *speed*, atau besar dari gerakan mata itu sendiri *movement*. Dalam

perkembangannya, terdapat tiga metode hasil implementasi dari penggunaan *threshold* ini, di mana fitur yang digunakan untuk diberi *threshold* berbeda-beda. Implementasi ini digunakan pada tiga metode: *Velocity Threshold Identification* (I-VT), *Movement Pattern Identification* (I-VMP), dan *Dispersion Threshold Identification* (I-DT) [3].

I-VT merupakan metode yang termasuk ke dalam jenis algoritme berbasis kecepatan (*velocity-based algorithms*) [30]. *Velocity-based algorithms* merupakan algoritme di mana proses klasifikasi gerakan mata dilakukan berdasarkan kecepatan dari pergerakan mata tersebut. Gerakan mata yang melampaui batas kecepatan tertentu akan diklasifikasikan antara sebagai *saccade* atau *smooth pursuit*, dan sisanya sebagai *fixation*. Pada metode ini, terdapat dua *threshold* yang didefinisikan terlebih dahulu. *Threshold* pertama akan menentukan apakah gerakan tersebut termasuk ke dalam *saccade* atau tidak. *Threshold* pertama merupakan *threshold* yang paling tinggi. Jika kecepatan yang terdeteksi berada di atas *threshold* pertama yang ditetapkan, maka gerakan mata akan dianggap sebagai *saccade*, sedangkan jika berada di bawah *threshold* pertama, maka akan dilanjutkan dengan perbandingan dengan *threshold* kedua. Berbeda dengan *threshold* pertama, *threshold* kedua akan menentukan apakah gerakan tersebut termasuk ke dalam *smooth pursuit* atau tidak. Jika kecepatan yang terdeteksi berada di bawah *threshold* pertama namun berada di atas *threshold* kedua, maka gerakan mata akan dianggap sebagai *smooth pursuit*. Sebaliknya, jika kecepatan berada di bawah *threshold* kedua, maka gerakan mata akan dianggap sebagai *fixation*.

Algoritme 1 *Velocity Threshold Identification*

Masukan: *array* dari titik posisi mata, *threshold* gerakan *saccade* T_{Vs} , *threshold* gerakan *smooth pursuit* T_{Vp} .

Luaran: label *fixation*, *saccade*, atau *smooth pursuit* pada data.

Algoritme:

for titik in data **do**

 Hitung kecepatan V_n dari titik ke titik

if $V_n > \text{threshold saccade } T_{Vs}$ **then**

 label titik = *saccade*

end if

if $V_n < \text{threshold saccade } T_{Vs}$ & $V_n > \text{threshold smooth pursuit } T_{Vp}$ **then**

 label titik = *smooth pursuit*

end if

if $V_n < \text{threshold smooth pursuit } T_{Vp}$ **then**

 label titik = *fixation*

end if

end for

Return: titik dengan label *fixation*, *smooth pursuit*, dan *saccade*.

Gambar 2.1. *Pseudocode* dari algoritme I-VT. [3]

Berbeda dengan I-VT, I-DT merupakan jenis metode yang termasuk ke dalam algoritme berbasis persebaran (*dispersion-based algorithms*) [31]. Sama layaknya metode I-VT, metode I-DT juga menggunakan *threshold* untuk mengklasifikasikan gerakan mata yang terdeteksi. Berbeda dengan I-VT yang menggunakan kecepatan sebagai fitur pembeda, I-DT menggunakan persebaran (*dispersion*) dari data untuk diklasifikasikan sebagai *saccade*, *smooth pursuit*, dan *fixation*. Pengecualian terdapat pada gerakan *saccade*, di mana fitur kecepatan *velocity* masih digunakan sebagai indikator apakah gerakan mata tersebut termasuk ke dalam *saccade* atau tidak. Jika kecepatan dari gerakan mata berada di bawah batas *threshold* pertama, maka selanjutnya akan dikomparasi dari segi *dispersion* untuk membedakan antara *smooth pursuit* dengan *fixation*.

Algoritme 2 *Dispersion Threshold Identification*

Masukan: array dari titik posisi mata, *threshold* kecepatan T_V , *threshold* persebaran (*dispersion*) T_D , serta ukuran *temporal window* T_W .

Luaran: label *fixation*, *saccade*, atau *smooth pursuit* pada data.

Algoritme:

for titik in data **do**

 Hitung kecepatan V_n dari titik ke titik

if $V_n > \text{threshold}$ kecepatan T_V **then**

 label titik = *saccade*

end if

end for

Inisialisasi *temporal window* dari titik-titik pertama pada gerakan yang belum dilabeli

while *temporal window* belum mencapai ujung dari array **do**

 Hitung persebaran (*dispersion*) D_n dalam satu window

if $D_n < \text{threshold}$ persebaran T_D **then**

while $D_n < \text{threshold}$ persebaran T_D **do**

 Tambah satu titik lain yang belum teridentifikasi ke dalam window

 Hitung ulang *dispersion* dalam window

end while

 label keseluruhan titik = *fixation*

 clear window

else

 Hapus titik pertama dari window

 label titik pertama = *smooth pursuit*

end if

end while

Return: titik dengan label *fixation*, *smooth pursuit*, dan *saccade*.

Gambar 2.2. *Pseudocode* dari algoritme I-DT. [3]

Yang terakhir adalah I-VMP. I-VMP juga tidak jauh berbeda dengan I-DT, di mana kedua metode sama-sama menggunakan *velocity threshold* untuk mengidentifikasi gerakan mata *saccade*. Berbeda dengan I-DT yang menggunakan *dispersion* untuk membedakan antara *smooth pursuit* dengan *fixation*, I-VMP menggunakan gerakan mata (*movement*) untuk membedakan antara kedua gerakan mata tersebut. I-VMP akan menghitung besar dari gerakan yang dibentuk pada mata (*magnitude*). Secara umum, I-VMP memberikan hasil evaluasi yang lebih baik dibandingkan dengan I-VT [32].

Algoritme 3 *Movement Threshold Identification*

Masukan: array dari titik posisi mata, *threshold* kecepatan T_V , *threshold* gerakan (*movement*) T_M , serta ukuran *temporal window* T_W .

Luaran: label *fixation*, *saccade*, atau *smooth pursuit* pada data.

Algoritme:

for titik in data **do**

 Hitung kecepatan V_n dari titik ke titik

if $V_n > \text{threshold}$ kecepatan T_V **then**

 label titik = *saccade*

end if

end for

while *temporal window* belum mencapai ujung dari array **do**

 Label seluruh titik yang belum teridentifikasi = *fixation*

for seluruh pasang titik yang berdekatan in *temporal window* T_W **do**

 Hitung sudut yang terbentuk dari pasangan titik terhadap sumbu horizontal

 Representasikan sudut yang terbentuk sebagai titik dalam koordinat lingkaran

end for

 Hitung rerata (*mean*) dari gerakan di koordinat x dan y (M_X dan M_Y) dari titik-titik yang didapat

if jarak antara rerata dan koordinat (0, 0) $> T_M$ **then**

 label titik = *smooth pursuit*

end if

end while

Return: titik dengan label *fixation*, *smooth pursuit*, dan *saccade*.

Gambar 2.3. *Pseudocode* dari algoritme I-VMP. [3]

Permasalahan muncul pada algoritme ini, di mana ketiga jenis algoritme sangat bergantung pada nilai batas (*threshold*) yang ditentukan [31]. Perlu ada konfigurasi secara manual oleh operator manusia dan besaran *threshold* yang ditetapkan tidak memungkinkan untuk berubah secara adaptif pada saat implementasi klasifikasi gerakan mata. Permasalahan lainnya muncul saat terdapat *noise* pada data, yang tidak bisa dideteksi secara baik oleh ketiga algoritme.

2.1.1.2 Pengklasifikasian Gerakan Mata Berbasis Probabilitas (*Probability-based Algorithms*)

Permasalahan yang muncul pada algoritme berbasis *threshold* membuat algoritme lain muncul sebagai solusi. Salah satunya adalah algoritme berbasis probabilitas

(*probability-based algorithms*), yang lebih sensitif terhadap *noise* [29]. Pengklasifikasi kelas pada metode ini dilakukan berdasarkan probabilitas [29]. Terdapat beberapa metode klasifikasi mata yang termasuk ke dalam jenis algoritme berbasis probabilitas ini, seperti *Naive Segmented Linear Regression-Hidden Markov Model* (NSLR-HMM) [33] dan *Bayesian Decision Theory Identification* (I-BDT) [29].

NSLR-HMM merupakan model berbasis probabilitas yang bertujuan untuk diimplementasikan pada kondisi data yang mengandung tingkat *noise* yang tinggi, yang sangat sesuai dengan kelemahan dari kebanyakan model berbasis *threshold* [33]. Sinyal gerakan mata yang masuk akan diurai menjadi beberapa segmen sinyal independen dan kemudian diterapkan sistem regresi linear. Pada metode ini, model *Hidden Markov* (*Hidden Markov Model*) digunakan sebagai *classifier*, dengan tiga hingga empat jenis gerakan mata yang akan diprediksi.

Sedangkan I-BDT merupakan metode yang juga digunakan untuk mengklasifikasikan gerakan mata secara langsung (*real-time*). Metode ini memanfaatkan Teori Keputusan Bayesian (*Bayesian Decision Theory*) [34] dalam proses klasifikasinya. Untuk data D , maka probabilitas untuk setiap kelas yang ada dapat didefinisikan sebagai berikut

$$f(x) = \frac{p(e)p(D|e)}{p(D)} \quad (2-1)$$

di mana $p(e)$ disebut sebagai *prior probability*, yang berarti merupakan probabilitas terdapatnya kelas e dalam populasi E , dengan $e \in E$. E merupakan keseluruhan kelas yang terdapat pada model, yang di mana dalam kasus ini adalah jenis gerakan pada mata yang diamati (*fixation*, *saccade*, dan *smooth pursuit*).

Meskipun algoritme berbasis probabilitas ini dapat mengatasi permasalahan yang terdapat pada algoritme berbasis *threshold*, algoritme ini tetap memiliki kelemahan yang dimiliki oleh algoritme tradisional lainnya, yaitu keterbatasan dalam pemilihan fitur, baik yang terlihat seperti fitur-fitur yang mendeskripsikan data maupun tidak, seperti fitur-fitur yang diekstrak secara khusus oleh sebuah model.

2.1.1.3 Pengklasifikasian Gerakan Mata Berbasis Data (*Data-driven algorithms*)

Data-driven algorithms merupakan jenis algoritme yang muncul seiring berkembangnya metode *machine learning* baik untuk menyelesaikan permasalahan klasifikasi (*supervised learning*), ataupun untuk permasalahan kluster/*clustering* (*unsupervised learning*). Oleh karena itu, *machine learning* juga dapat dijadikan sebagai sarana untuk mengklasifikasikan gerakan mata manusia. Perkembangan dari *machine learning* ini mengakibatkan proses pengklasifikasian dari gerakan-gerakan mata menjadi dapat dilakukan secara otomatis tanpa campur tangan manusia [22].

Perkembangan dari *eye movement classification* membuat banyaknya algoritme

machine learning yang diterapkan pada kasus tersebut. Zemblys *et al.* [35] melakukan komparasi antara sepuluh algoritme *machine learning* untuk mengklasifikasi gerakan mata manusia, seperti *K-Nearest Neighbor*, *Gradient Boosting*, hingga *Random Forest*. Selain penggunaan *machine learning*, penggunaan *deep learning* juga mulai digunakan dikarenakan kemampuannya yang dapat mencapai akurasi yang jauh lebih baik dibandingkan dengan *machine learning*. Selain itu, *machine learning* memiliki beberapa kelemahan seperti pada proses *feature engineering*, di mana model *machine learning* masih bergantung pada fitur yang diekstrak secara manual, serta skalabilitas yang lebih rendah dibandingkan dengan *deep learning*, yang sangat cocok digunakan terutama jika ukuran data sangat besar. Hoppe *et al.* [21] merancang sebuah arsitektur *Convolutional Neural Network* (CNN) sederhana dengan jumlah satu lapis (*layer*), yang dapat memberikan hasil yang lebih baik dibandingkan metode-metode *machine learning* sebelumnya dengan penggunaan dua fitur masukan utama, yakni kecepatan (*velocity*) dan persebaran (*dispersion*).

Lebih lanjut, Startsev *et al.* [22] memperkenalkan arsitektur 1D-CNN-BLSTM, campuran dari 1D *Convolutional Neural Network* dengan arsitektur *Bidirectional Long Short-Term Memory*. Hasilnya menunjukkan bahwa model ini mampu menghasilkan akurasi yang melampaui keseluruhan model *deep learning* serta *machine learning* yang ada. Hingga kemudian Elmadjan *et al.* [23] merancang arsitektur TCN yang menjadi *state-of-the-art* pada *eye movement classification* ini. Dibandingkan dengan model seperti LSTM, TCN memiliki beberapa kelebihan, seperti memori yang lebih panjang dikarenakan arsitektur TCN tidak memiliki mekanisme *gating* seperti yang terdapat pada LSTM. Selain itu, TCN memiliki struktur konvolusional pada lapisannya. Dengan jumlah memori yang diperlukan untuk proses latihan (*training*) yang lebih rendah, struktur konvolusional ini juga mengakibatkan arsitektur menjadi lebih mudah untuk diparalelkan, sehingga waktu yang diperlukan dalam proses *training* menjadi lebih cepat [36].

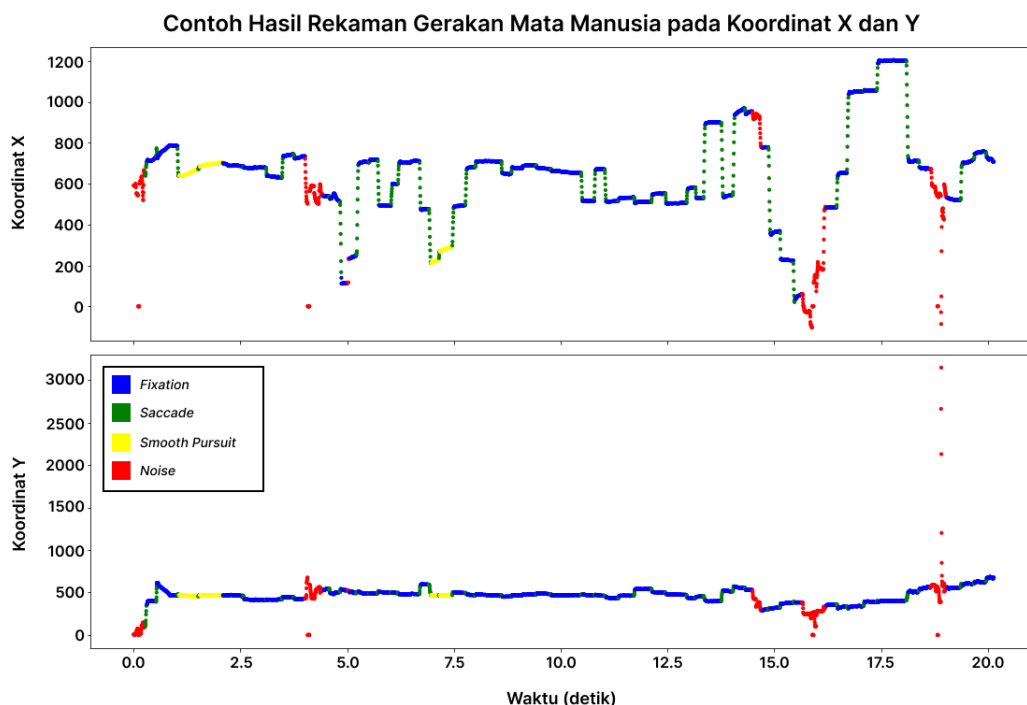
2.1.2 Penelitian Mengenai *Hyperparameter Optimization*

Selain berkembangnya penelitian mengenai *eye movement classification*, penelitian pada *hyperparameter optimization* (HPO) juga terus dilakukan. Hal ini bertujuan agar dapat diciptakan metode yang dapat mencari nilai-nilai *hyperparameter* pada sebuah model *machine learning* maupun *deep learning* secara otomatis. Terdapat banyak sekali riset mengenai metode HPO yang digunakan pada model *deep learning*. Permasalahan pada HPO dibahas secara umum oleh Yu dan Zhu [24], yang meliputi *hyperparameter* yang umum dicari, algoritme-algoritme optimisasi yang umum, serta pengaplikasiannya pada *deep learning*. Lanjut, Ashraf *et al* [37]. menyediakan peninjauan mendalam mengenai algoritme *evolutionary* dan *swarm intelligence* untuk mengoptimisasi *hyperparameter* terutama pada model *deep learning* dengan ukuran data yang sangat besar (*big data*).

2.2 Dasar Teori

2.2.1 Fisiologi Gerakan Mata Manusia

Sebagian besar gerakan mata pada manusia dikontrol oleh sebuah saraf yang disebut sebagai saraf okulomotor [38]. Saraf ini merupakan saraf kranial yang juga berfungsi sebagai sarana untuk mengontrol kontraksi dari pupil, mengakibatkan mata untuk dapat fokus ke suatu objek. Baik *fixation*, *saccade*, maupun *smooth pursuit* merupakan jenis gerakan mata yang umumnya digunakan baik oleh manusia maupun jenis hewan vertebrata lainnya untuk melacak objek dalam pandangan. Umumnya mata hanya melakukan gerakan *fixation* dan *saccade* secara bergantian saat melihat sekitar. *Smooth pursuit* merupakan kasus spesial di mana gerakan ini terjadi saat mata manusia terfokus pada suatu objek yang bergerak.



Gambar 2.4. Representasi secara grafis dari ketiga gerakan mata *fixation*, *saccade*, dan *smooth pursuit*. Warna biru menandakan gerakan *fixation*, hijau menandakan *saccade*, kuning menandakan *smooth pursuit*, dan merah menandakan *noise* pada data.

2.2.2 Pelacakan Mata dan Akuisisi Data Gerakan Mata

2.2.2.1 Prinsip Dasar Pelacakan Mata

Gerakan yang dibentuk pada mata dapat ditangkap dengan menggunakan teknologi *eye tracking*. *Eye tracking* menggunakan teknologi sensor untuk melacak gerakan mata dan mencocokkannya dengan koordinat pada layar, membuat perangkat *eye tracking* dapat mengetahui lokasi mata, serta menentukan fokus dan keberadaan dari mata [39].

Secara umum, terdapat dua jenis perangkat *eye tracking* yang dapat digunakan, yaitu *head mounted* dan *remote*. *Head mounted* merupakan jenis *eye tracker* yang digunakan oleh manusia. Bentuk dari jenis ini berbeda-beda, mulai dari kacamata, hingga pita mengelilingi kepala. Sedangkan jenis *remote* merupakan jenis yang memerlukan partisipan untuk duduk di depan sebuah layar untuk berinteraksi dengan diberi stimulus atau konten visual [40].

Berdasarkan hasil tangkapan dari *eye tracker*, informasi maupun data yang diperoleh dapat diolah lebih lanjut untuk pengembangan berbagai jenis aplikasi, salah satunya yaitu klasifikasi gerakan mata. Data yang dihasilkan oleh *eye tracker* ini berupa koordinat dari gerakan mata dalam posisi x dan y (horizontal dan vertikal). Selain itu *eye tracker* juga menghasilkan data runtun atau *timestamp*, sehingga menandakan bahwa data merupakan data *time series*.



Gambar 2.5. Contoh perangkat *eye tracker* jenis *head mounted* [4].

Data hasil *fixation* umumnya diidentifikasi sebagai sekumpulan titik-titik yang berada dekat baik dari segi waktu maupun posisi. Umumnya durasi dari data *fixation* berada di antara 100 hingga 300 milisekon. Pada *smooth pursuit* umumnya diidentifikasi oleh adanya perpindahan koordinat baik secara horizontal, vertikal, ataupun dua-duanya yang terjadi secara instan (dalam jangka waktu yang sangat pendek).

2.2.3 Ekstraksi Fitur Gerakan Mata

Data hasil tangkapan dari *eye tracker* dapat diolah lebih lanjut untuk menghasilkan fitur-fitur yang dapat lebih merepresentasikan masing-masing jenis gerakan mata. Dalam pengembangan model klasifikasi gerakan mata, Startsev *et al.* [22] menambahkan beberapa fitur tambahan yang diekstrak dari data koordinat serta waktu yang tercatat dari

hasil *eye tracker*. Fitur-fitur yang diekstrak adalah kecepatan (*speed*), akselerasi (*acceleration*), dan arah (*direction*) dari tiap titik. Dikarenakan data *eye tracker* merupakan jenis data *time series* yang memiliki informasi temporal dari tiap rekaman data, ketiga fitur gerakan tersebut diekstrak dengan bantuan kerangka temporal (*temporal time windows*). Masing-masing fitur tersebut diekstrak dengan menggunakan berbagai ukuran *temporal windows*, dengan tujuan untuk merepresentasikan hubungan antar titik data dengan lebih baik. Lanjut, Elmadjian *et al.* [23] juga menginvestigasi penambahan dua fitur lain dalam performa model klasifikasi gerakan mata, yaitu deviasi standar (*standard deviation*) dan perpindahan (*displacement*).

2.2.3.1 Kecepatan (*Speed*)

Kecepatan dapat didefinisikan sebagai laju perubahan dari sebuah benda berpindah. Fitur kecepatan dari tiap data dapat dihitung sebagai berikut

$$speed_i = \sqrt{\frac{(x_b - x_a)^2 + (y_b - y_a)^2}{\Delta t}} \quad (2-2)$$

di mana:

$speed_i$ = Kecepatan yang ditempuh dari dua titik tepi dalam satu *temporal window*,

x_b dan x_a = Posisi akhir dan awal titik dalam satu *temporal window* dalam koordinat x (horizontal),

y_b dan y_a = Posisi akhir dan awal titik dalam satu *temporal window* dalam koordinat y (vertikal),

Δt = Perubahan waktu yang terjadi dalam satu *temporal window*.

2.2.3.2 Arah (*Direction*)

Arah dapat didefinisikan sebagai sudut yang dibentuk dari perubahan posisi yang terjadi antara kedua sumbu horizontal dan vertikal. Fitur arah dari tiap data dapat dihitung sebagai berikut

$$direction_i = \tan^{-1}((x_b - x_a)^2 + (y_b - y_a)^2) \quad (2-3)$$

di mana:

$direction_i$ = Arah yang ditempuh dari dua titik tepi dalam satu *temporal window* (dalam pi radian),

x_b dan x_a = Posisi akhir dan awal titik dalam satu *temporal window* dalam koordinat x (horizontal),

y_b dan y_a = Posisi akhir dan awal titik dalam satu *temporal window* dalam koordinat y (vertikal).

2.2.3.3 Percepatan (*Acceleration*)

Percepatan dapat didefinisikan sebagai laju perubahan dari kecepatan benda dalam selang waktu tertentu. Percepatan juga dapat dikatakan sebagai turunan dari vektor kecepatan terhadap waktu. Fitur percepatan dari tiap data dapat dihitung sebagai berikut

$$acceleration_i = \sqrt{\left(\frac{(v_x^b - v_x^a)^2}{\Delta t}\right)^2 + \left(\frac{(v_y^b - v_y^a)^2}{\Delta t}\right)^2} \quad (2-4)$$

di mana:

$acceleration_i$ = Percepatan yang ditempuh dari dua titik tepi dalam satu *temporal window*,

v_x^b dan v_x^a = Komponen kecepatan akhir dan awal dalam satu *temporal window* dalam koordinat x (horizontal),

v_y^b dan v_y^a = Komponen kecepatan akhir dan awal dalam satu *temporal window* dalam koordinat y (vertikal),

Δt = Perubahan waktu yang terjadi dalam satu *temporal window*.

2.2.3.4 Perpindahan (*Displacement*)

Perpindahan dapat didefinisikan sebagai besar perubahan dari posisi awal dan akhir yang terjadi antara kedua sumbu horizontal dan vertikal. Fitur perpindahan dari tiap data dapat dihitung sebagai berikut

$$displacement_i = \sqrt{\left(\sum_{n=a}^b (x_{n+1} - x_n)\right)^2 + \left(\sum_{n=a}^b (y_{n+1} - y_n)\right)^2} \quad (2-5)$$

di mana:

$displacement_i$ = Perpindahan yang ditempuh dari dua titik tepi dalam satu *temporal window*,

x_n dan y_n = Posisi data di indeks n pada sumbu x (horizontal) dan y (vertikal) dalam satu *temporal window*.

2.2.3.5 Deviasi Standar (*Standard Deviation*)

Deviasi standar atau lebih dikenal dengan sebutan simpangan baku, merupakan ukuran dari jumlah variasi atau persebaran pada sebuah kumpulan nilai [41]. Nilai deviasi standar yang rendah mengindikasikan bahwa nilai-nilai pada sekumpulan data cenderung berada dekat dengan rata-rata (*mean*) dari kumpulan data tersebut, sedangkan nilai deviasi standar yang tinggi mengindikasikan bahwa nilai-nilainya berada tersebar dalam jangkauan yang lebih luas.

Fitur standar deviasi dari tiap data dapat dihitung sebagai berikut

$$stddev_i = \sqrt{\left(\sum_{n=a}^b (x_{n+1} - x_n)\right)^2 + \left(\sum_{n=a}^b (y_{n+1} - y_n)\right)^2} \quad (2-6)$$

di mana:

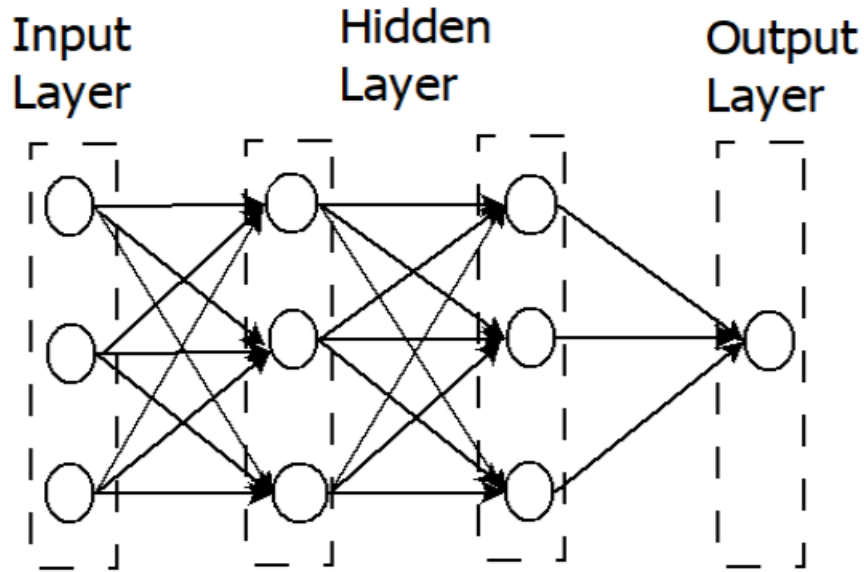
$stddev_i$ = Deviasi standar dari keseluruhan data dalam satu *temporal window*,

x_n dan y_n = Posisi data di indeks n pada sumbu x (horizontal) dan y (vertikal) dalam satu *temporal window*.

2.2.4 Pembelajaran Dalam (*Deep Learning*)

Deep learning merupakan salah satu bidang dari bagian *machine learning* yang mempelajari teknik penentuan satu atau lebih variabel keluaran seperti klasifikasi dan prediksi berdasarkan variabel masukan berupa fitur-fitur. Terdapat berbagai macam algoritme *machine learning* yang dapat melakukan pembelajaran data sederhana untuk memprediksi variabel tertentu dengan berprinsip mengurangi galat/error dalam melakukan prediksi. Namun, pada *deep learning*, sistem akan melatih data yang sangat besar dan kompleks ke dalam beberapa lapisan yang mana hasil dari setiap lapisan akan berpengaruh terhadap proses pada lapisan berikutnya. Untuk meningkatkan tingkat keakuratan, maka pada setiap prosesnya diberikan bobot tertentu untuk memberikan detail pada setiap lapisan [5].

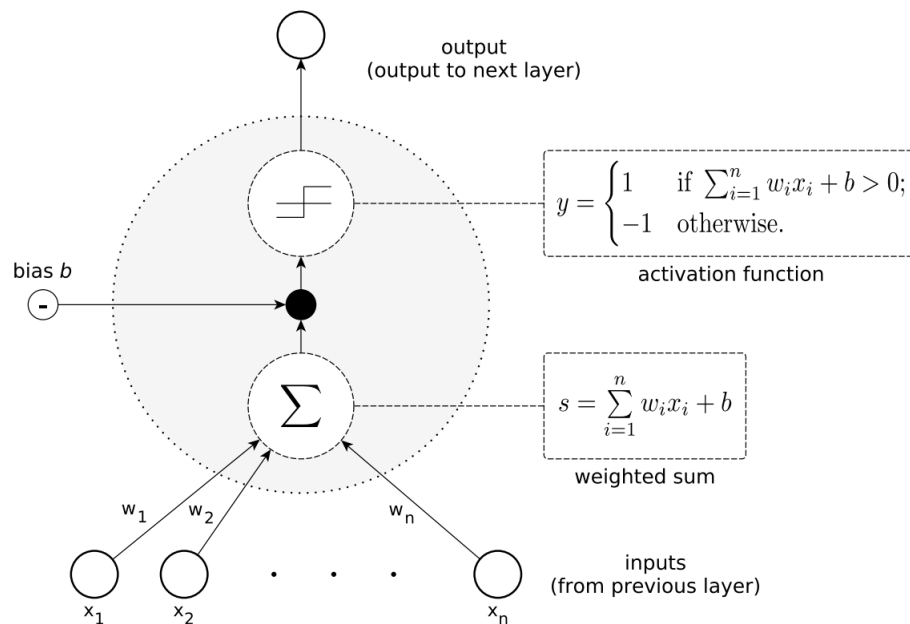
Elemen yang mendasari sebagian besar teknik *deep learning* adalah *Artificial Neural Network* (ANN). ANN secara sederhana terdiri dari 3 *layer* seperti pada Gambar 2.6, yaitu *input layer*, *hidden layer*, dan *output layer*. *Input* dan *output layer* merupakan nilai variabel dari masukan dan keluaran yang diharapkan dari sistem. *Hidden layer* merupakan proses yang tidak terlihat yang berjalan di sistem saraf [5].



Gambar 2.6. Layer pada *deep learning* [5].

2.2.4.1 Artificial Neural Network

Artificial Neural Network (ANN) merupakan cabang dari model *machine learning* yang terinspirasi dari model sistem pembelajaran biologis yang dibangun berdasarkan koneksi dari berbagai neuron yang saling berhubungan dalam sebuah otak [42]. Pemodelan matematis dari sebuah neuron disebut sebagai *perceptron*. Terdapat empat bagian yang menyusun sebuah *perceptron*, yakni lapisan masukan/*input*, bobot (*weight*) atau bias, penjumlah, serta lapisan luaran yang umumnya terhubung dengan sebuah *activation function*.



Gambar 2.7. Struktur dari satu unit *perceptron* dasar. [6].

Gambar 2.7 menunjukkan struktur dari *perceptron* yang dasar. Terlihat bahwa masukan x_1 hingga x_n akan masing-masing diberi bobot yang berbeda-beda dan dijumlahkan. Teknik ini dikenal sebagai *weighted sum*. Persamaan perhitungan sederhana dari sebuah *perceptron* adalah sebagai berikut

$$\hat{y} = g(w_0 + \sum_{i=1}^m x_i w_i) \quad (2-7)$$

di mana:

\hat{y} = keluaran dari *perceptron*,

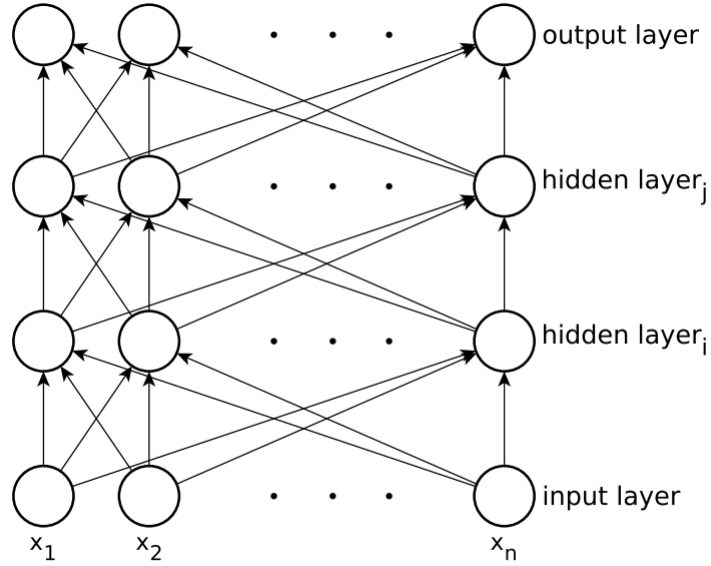
g = *activation function* yang digunakan. Umumnya berupa fungsi non linier,

x_i = *node* dari masukan pada indeks i ,

w_i = bobot (*weight*) dari sebuah *node*.

2.2.4.2 Jaringan Saraf Umpan Maju dan Jaringan Saraf Maju Umpan Mundur (*Forward Propagation and Backpropagation*)

Di dalam jaringan ANN, sekumpulan neuron disusun ke dalam bentuk lapisan-lapisan (*layer*), di mana tiap neuron akan menghitung jumlahan bobot (*weighted sum*) dari tiap masukannya. Dalam penerapannya, terdapat beberapa jaringan saraf penyusun ANN, salah satunya adalah jaringan saraf umpan maju, atau lebih dikenal sebagai *forward propagation*. Contoh sederhana dari lapisan ini yaitu jaringan *perceptron* satu lapis, yang berisikan sekumpulan neuron masukan (*input layer*), serta sekumpulan neuron keluaran (*output layer*) [6]. Pada jaringan ini, masukan akan secara langsung terkoneksi dengan lapisan luaran, dengan sekumpulan bobot yang dipasang pada tiap koneksi yang dibentuk. Beberapa jaringan *perceptron* yang digabung dapat membentuk sebuah jaringan umpan depan berlapis (*forward multilayer network*). Pada jaringan ini, masukan dan luaran akan terhubung melewati sebuah lapisan khusus yang disebut sebagai lapisan tersembunyi (*hidden layer*). *Hidden layer* merupakan lapisan penghubung antara lapisan masukan dan luaran dari sebuah jaringan, dan umumnya memproses data dengan menerapkan fungsi non linier pada masukan sebelum dimasukkan ke dalam lapisan luaran. Gambar 2.8 menunjukkan contoh jaringan *feed-forward multilayer* sederhana dengan tiga jenis lapisan.



Gambar 2.8. Contoh jaringan *feed-forward multilayer* dengan satu lapisan masukan, dua lapisan tersembunyi, serta satu lapisan luaran. [6].

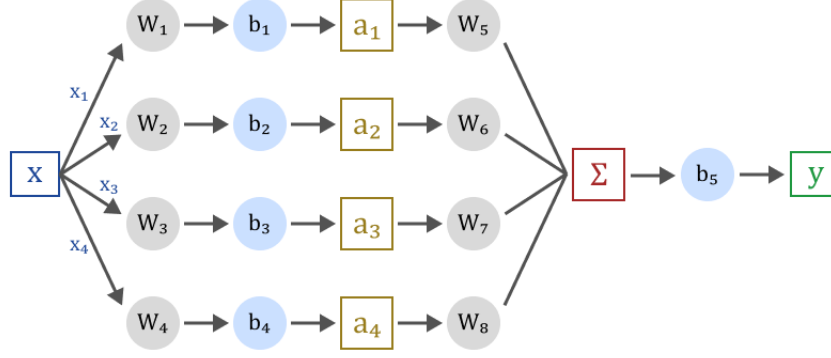
Dengan menggunakan *activation function* pada lapisan luaran, jaringan *multilayer* dapat menghasilkan luaran yang bersifat non-linier. Fungsi non-linier digunakan agar dapat menangkap pola dari data dengan lebih baik dibandingkan dengan fungsi linier. Untuk mengevaluasi hasil dari *output layer*, salah satu teknik yang umum digunakan adalah algoritme umpan balik (*backpropagation*). *Backpropagation* merupakan algoritme umpan balik yang menggunakan nilai yang dihasilkan oleh *output layer* untuk dibandingkan dengan nilai sebenarnya (*true value*) dengan tujuan untuk menyesuaikan bobot serta bias pada *hidden layer* agar dapat memaksimalkan performa dari jaringan [43]. Performa dari jaringan ini diukur dengan sebuah fungsi yang disebut sebagai *loss function* \mathcal{L} [44]. Untuk dapat mengatur bobot yang terdapat pada jaringan, digunakan juga sebuah algoritme yang disebut sebagai *gradient descent*. Algoritme ini akan bekerja mulai dari *output layer* mengarah ke *input layer*. Agar dapat melakukan *gradient descent*, *activation function* yang digunakan harus dapat diturunkan (*differentiable*). *Gradient descent* akan mencari *local minima* dari *activation function*, untuk meminimalisir *loss function* [45].

Dalam penerapannya, jaringan akan menentukan bobot pada masing-masing *node* secara acak terlebih dahulu. Gambar 2.9 menunjukkan contoh jaringan ANN sederhana pada proses *backpropagation*. Bobot dari suatu lapisan didefinisikan sebagai $w^{[l]}$, di mana l adalah indeks dari lapisan (untuk jaringan *multilayer*), dengan $l \in L$ jumlah lapisan yang ada dalam jaringan. Sebuah vektor bobot $w^{[l]}$ pada suatu lapisan dapat diidentifikasi sebagai berikut

$$w^{[l]} = (w_1^{[l]}, w_2^{[l]}, w_3^{[l]}, \dots, w_n^{[l]})^T \quad (2-8)$$

dengan n yaitu jumlah *node* pada lapisan tersebut. Masukan x juga merupakan vektor yang dapat diidentifikasi sebagai berikut.

$$x^{[l]} = (x_1^{[l]}, x_2^{[l]}, x_3^{[l]}, \dots, x_n^{[l]})^T \quad (2-9)$$



Gambar 2.9. Contoh jaringan ANN sederhana pada proses *backpropagation*.

Dari Gambar 2.9 juga terlihat bahwa masing-masing masukan $x^{[l]}$ akan dikalikan dengan bobot $w^{[l]}$ dan dijumlahkan dengan bias $b^{[l]}$, menghasilkan nilai yang disebut sebagai *weighted input* $z^{[l]}$.

$$z^{[l]} = w^{[l]} \cdot x^{[l]} + b^{[l]} \quad (2-10)$$

Masukan berbobot $z^{[l]}$ akan dipetakan dengan menggunakan *activation function* $g^{[l]}$, menyebabkan persamaan menjadi

$$a^{[l]} = g^{[l]}(z^{[l]}) \quad (2-11)$$

di mana $a^{[l]}$ merupakan hasil luaran pemetaan dari *activation function* $g^{[l]}$ terhadap $z^{[l]}$, sehingga merupakan fungsi terhadap $z^{[l]}$. Lanjut, $a^{[l]}$ akan kemudian lanjut dikalikan dengan bobot pada lapisan kedua yaitu $w^{[l+1]}$, untuk kemudian dijumlahkan dengan menambahkan bias kedua $b^{[l+1]}$, sehingga menjadi:

$$y^{[l]} = \sum_{i=1}^n (a_i^{[l]} \cdot w_i^{[l+1]}) + b^{[l]} \quad (2-12)$$

di mana $y^{[l]}$ merupakan luaran akhir dari jaringan. Luaran akhir ini akan kemudian dibandingkan dengan *true value* dengan menggunakan fungsi *loss function* \mathcal{L} yang telah ditentukan. Jika kita misalkan *loss function* \mathcal{L} sebagai fungsi terhadap luaran $y^{[l]}$:

$$\mathcal{L}(y_1^{[l]}, y_2^{[l]}, y_3^{[l]}, \dots, y_n^{[l]}) \quad (2-13)$$

maka selanjutnya untuk mencari nilai masing-masing bobot yang optimal, algoritme *gra-*

gradient descent akan diterapkan dengan melakukan turunan dari *loss function* terhadap masing-masing bobot itu sendiri. Algoritme *backpropagation* akan mulai dari lapisan terbelakang L dan secara iteratif mundur satu lapisan setelah nilai bobot maupun bias pada lapisan sebelumnya sudah didapat. Gradien dari masing-masing bobot $w^{[l]}$ dapat dihitung dengan menggunakan persamaan sebagai berikut.

$$\frac{\partial \mathcal{L}}{\partial w_1^{[l]}}, \frac{\partial \mathcal{L}}{\partial w_2^{[l]}}, \dots, \frac{\partial \mathcal{L}}{\partial w_n^{[l]}} \quad (2-14)$$

Dengan menggunakan properti yang sudah didefinisikan sebelumnya, kita dapat memanfaatkan teknik *chain rule* [46] untuk memecah turunan parsial tersebut menjadi beberapa bagian. Sebagai contoh, untuk mencari bobot pertama $w_1^{[l]}$, kita dapat memecah turunan parsial $\frac{\partial \mathcal{L}}{\partial w_1^{[l]}}$ menjadi:

$$\frac{\partial \mathcal{L}}{\partial w_1^{[l]}} = \frac{\partial \mathcal{L}}{\partial y_1^{[l]}} \cdot \frac{\partial y_1^{[l]}}{\partial a_1^{[l]}} \cdot \frac{\partial a_1^{[l]}}{\partial z_1^{[l]}} \cdot \frac{\partial z_1^{[l]}}{\partial w_1^{[l]}} \quad (2-15)$$

Sehingga, keseluruhan bobot $w^{[l]}$ dapat dihitung dengan menggunakan persamaan umum sebagai berikut.

$$\frac{\partial \mathcal{L}}{\partial w^{[l]}} = \sum_{i=1}^n \left(\frac{\partial \mathcal{L}}{\partial y_i^{[l]}} \cdot \frac{\partial y_i^{[l]}}{\partial a_i^{[l]}} \cdot \frac{\partial a_i^{[l]}}{\partial z_i^{[l]}} \cdot \frac{\partial z_i^{[l]}}{\partial w_i^{[l]}} \right) \quad (2-16)$$

Setelah gradien dari masing-masing bobot didapatkan, proses iterasi akan berlanjut hingga didapatkan nilai gradien yang paling optimal tergantung pada *loss function* yang digunakan. Pada iterasi selanjutnya, nilai bobot akan diperbarui dengan menggunakan nilai gradien yang didapat dikalikan dengan sebuah parameter yang disebut sebagai *learning rate*. *Learning rate* akan mengatur seberapa cepat sebuah gradien akan berubah dari tiap iterasi. Sehingga nilai bobot pada iterasi selanjutnya adalah sebagai berikut

$$w_2^{[l]} = w_n^{[l]} - step_n^{[l]} \quad (2-17)$$

dengan

$$step_n^{[l]} = \eta \cdot \frac{\partial \mathcal{L}}{\partial w_n^{[l]}} \quad (2-18)$$

di mana η adalah nilai *learning rate* yang digunakan. *Gradient descent* akan terus memperbarui nilai bobot maupun bias hingga ukuran $step \ step_n^{[l]}$ sudah melewati batas optimal suatu *loss function*.

2.2.4.3 Loss Function

Loss function sangat berpengaruh pada bagaimana nilai luaran hasil prediksi dibandingkan dengan *true value*. Pemilihan *loss function* yang salah dapat mengakibatkan

model menjadi tidak akurat dalam mempelajari pola yang dapat meminimalisir perbedaan antara luaran model dengan *true value*. Dalam beberapa tahun terakhir, beberapa riset mengenai *loss function* dilakukan untuk merancang berbagai jenis *loss function* yang dapat digunakan pada model *deep learning* [47] [48] [49]. Salah satu *loss function* yang umum digunakan pada kasus regresi adalah *Mean Squared Error* (MSE), yang didapat dari menghitung rerata dari kuadrat selisih antara masing-masing nilai sebenarnya (*true value*) terhadap hasil prediksi [50] [51]. MSE dari sebuah bobot dapat dihitung sebagai berikut

$$\mathcal{L}(W) = \frac{1}{n} \sum_{i=1}^n (y^i - \hat{y}^i(W))^2 \quad (2-19)$$

di mana $\mathcal{L}(W)$ adalah *loss* dari bobot W , n yaitu jumlah dari bobot yang dihitung, y^i yaitu nilai sebenarnya pada indeks i , dan $\hat{y}^i(W)$ nilai prediksi pada indeks i yang juga merupakan fungsi terhadap W .

Pada kasus klasifikasi, beberapa *loss function* yang lain digunakan. Salah satunya adalah *Cross Entropy* (CE) [52]. Terdapat beberapa kelas dari *Cross Entropy*, seperti *Binary Cross Entropy*. *Binary Cross Entropy* merupakan fungsi *loss* yang khusus digunakan pada jenis klasifikasi dengan dua target (*binary*) [53]. Fungsi *loss* ini dapat dihitung sebagai berikut.

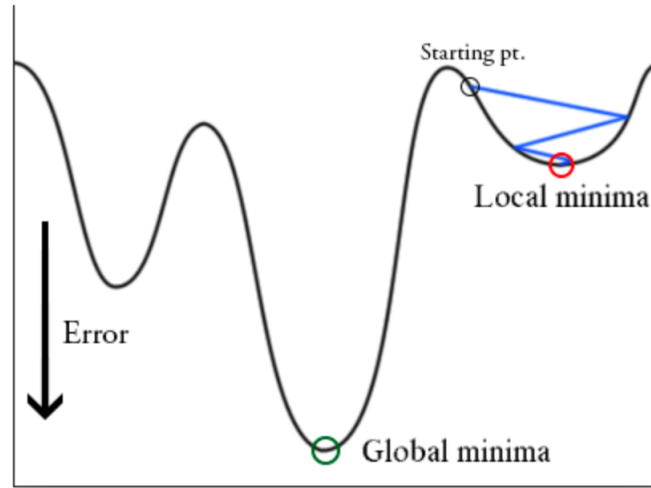
$$\mathcal{L}(W) = \frac{1}{n} \sum_{i=1}^n y^i \log(\hat{y}^i(W)) + (1 - y^i) \log(1 - \hat{y}^i(W)) \quad (2-20)$$

Kelas lain dari *Cross Entropy* yang digunakan pada kasus klasifikasi dengan jumlah target lebih dari dua (*multilabel*) adalah *Categorical Cross Entropy*. Fungsi ini dapat dihitung sebagai berikut.

$$\mathcal{L}(W) = - \sum_{i=1}^n y^i \log(\hat{y}^i(W)) \quad (2-21)$$

2.2.4.4 Optimizer Algorithm

Algoritme *gradient descent* bekerja dengan cara mencari gradien dari suatu parameter pada suatu iterasi dengan tujuan untuk mencapai *local minima* dari sebuah *loss function*. *Local minima* dapat dikatakan sebagai titik terendah dari sebuah bagian pada *loss function*. Nyatanya, nilai optimal yang ingin dicari dari sebuah *loss function* terdapat pada *global minima*, yakni titik terendah dari keseluruhan grafik *loss function*. Permasalahan muncul pada *gradient descent*. Dikarenakan pemilihan nilai mula-mula umumnya dilakukan secara acak, maka akan ada potensi di mana algoritme *gradient descent* memberikan nilai akhir pada *local minima*, bukan *global minima*.



Gambar 2.10. Ilustrasi permasalahan *gradient descent* yang berada di *local minima*.

Gambar 2.10 menunjukkan ilustrasi dari permasalahan yang muncul pada *gradient descent*, saat *loss function* memiliki lebih dari satu lembah. Dikarenakan iterasi pada *gradient descent* akan secara otomatis berhenti saat mencapai titik terendah dari suatu lembah tempat mula titik tersebut tanpa memperhatikan lembah yang lain, maka nilai akhir yang diberikan akan menjadi tidak akurat. Salah satu solusi yang umum digunakan adalah dengan menggunakan algoritme *Stochastic Gradient Descent* (SGD) [54]. *Stochastic Gradient Descent* dalam penerapannya memperkenalkan konsep *stochastic*, yang bermakna 'acak'.

Dibandingkan dengan menghitung gradien pada keseluruhan dataset pada setiap iterasi, SGD menghitung gradien hanya berdasarkan subset dari data yang dipilih secara acak, yang disebut sebagai *mini-batch*. Selain itu, algoritme ini memperkenalkan derau (*noise*) dari hasil perhitungan gradien, dengan tujuan agar algoritme dapat mengeksplorasi berbagai macam daerah yang lain dan keluar dari *local minima*. Adapun persamaan dari algoritme SGD adalah sebagai berikut

$$\theta_i = \theta_i - \eta \frac{\partial \mathcal{L}(\theta)}{\partial \theta_i} \quad (2-22)$$

di mana θ_i adalah parameter yang dicari, baik itu bobot maupun bias.

Selain SGD, terdapat juga algoritme *optimization* lain yang umum digunakan, yaitu Adam (*Adaptive Moment Estimation*) [7]. Adam merupakan algoritme yang memanfaatkan properti dari SGD. Jika dalam implementasinya SGD memperbarui parameter dari model berdasarkan gradien yang dihitung dari kumpulan data yang dipilih secara acak, Adam menggunakan *learning rate* yang bersifat adaptif untuk mempercepat konvergensi ke nilai yang optimal. Sehingga, berbeda dengan SGD yang menggunakan nilai *learning rate* yang tetap, Adam menggunakan nilai *learning rate* yang dapat berubah-

ubah. Adapun *pseudocode* dari algoritme Adam dapat dilihat sebagai berikut.

Algoritme 4 Algoritme *Adam Optimizer*

Masukan: α = ukuran *step* (*step size*)

Masukan: $\beta_1, \beta_2 \in [0, 1)$ = Derajat peluruhan eksponensial (*exponential decay rate*)

Masukan: $\mathcal{L}(\theta)$ = Fungsi objektif *stochastic* dengan parameter θ

Masukan: θ_0 = Vektor parameter mula-mula

Luaran: θ_t = Vektor parameter hasil

Algoritme:

 Inisialisasi momen vektor pertama $m_0 = 0$

 Inisialisasi momen vektor kedua $v_0 = 0$

 Inisialisasi *timestep* $t = 0$

while θ_t belum konvergen **do**

$t \leftarrow t + 1$

 Hitung gradien pada waktu t $g_t = \nabla_{\theta} \mathcal{L}_t(\theta_{t-1})$

 Perbarui estimasi momen pertama $m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$

 Perbarui estimasi momen kedua $v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$

 Perbarui estimasi momen pertama dengan perbaikan pada bias $\widehat{m}_t = m_t / (1 - \beta_1^t)$

 Perbarui estimasi momen kedua dengan perbaikan pada bias $\widehat{v}_t = v_t / (1 - \beta_2^t)$

 Perbarui parameter $\theta_t = \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$

end while

Return: θ_t parameter hasil.

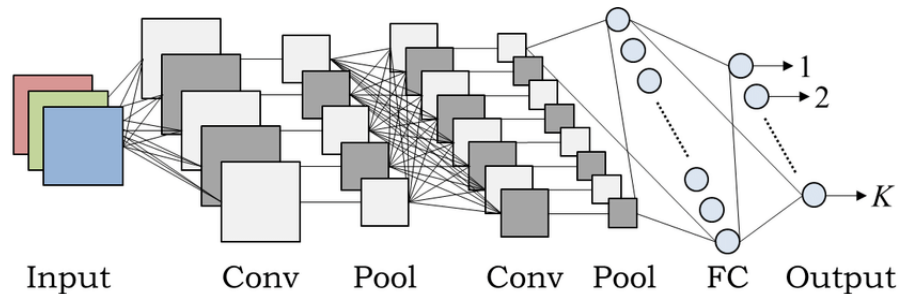
Gambar 2.11. *Pseudocode* dari algoritme *optimizer* Adam. [7]

2.2.4.5 1 Dimensional-Convolutional Neural Network

Fukushima [55] memperkenalkan arsitektur *Convolutional Neural Network* (CNN), dan merupakan arsitektur paling umum dalam pemrosesan gambar dan visi komputer, di mana ukuran matriks yang diolah sangat besar. Arsitektur CNN secara umum terdiri dari tiga bagian: *convolutional*, *pooling*, dan *fully-connected layer*. *Convolution layer* dan *pooling layer* merupakan tahap *preprocessing* yang bertujuan untuk mengurangi jumlah fitur-fitur pada data masukan untuk kemudian diproses pada *fully-connected layer* [5].

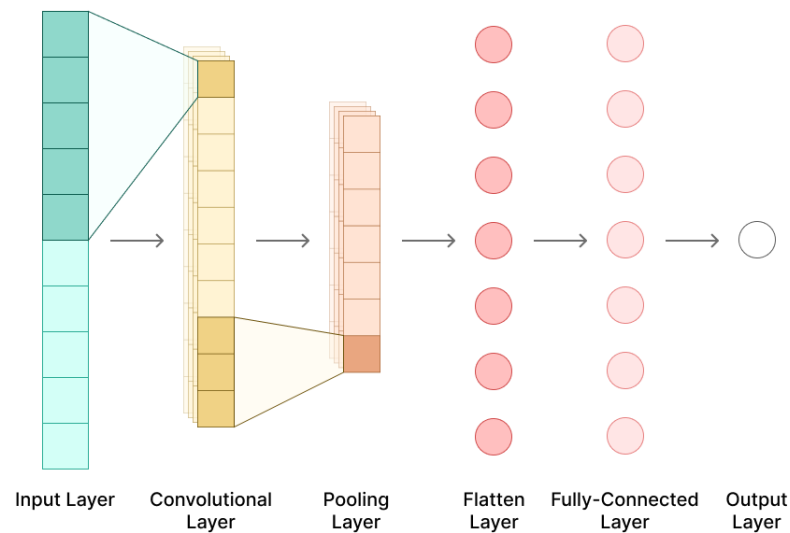
Convolutional layer memiliki tugas utama yaitu sebagai lapisan yang memeriksa karakteristik dari data masukan. Sesuai namanya, pada lapisan ini, operasi konvolusi akan dilakukan dengan menggunakan sebuah *filter* dengan ukuran tertentu. Lanjut, pada lapisan *pooling*, hasil konvolusi akan dihimpun untuk kemudian digunakan pada pembagian bobot di antara beberapa koneksi. Proses ini bertujuan untuk mengurangi fitur-fitur yang diperlukan untuk prediksi. Hasil dari *pooling layer* adalah fitur-fitur dengan ukur-

an yang lebih sedikit namun merupakan *summary* dari masukan matriks hasil konvolusi sebelumnya yang berukuran lebih besar. Terakhir, proses prediksi akan dilakukan pada *fully-connected layer*. CNN sudah terbukti sangat berguna dalam melakukan pengolahan citra karena kemampuannya dalam mengenali level fitur yang tinggi yang merupakan komposisi dari fitur-fitur tingkat rendah [56].



Gambar 2.12. Struktur arsitektur CNN. [8]

Meskipun CNN umumnya digunakan pada pengolahan citra, di mana data berukuran 2D, terdapat juga varian dari CNN yang dapat mengolah data 1D seperti data *time series* [57]. Berbeda dengan 2D CNN yang menggunakan matriks 2D sebagai kernel *filter* pada *convolution layer*, 1D CNN menggunakan *array* satu dimensi yang memiliki lebar tetap dan bergerak sepanjang sumbu x. Oleh karena itu, kompleksitas dari komputasi yang dilakukan pada 1D CNN lebih rendah dibandingkan dengan 2D CNN.



Gambar 2.13. Struktur arsitektur 1D CNN.

2.2.4.6 Temporal Convolutional Network (TCN)

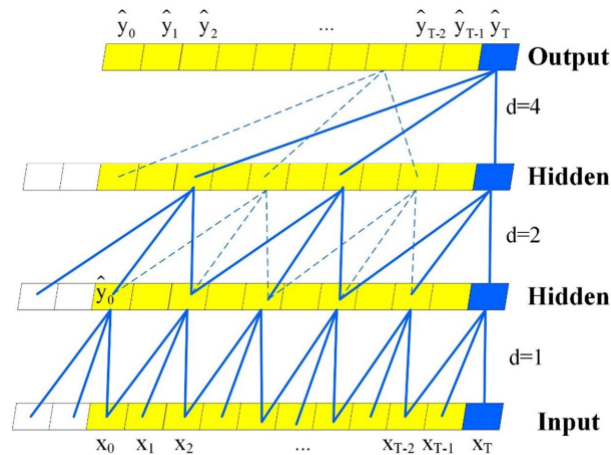
Pada bidang analisis data *time series*, salah satu metode yang umumnya digunakan adalah *Reccurent Neural Network* (RNN) [58]. Pada RNN, koneksi antar simpul (*nodes*) dapat membentuk sebuah siklus, di mana luaran dari sebuah *node* dapat mempengaruhi input dari *node* yang sama. Hal ini mengakibatkan RNN bersifat temporal. Perkembangan dari arsitektur RNN menghasilkan arsitektur *Long Short-Term Memory* (LSTM), yang dapat memproses ribuan hingga jutaan titik waktu, dengan kemampuan pemrosesan yang baik [59].

Namun, riset terbaru menunjukkan bahwa TCN, salah satu bagian dari arsitektur 1D CNN, dapat menghasilkan performa yang lebih baik dari LSTM, terutama pada pemrosesan urutan data masukan yang panjang [60]. TCN memiliki karakteristik umum, seperti: 1) Strukturnya yang bersifat konvolusional, layaknya 1D CNN. Konvolusi yang dilakukan pada dasarnya adalah konvolusi kausal (*causal convolution*), yang berarti luaran pada tiap indeks waktu hanya bergantung pada masukan saat ini dan masukan sebelumnya. 2) TCN dapat memproses masukan dengan ukuran panjang berapapun dan dapat menghasilkan luaran dengan panjang yang sama dengan panjang dari masukan, seperti halnya pada RNN. Selain itu, dikarenakan TCN memiliki lapisan konvolusional, proses ini dapat diparalelkan, sehingga dapat menghasilkan waktu *training* yang lebih cepat dibandingkan dengan RNN maupun LSTM [36].

Pada implementasinya, TCN akan menggunakan arsitektur *one-dimensional fully connected network* (1D FCN). Dikarenakan luaran yang dihasilkan memiliki panjang yang sama dengan masukan, tiap bagian pada *hidden layer* akan diisi dengan nilai 0 hingga mencapai panjang yang sama dengan masukan. Lanjut pada lapisan konvolusi, proses konvolusi akan dilakukan sebagaimana umumnya dilakukan pada 2D CNN, dengan perbedaan ukuran kernel dan masukan data. FCN pada TCN memiliki perbedaan dengan *fully connected layer* pada 2D CNN. Pada *fully connected layer*, luaran dari masukan akan menghasilkan vektor dengan panjang yang tetap, di mana vektor ini sudah merepresentasikan luaran-luaran yang diatur pada model. Namun pada 1D FCN di TCN, proses dekonvolusi akan dilakukan setelah adanya proses konvolusi. Proses dekonvolusi inilah yang menyebabkan luaran dari arsitektur dapat memiliki panjang yang sama dengan masukan data.

Permasalahan muncul saat data yang digunakan adalah *time series* dengan panjang yang cukup besar. Penerapan *causal convolution* secara langsung terhadap data masukan menjadi masalah dikarenakan data yang dapat dilihat sangat terbatas. Oleh karena itu, diterapkan sebuah teknik konvolusi yang disebut sebagai konvolusi dilasi (*dilated convolution*) [61]. *Dilated convolution* memiliki ukuran kernel yang sama dengan *causal convolution*, namun jarak dan *step* yang diambil oleh *dilated convolution* ditentukan oleh faktor dilasi yang ditetapkan. Gambar 2.14 menunjukkan cara kerja dari *dilated convolu-*

tion saat sistem kausal. Jika faktor dilasi bernilai 1, maka konvolusi akan bekerja seperti layaknya *causal convolution* biasa. Namun, saat faktor dilasi bernilai 2, 4 dan seterusnya, akan ada jarak antar satu titik ke titik lainnya dalam proses konvolusi masukan dengan kernel. Walaupun ukuran kernel tetap sama, namun indeks yang diambil berubah.



Gambar 2.14. Arsitektur *dilated convolution* kausal dengan faktor dilasi 1, 2, dan 4 dan ukuran kernel 3. [9]

2.2.5 Hyperparameter pada Deep Learning

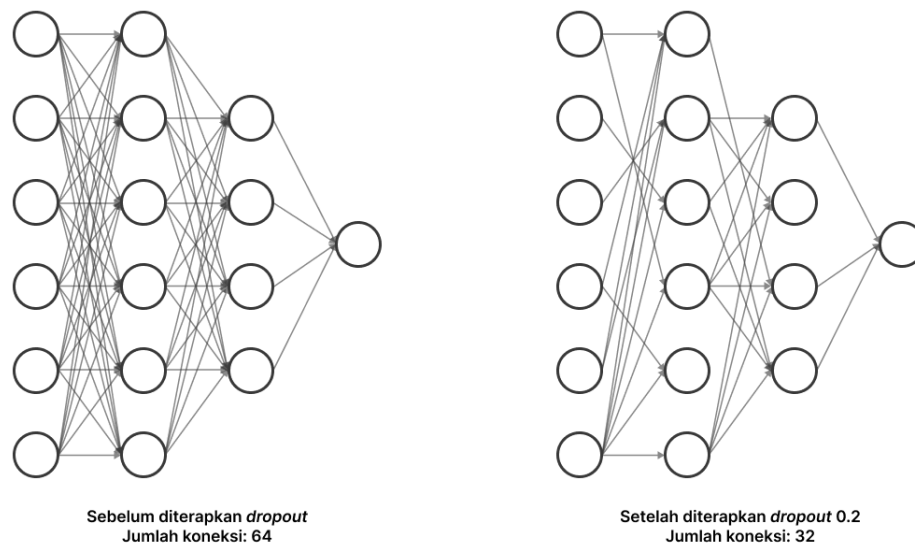
Hyperparameter merupakan parameter pada model *machine learning* yang ditentukan sebelum model dilatih, dengan nilai yang tetap (tidak dapat berubah saat proses *training* berlangsung). Selain model *machine learning*, model *deep learning* juga memiliki *hyperparameter* yang dapat ditentukan. Pada *deep learning*, *hyperparameter* umumnya mengacu pada arsitektur dari model itu sendiri, terutama pada arsitektur dengan jenis konvolusional.

2.2.5.1 Dropout Rate

Dropout rate atau *dropout regularization* merupakan teknik pada *deep learning* maupun model *machine learning* yang lain yang digunakan untuk mencegah terjadinya *overfitting* pada proses *training* [62]. *Overfitting* merupakan kondisi di mana model dapat bekerja dengan baik pada data latih (*training*), namun tidak pada data uji (*test*) [63]. Ini mengakibatkan model terlalu dalam mempelajari pola pada *training* tanpa ada proses generalisasi untuk data yang belum terlihat (seperti pada data *testing*).

Pada model yang diberi *dropout rate*, terdapat beberapa neuron yang secara acak dipilih untuk kemudian dikosongkan bobotnya pada tiap iterasi. Sehingga tidak ada informasi yang masuk dari neuron tersebut ke *layer* selanjutnya. Proses ini terjadi secara berulang-ulang sehingga model akan 'dipaksa' untuk mempelajari data secara umum, dan akan lebih mudah untuk memahami data yang belum pernah dilihat. Gambar 2.15

menunjukkan contoh arsitektur *neural network* sebelum dan sesudah diberi *dropout* pada tiap lapisannya.



Gambar 2.15. Contoh ilustrasi penggunaan *dropout* pada *neural network* sebesar 50%.

2.2.5.2 *Filter Number*

Pada *deep learning* khususnya pada arsitektur TCN, *filter number* merupakan *hyperparameter* yang menentukan jumlah *filter* atau kanal yang digunakan pada *layer dilated convolutional*. *Filter number* akan menentukan ukuran dari luaran yang dihasilkan oleh model TCN sebelum dimasukkan ke *layer* terakhir (prediktor). Parameter *filter number* akan mengatur kompleksitas dan kapasitas dari model, di mana semakin tinggi ukuran *filter* akan membuat model yang dapat mempelajari representasi yang lebih kompleks dari masukan. Selain itu, mengurangi jumlah *filter* juga dapat berperan dalam mencegah *overfitting*, dikarenakan kompleksitas dari model berkurang.

2.2.5.3 *Kernel Size*

Kernel size merupakan *hyperparameter* yang menentukan ukuran dari *kernel* yang digunakan pada *convolutional layer*, termasuk pada *dilated convolutional layer* jika menggunakan arsitektur TCN. Parameter ini yang akan menentukan lebar dari *kernel* yang akan bergerak sepanjang masukan dalam proses konvolusi, dengan memperhatikan faktor dilasi yang ditentukan. Nilai *kernel size* yang tinggi akan mengakibatkan model menjadi lebih mudah dalam menangkap konteks dengan cakupan yang lebih luas, terutama pada data *time series*. Selain itu, jumlah *kernel* yang lebih besar akan berdampak pada meningkatnya kompleksitas dari model.

2.2.5.4 *Dilation Rate*

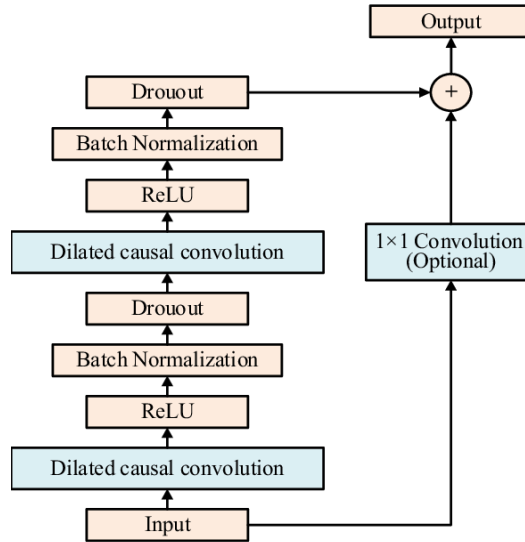
Dilation rate merupakan *hyperparameter* yang terdapat pada arsitektur yang menggunakan *dilated convolution layer* seperti pada TCN. Parameter ini menentukan faktor-faktor dilasi yang akan digunakan pada tiap *hidden layer* yang terdapat pada TCN. La-
yaknya *kernel size*, parameter ini juga menentukan kompleksitas model di mana faktor dilasi yang semakin besar akan berakibat pada cakupan konteks yang lebih luas pada proses konvolusi.

2.2.5.5 *Stacks Number*

Stacks number merupakan *hyperparameter* yang mengatur jumlah *stacking* pada blok residu (*residual blocks*). Pada arsitektur TCN, konsep *residual block* diperkenalkan untuk menanggulangi *vanishing gradient problem* [9], yang umumnya terjadi pada mode *deep learning*. *Vanishing gradient problem* merupakan kejadian di mana gradien mulai 'hilang' secara eksponen seiring bertambah panjangnya *layer* pada model. Fenomena ini mengakibatkan model menjadi sulit untuk mempelajari pola pada proses *training*. Untuk menanggulangnya, TCN memperkenalkan konsep *residual learning*.

Residual learning merupakan konsep di mana sebuah jaringan akan fokus mempelajari residu atau selisih dari masukan dan luaran yang diinginkan. Pada TCN, *residual learning* ini diterapkan pada sebuah *residual block*, di mana pada *residual block* terdapat sebuah *convolutional layer* (dengan tambahan lain seperti *activation function* maupun *dropout layer*) dan sebuah *skip connection layer*. Masukan pada *residual block* akan dilakukan proses konvolusi seperti biasa, dan ditandai sebagai luaran dari *building block* ini. Di waktu yang sama, masukan juga akan dihubungkan dengan luaran yang sebelumnya sudah diproses oleh *convolutional layer* untuk kemudian dibandingkan. Adanya *skip connection* ini mengakibatkan model jadi dapat mempelajari selisih antara masukan dari *residual block* dengan masukan mula-mula, untuk kemudian dipelajari dan dimaksimalkan. Konsep *residual learning* ini akan membuat model menjadi lebih mudah konvergen lebih cepat ke titik yang diinginkan pada proses *training*.

Parameter *Stacks number* akan menentukan jumlah dari *residual block* yang terdapat pada model. Semakin banyak *residual block* yang ada, maka model akan memiliki kapasitas yang semakin tinggi, di mana model dapat mempelajari dependensi yang lebih dalam. Namun, jumlah *residual block* yang tinggi juga dapat mengakibatkan *overfitting*, dan sebaliknya jumlah *residual block* yang terlalu rendah dapat mengakibatkan *underfitting*.



Gambar 2.16. Struktur *residual block* pada TCN [10]

2.2.5.6 Activation Function

Activation function merupakan fungsi matematis yang dipetakan pada luaran dari sebuah neuron atau *layer* [64]. Tujuan dari *activation function* yaitu untuk membuat luaran model menjadi non linier. Adanya non linier pada luaran membuat model dapat mempelajari pola yang lebih kompleks serta membatasi luaran dari sebuah neuron ataupun *layer*. Beberapa *activation function* juga dapat bertindak sebagai regularisasi pada *layer* untuk mencegah terjadinya *overfitting*. Terdapat beberapa *activation function* yang umumnya digunakan pada model *deep learning*, seperti *sigmoid*, ReLU, Tanh, dan lain-lain [65]. Penggunaan *activation function* umumnya akan berubah-ubah tergantung dari objektif dari model itu sendiri. Sebagai contoh, fungsi *sigmoid* akan memetakan luaran dengan rentang nilai antara 0 hingga 1. Sehingga *sigmoid* umumnya digunakan pada kasus *binary classification* (klasifikasi dengan dua jenis label).

Adapun berbagai persamaan *activation function* beserta dengan turunannya adalah sebagai berikut.

1. *Sigmoid Function*: Persamaan pada *sigmoid function* dapat ditulis sebagai berikut

$$g(z) = \frac{1}{1 + e^{-z}} \quad (2-23)$$

$$g'(z) = g(z)(1 - g(z)) \quad (2-24)$$

2. *Hyperbolic Tangent (Tanh) Function*: Persamaan pada *hyperbolic tangent function* dapat ditulis sebagai berikut

$$g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \quad (2-25)$$

$$g'(z) = 1 - g(z)^2 \quad (2-26)$$

3. *Rectified Linear Unit (ReLU) Function*: Persamaan pada *rectified linear unit function* dapat ditulis sebagai berikut

$$g(z) = \max(0, z) \quad (2-27)$$

$$g'(z) = \begin{cases} 1, & z > 0 \\ 0, & \text{selainnya} \end{cases} \quad (2-28)$$

2.2.5.7 Batch Size

Batch size didefinisikan sebagai jumlah dari sampel *training* yang digunakan dalam satu iterasi pada proses *training* sebelum memperbarui parameter internalnya [66]. Pada CNN, parameter ini merupakan salah satu parameter yang sangat berperan penting dalam proses *training* [67]. Secara spesifik, parameter *batch size* menentukan seberapa banyak prediksi yang harus dibentuk dalam satu waktu/iterasi. *Batch size* juga berperan pada waktu yang diperlukan pada model untuk konvergen ke suatu nilai, serta berpengaruh pada apakah model *overfitting* atau tidak. Ukuran *batch size* yang besar akan memerlukan memori yang lebih besar pula dalam proses *training* untuk menyimpan hasil pemrosesan pada saat *training*.

2.2.5.8 Epoch

Epoch dapat diartikan sebagai jumlah iterasi yang dilakukan pada model saat proses *training* pada keseluruhan data. Hal ini berarti satu iterasi menandakan proses *training* secara keseluruhan pada data, hingga proses pembaruan parameter, beban (*weight*), serta bias dari model [68]. Optimisasi pada *epoch* umumnya untuk menyeimbangkan performa model agar tidak terjadi *overfitting* maupun *underfitting* terhadap data latih. Jumlah *epoch* yang tinggi menandakan iterasi yang dilakukan serta pembaruan parameter yang dilakukan akan lebih banyak, dengan potensi adanya *overfitting* dikarenakan model semakin mudah menghafal data latih, namun tidak dapat digeneralisasikan pada data uji. Sebaliknya, jumlah *epoch* yang terlalu rendah memiliki potensi terjadinya *underfitting*, di mana model belum cukup menangkap pola dari data latih.

2.2.6 Hyperparameter Optimization (HPO)

Hyperparameter pada model *machine learning* maupun *deep learning* dapat diubah terlebih dahulu sebelum dilakukan proses *training*. Oleh karena itu, pemilihan nilai yang tepat pada tiap *hyperparameter* diperlukan. Hal ini karena tiap data memiliki karakteristik yang berbeda-beda [69], dan model harus dikonfigurasi terlebih dahulu untuk

memaksimalkan potensinya pada data terkait. Data yang berbeda akan menghasilkan nilai parameter yang berbeda pula. Proses mendapatkan nilai terbaik pada tiap *hyperparameter* di suatu model disebut sebagai proses *tuning* [70].

Proses *tuning* pada *hyperparameter* berbeda-beda pada tiap algoritme *machine learning* maupun *deep learning* yang berbeda. Ini dikarenakan setiap model memiliki *hyperparameter* yang spesifik dan tidak semuanya sama. Terdapat beberapa jenis dari *hyperparameter*, seperti kategorikal, diskrit, maupun kontinu [71]. Terdapat beberapa teknik dalam melakukan *tuning* pada *hyperparameter* ini. Pengujian secara manual merupakan teknik yang saat ini masih digunakan oleh beberapa riset umum. Namun, umumnya teknik ini memerlukan pemahaman yang dalam mengenai *hyperparameter* yang dicari. Teknik ini sangat tidak efektif dikarenakan banyaknya jumlah *hyperparameter* serta kemungkinan kombinasi dari nilai-nilai *hyperparameter* yang ada, terlebih jika terdapat *hyperparameter* dengan nilai kontinu.

Hal ini mengakibatkan banyaknya riset yang berfokus pada pencarian teknik yang dapat mencari nilai-nilai *hyperparameter* ini secara otomatis. Proses pencarian *hyperparameter* secara otomatis ini dikenal dengan sebutan *Hyperparameter Optimization* (HPO) [72]. Terdapat banyak sekali metode HPO yang dapat digunakan untuk mencari nilai *hyperparameter* yang paling optimal pada sebuah model. Beberapa metode yang paling umum meliputi *Grid Search* [73] dan *Random Search* [74]. *Grid Search* merupakan salah satu metode yang paling umum digunakan untuk mencari *hyperparameter* pada model *machine learning* maupun *deep learning*. Metode ini bekerja dengan prinsip *brute force*, yang berarti akan mencoba keseluruhan kombinasi yang ada dari rentang nilai-nilai tiap *hyperparameter* yang ditentukan. Sedangkan *Random Search* bekerja sesuai namanya, di mana pada tiap iterasi, metode ini akan memilih secara acak sekumpulan nilai *hyperparameter* yang telah didefinisikan sebelumnya untuk dicoba. Kedua metode HPO ini dianggap tidak optimal dikarenakan secara komputasional sangat berat dan tidak efektif dalam menyeleksi rentang nilai-nilai yang potensial.

Selain *Grid Search* dan *Random Search*, terdapat metode HPO lain yang sering digunakan untuk mencari *hyperparameter* terbaik dikarenakan lebih optimal serta memerlukan waktu komputasi yang lebih sedikit. Salah satunya adalah *Bayesian Optimization* [26]. Berbeda dengan *Grid Search* maupun *Random Search*, *Bayesian Optimization* menentukan nilai *hyperparameter* yang akan diuji pada iterasi selanjutnya berdasarkan nilai-nilai *hyperparameter* hasil pengujian pada iterasi sebelumnya, sehingga dapat menghindari pengujian yang terlalu intensif, serta menghasilkan pencarian yang lebih konvergen ke nilai-nilai yang paling optimal.

Bayesian Optimization bekerja dengan cara membuat sebuah model 'pengganti' (*surrogate model*), di mana model ini merupakan model statistik yang dibentuk dari model probabilistik yang memperkirakan sebuah *objective function* yang dipilih. *Objective*

function ini merupakan fungsi utama yang digunakan pada *Bayesian Optimization* untuk dioptimalkan. Model pengganti ini nantinya akan diisi oleh sekumpulan nilai *hyperparameter* mula-mula dan akan dievaluasi menurut *objective function* yang dipilih. Proses ini akan dilakukan secara iteratif dengan melakukan proses *fitting*, pembaruan (*update*) pada model, serta pemilihan konfigurasi *hyperparameter* yang lain untuk mengoptimalkan khususnya *objective function* yang dipilih, hingga didapatkan konfigurasi yang paling optimal.

Algoritme 5 Algoritme *Bayesian optimization*

Masukan: $\theta_0 = \text{Hyperparameter mula-mula}$

Masukan: $H(\theta_0) = \text{Fungsi objektif}$

Luaran: θ_{final} dan y_{final}

Algoritme:

```

Tentukan konfigurasi parameter mula-mula  $\theta_0 \in \Theta$ 
Evaluasi nilai mula-mula  $y_0 = H(\theta_0)$ 
Inisialisasi  $\theta_{final} = \theta_0, y_{final} = H(\theta_0), S_0 = \{\theta_0, y_0\}$ 
for  $n = 1$  to  $n_{max}$  do
    Pilih kombinasi hyperparameter baru  $\theta_n \in \Theta$  dengan mengoptimisasi fungsi akuisisi  $U_n$ .  $\theta_n = \text{argmax} U_n(\theta; S_t)$ 
    Evaluasi  $H$  dalam  $\theta_n$  untuk mendapatkan luaran baru  $y_n = H(\theta_n)$ 
    Gabung data  $S_n = S_{n-1} \cup \{\theta_n, y_n\}$ 
    Perbarui model pengganti (surrogate). Jika  $y_n < F_{final}$ ,  $\theta_{final} = \theta_n$  dan  $y_{final} = y_n$ 
end for

```

Gambar 2.17. *Pseudocode* dari algoritme *Bayesian optimization*.

2.2.6.1 *Hyperband*

Baik *Grid Search*, *Random Search* maupun *Bayesian Optimization* termasuk ke dalam *Black Box Optimization*. *Black Box Optimization* dikatakan sebagai "*Black Box*" dikarenakan metode tersebut tidak mengetahui proses yang terjadi di dalam model maupun *objective function* yang sedang dioptimalkan. Meskipun *Bayesian Optimization* melampaui performa *Random Search* serta *Grid Search* di kebanyakan kondisi dengan menghasilkan nilai yang optimal, metode ini masih memiliki beberapa kelemahan. Pada *Bayesian Optimization*, penggunaan *objective function* dalam mengestimasi nilai-nilai *hyperparameter* sangat memakan sumber daya komputasi [75]. Selain itu, efektivitas pada pencarian nilai-nilai pada metode ini bergantung pada *objective function* yang digunakan. Jika *objective function* yang dipilih tidak secara akurat menggambarkan tujuan dari optimalisasi yang dilakukan, metode ini menjadi tidak efektif.

Terdapat metode lain yang mampu menanggulangi permasalahan yang dialami khususnya pada metode *Bayesian Optimization*. Metode ini disebut sebagai *Hyperband* [27]. *Hyperband* termasuk ke dalam jenis metode *Multi-fidelity Algorithm*. Metode ini sangat efisien dari segi waktu dibandingkan metode *Black Box* dalam mencari nilai-nilai optimal pada *hyperparameter*. Pada pengaplikasiannya, algoritme ini menggabungkan dua macam evaluasi, *low-fidelity* dan *high-fidelity*. Pada evaluasi yang pertama, beberapa subset nilai-nilai *hyperparameter* akan dicoba dengan daya komputasi yang lebih rendah dengan generalisasi yang buruk (umumnya pada nilai *epoch* yang rendah). Selanjutnya, subset-subset nilai *hyperparameter* yang menghasilkan evaluasi yang buruk akan langsung dieliminasi pada tahap *high-fidelity*, sehingga pada tahap tersebut hanya menyisakan kombinasi *hyperparameter* yang terbaik untuk kemudian dievaluasi lagi namun dengan menggunakan daya komputasi yang lebih tinggi. Sehingga, hanya konfigurasi *hyperparameter* yang baik yang akan dievaluasi lebih lanjut pada algoritme ini.

Algoritme 6 Algoritme *Hyperband optimization*

Masukan: R = Jumlah maksimum sumber daya yang dapat dialokasikan ke satu konfigurasi *hyperparameter*

Masukan: η = Pengendali proporsi (*proportion controller*)

Luaran: θ_{final} = konfigurasi *hyperparameter*

Algoritme:

```

Inisialisasi  $s_{max} = \lceil \log_n(R) \rceil$ ,  $B = (s_{max} + 1)R$ 
for  $s \in \{s_{max}, s_{max} - 1, \dots, 0\}$  do
     $n = \left\lceil \frac{B}{R} \frac{\eta^s}{(s+1)} \right\rceil$ ,  $r = R\eta^{-s}$ 
     $X = \text{get\_hyperparameter\_configuration}(n)$ 
    for  $i \in 0, \dots, s$  do
         $n_i = \lceil n\eta^{-i} \rceil$ 
         $r_i = r\eta^i$ 
         $F = \{\text{run\_then\_return\_obj\_val}(x, r_i) : x \in X\}$ 
         $X = \text{top\_k}(X, F, \lceil n_i/\eta \rceil)$ 
    end for
end for

```

Gambar 2.18. *Pseudocode* dari algoritme *Hyperband optimization*.

Hal yang sama juga dilakukan pada *Hyperband*. Alih-alih menggunakan *objective function* sebagai sarana untuk mengoptimalisasi nilai-nilai *hyperparameter* yang ada, *Hyperband* akan langsung secara iteratif mengeliminasi konfigurasi *hyperparameter* yang tidak bekerja dengan baik dan mengalokasikan sisa sumber daya pada konfigurasi yang menjanjikan. Prinsip kerja dari *Hyperband* dapat dikatakan seperti metode *Ran-*

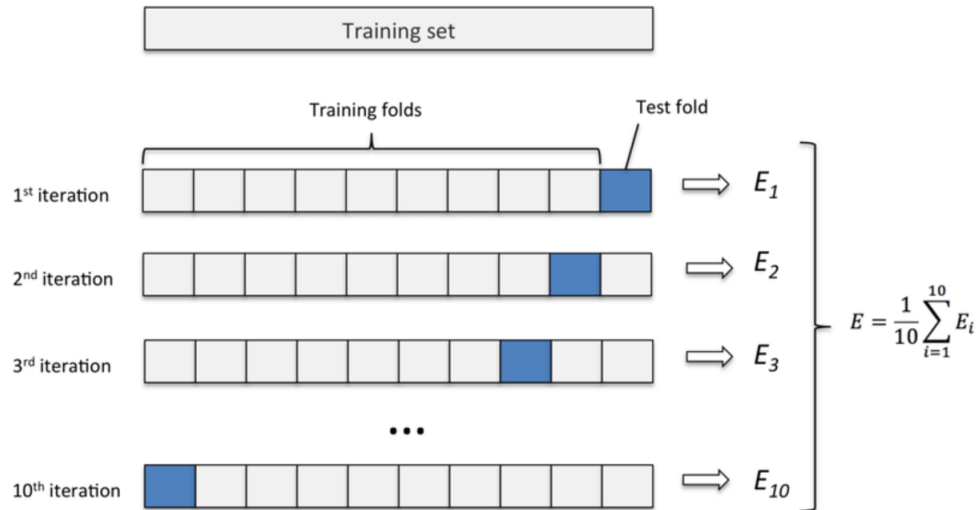
dom Search dengan tambahan *early stopping*. Oleh karena itu, metode *Hyperband* dapat konvergen lebih cepat ke nilai-nilai *hyperparameter* yang paling optimal, terutama saat jangkauan nilai-nilai *hyperparameter* yang ditetapkan lebih luas.

2.2.7 Evaluasi Model

Sebuah model *machine learning* dapat diukur performanya dengan berbagai cara. Pada permasalahan klasifikasi, riset mengenai evaluasi pada model sudah diteliti sejak lama [76]. Setidaknya ada dua jenis dari evaluasi, yakni evaluasi kualitatif dan kuantitatif [77]. Evaluasi kualitatif merupakan jenis evaluasi yang melibatkan secara langsung pengalaman pengguna terhadap sebuah sistem, tanpa ada sesuatu yang diukur. Berbeda dengan evaluasi kualitatif, evaluasi kuantitatif melibatkan suatu ukuran yang dapat diukur. Evaluasi ini dapat direpresentasikan dalam bentuk angka eksak, sebagai metrik pengukuran dari performa model.

2.2.7.1 Metode Evaluasi

Untuk dapat mengukur performa dari model, terdapat berbagai metode yang dapat digunakan. Salah satu metode yang umum digunakan adalah *K-Fold Cross Validation* [78]. Pada metode ini, data dipecah menjadi bagian dengan jumlah sebanyak K . Dari K bagian tersebut, sebanyak $K-1$ bagian digunakan sebagai data latih (*training*), dan 1 bagian disisakan sebagai data uji. Proses ini akan dilakukan berulang kali sebanyak K kali hingga keseluruhan bagian yang terdapat pada K sudah diuji. Umumnya nilai K yang digunakan adalah 10, sehingga disebut sebagai *10-Fold Cross Validation* [79], walau tidak sedikit juga yang menggunakan nilai K sebanyak 5, dikarenakan dapat menghasilkan jumlah data latih yang lebih banyak, terutama pada dataset dengan variabilitas yang tinggi. Gambar 2.19 menunjukkan ilustrasi dari metode *K-Fold Cross Validation* dengan menggunakan nilai K sebesar 10.



Gambar 2.19. Ilustrasi diagram dari metode evaluasi *K-Fold Cross Validation* dengan jumlah sebesar 10. [11]

Selain metode *K-Fold Cross Validation*, terdapat metode evaluasi lain seperti *Leave-One-Out Cross Validation* (LOO CV) [80]. LOO merupakan metode konfigurasi dari *K-Fold Cross Validation*, di mana nilai K diatur sebesar jumlah data yang terdapat pada dataset. Sehingga dapat dikatakan sebagai versi ekstrim dari *K-Fold Cross Validation* dengan daya komputasi yang sangat tinggi. Pada penerapannya, teknik ini hanya cocok digunakan jika ukuran dari dataset sangat kecil, misal 100 [81]. Meskipun demikian, terdapat beberapa variasi lain dari metode LOO yang memanfaatkan cara kerja dari metode tersebut. Startsev *et al* [22], menggunakan metode evaluasi khusus *Leave-One-Video-Out* (LOVO) dengan memanfaatkan properti dari dataset yang digunakan. Dataset gerakan mata yang digunakan oleh Startsev *et al*. direkam pada 18 video yang berbeda-beda. Sehingga proses *training* dilakukan secara iteratif sebanyak 18 kali, di mana pada tiap iterasi model akan dilatih dengan menggunakan 17 video, dan dievaluasi dengan menggunakan 1 video. Berbeda dengan *K-Fold Cross Validation* yang memilih data yang berada dalam 1 *fold* secara acak, metode LOVO secara otomatis menangkap pola karakteristik dari masing-masing jenis video.

2.2.7.2 Metrik Evaluasi

Pada evaluasi kuantitatif, agar dapat mengukur performa dari sebuah model diperlukan sebuah metrik yang dapat merepresentasikan karakteristik dari hasil prediksi terhadap nilai sebenarnya. Pada kasus regresi, beberapa metrik yang umum digunakan seperti *Mean Squared Error* (MSE), *Mean Absolute Error* (MAE), ataupun *Root Mean Squared Error* (RMSE), yang masing-masing dilakukan untuk mencari selisih antara nilai yang diprediksi dengan nilai sebenarnya. Pada ketiga metrik, semakin kecil nilai yang dihasilkan maka menandakan performa model yang semakin bagus.

Sedangkan pada kasus klasifikasi, terdapat beberapa metrik yang dapat digunakan untuk mengevaluasi model. Salah satu bentuk representasi dari beberapa metrik evaluasi terhadap model adalah *Confusion Matrix* [12]. *Confusion matrix* didefinisikan sebagai matriks yang menampilkan campuran dari berbagai metrik performa yang diukur pada hasil prediksi dan *true value*. Gambar 2.20 menampilkan contoh dari *confusion matrix*. *Confusion matrix* akan menampilkan empat nilai, yakni *True Positive* (TP), *False Positive* (FP), *False Negative* (FN), serta *True Negative* (TN).

1. *True Positive* (TP): TP merupakan jumlah dari nilai positif yang terklasifikasi benar oleh model.
2. *False Positive* (FP): FP merupakan jumlah dari nilai positif yang terklasifikasi salah oleh model. Nilai ini juga dikenal sebagai *error* tipe 1.
3. *True Negative* (TN): TN merupakan jumlah dari nilai negatif yang terklasifikasi salah oleh model.
4. *False Negative* (FN): FN merupakan jumlah dari nilai negatif yang terklasifikasi benar oleh model. Nilai ini juga dikenal sebagai *error* tipe 2.

		Predicted Class	
		Positive	Negative
Actual Class	Positive	TP	FN
	Negative	FP	TN

Gambar 2.20. Ilustrasi dari *confusion matrix*. [12]

Dengan menggunakan metrik statistik yang telah dijelaskan di atas, terdapat beberapa ukuran yang dapat diperoleh. Seperti akurasi, presisi (*precision*), atau sensitivitas (*recall/sensitivity*).

1. Akurasi merupakan ukuran dari seberapa banyak hasil prediksi yang benar terprediksi oleh model. Akurasi merupakan metrik dasar yang paling umum digunakan untuk melihat performa dari model. Akan tetapi, metrik ini menjadi tidak efisien digunakan saat dataset yang dilatih bersifat tidak seimbang (*unbalanced*). Perhitungan dari akurasi dapat dilihat sebagai berikut.

$$akurasi = \frac{TP + TN}{TP + TN + FP + FN} \quad (2-29)$$

2. Presisi (*precision*) merupakan metrik yang memberi tahu rasio dari prediksi yang bernilai positif dibandingkan dengan keseluruhan hasil yang diprediksi positif.

Precision dapat dihitung sebagai berikut.

$$precision = \frac{TP}{TP + FP} \quad (2-30)$$

3. Sensitivitas (*recall*) merupakan metrik yang memberi tahu rasio dari prediksi yang bernilai positif dibandingkan dengan keseluruhan hasil yang benar (baik itu TP maupun FN). *Recall* dapat dihitung sebagai berikut.

$$recall = \frac{TP}{TP + FN} \quad (2-31)$$

Metrik terakhir yakni F1-Score, yang merupakan perbandingan rata-rata dari *precision* dan *recall* yang dibobotkan. F1-Score menjadi umum dipakai dikarenakan dapat memberikan ide secara umum mengenai kedua metrik tersebut secara langsung. Metrik ini dapat dihitung sebagai berikut.

$$F1_Score = 2 \cdot \frac{recall \cdot precision}{recall + precision} \quad (2-32)$$

2.3 Analisis Perbandingan Metode

Berdasarkan tinjauan pustaka yang dilakukan pada bagian sebelumnya, terdapat dua metode yang sangat menonjol pada permasalahan *eye movement classification* ini, yaitu arsitektur 1D-CNN-BLSTM rancangan Startsev *et al.* [22] serta arsitektur TCN rancangan Elmadjian *et al* [23]. Kedua metode terbukti mampu mengklasifikasikan gerakan mata manusia dengan akurasi yang tinggi.

Tabel 2.2. Tabel perbandingan metode riset *eye movement classification* dengan menggunakan *F1 Score*.

Model	F1 Fixation	F1 Saccade	F1 Smooth Pursuit
TCN + <i>grid search</i>			
(Elmadjian <i>et al.</i> [23])	0,944	0,893	0,762
1D-CNN-BLSTM (Startsev <i>et al.</i> [22])	0,939	0,893	0,703
I-VMP	0,909	0,680	0,581
I-VDT	0,882	0,676	0,321
I-HMM	0,891	0,712	-
I-VT	0,891	0,705	-

Tabel 2.2 menunjukkan perbandingan performa dari beberapa metode yang digunakan dalam berbagai riset mengenai *eye movement classification*. Berdasarkan hasil

perbandingan di atas, terlihat bahwa TCN unggul di ketiga kelas mata, terutama pada gerakan *smooth pursuit*. TCN dalam implementasi yang dilakukan oleh Elmadjian *et al.* masih menggunakan metode *grid search* sebagai metode *hyperparameter optimization*. Adanya potensi peningkatan performa dari model TCN setelah diterapkannya metode *Hyperband* membuat model TCN rancangan Elmadjian *et al.* dipilih sebagai topik utama pada riset ini.

BAB III

METODE PENELITIAN

3.1 Alat dan Bahan Tugas Akhir

3.1.1 Alat Tugas Akhir

Tugas akhir ini dilakukan menggunakan alat-alat berupa perangkat keras dan perangkat lunak, yaitu:

1. *Notebook* dengan spesifikasi sistem operasi Windows 10, *processor* AMD Ryzen 2500U @ 2.0 GHz, memori 8GB DDR4, *graphic card* AMD Radeon Vega 8 Graphics, penyimpanan SATA HDD sebesar 1 TB dan SATA SSD sebesar 512 GB.
2. Visual Studio Code, yaitu perangkat lunak *code editor* yang digunakan dalam beberapa tahap pemrosesan data dengan menggunakan bahasa pemrograman Python.
3. Google Colab Pro, yaitu *platform* berbasis *cloud* yang menyediakan layanan *notebook* berbasis Jupyter dengan akses ke GPU virtual. Proses *training* pada penelitian ini sepenuhnya dilakukan dengan menggunakan GPU NVIDIA A100 dengan memori sebesar 40 GB. Google Colab Pro yang digunakan dijalankan pada *runtime* Python dengan versi 3.9.
4. *Browser* untuk mengakses Google Colab. Pada tugas akhir ini digunakan Microsoft Edge sebagai *browser* utama.
5. Python sebagai bahasa pemrograman untuk tugas akhir. Di dalam implementasinya, terdapat beberapa pustaka (*library*) yang digunakan pada proses pengolahan data, serta pemodelan arsitektur TCN, di antaranya:
 - Pandas, pustaka Python yang berfungsi dalam membaca serta memproses data berbasis tabular. Versi yang digunakan pada penelitian ini adalah versi 1.5.3.
 - NumPy, pustaka Python yang digunakan pada hampir keseluruhan pemrosesan matriks maupun *array* pada Python. Versi yang digunakan pada penelitian ini adalah 1.22.4.
 - Matplotlib, pustaka Python yang berfungsi untuk pembuatan visualisasi data seperti *lineplot*, *scatterplot*, maupun *training history plot*.
 - Pickle, pustaka Python yang berfungsi untuk melakukan *serialize* dan *de-serialize* pada objek berbasis Python, seperti model *deep learning*, hingga *array* dengan dimensi dan ukuran yang besar.
 - Scipy.io, pustaka Python yang khusus digunakan untuk membaca dan

memanipulasi *file* dengan format *.arff*, format yang digunakan pada dataset yang dipakai pada riset ini.

- Tensorflow, pustaka Python yang berfungsi dalam pembuatan, pelatihan, serta evaluasi model *deep learning* yang dibuat. Versi yang digunakan pada penelitian ini adalah versi 2.12.0.
- Keras-TCN, pustaka utama pada riset ini yang berfungsi untuk merancang arsitektur TCN pada Python. Versi yang digunakan pada penelitian ini adalah versi 3.5.0.

3.1.2 Bahan Tugas akhir

Bahan yang digunakan pada tugas akhir ini adalah Dataset GazeCom [2]. Dataset *GazeCom* merupakan dataset yang digunakan pada riset ini. Dataset *GazeCom* dipilih dikarenakan merupakan dataset terbesar yang dapat diakses secara publik dengan anotasi gerakan mata yang dilakukan secara manual. Dataset *GazeCom* merupakan dataset rekaman gerakan mata manusia yang dapat digunakan sebagai bahan latih untuk mengklasifikasikan gerakan mata manusia. Dataset ini terdiri dari 18 video, di mana tiap video terdiri dari 47 hingga 52 partisipan yang direkam gerakan matanya. Tiap partisipan akan dipersilakan untuk melihat sebuah video dengan durasi sekitar 20 detik, untuk kemudian direkam pergerakan matanya sepanjang durasi video.

Gerakan mata pada tiap partisipan ditangkap dengan menggunakan *eye tracker* dengan jenis *remote*, dengan menggunakan *sampling rate* sebesar 250 Hz, sehingga dihasilkan sekitar 5000 baris data pada setiap rekaman yang dilakukan. Pada tiap baris data terdapat kolom waktu, koordinat x dan y, serta label klasifikasi gerakan mata yang didapat dari hasil pelabelan yang dilakukan oleh dua sistem pakar (*expert system*). Pada dataset terdapat 4 label yang diberikan untuk masing-masing gerakan, yakni *fixation*, *saccade*, *smooth pursuit*, serta *noise*. *Noise* merupakan label yang diberikan saat terjadi galat pada *eye tracker* atau saat mata berkedip. Dari hasil pengamatan, ditemukan bahwa sekitar 72,5% dari dataset merupakan *fixation*, 10,5% *saccade*, 11% *smooth pursuit* dan 5,9% *noise*.

Gambar 3.1 menunjukkan cuplikan dari Dataset *GazeCom* yang digunakan. Deskripsi dari fitur-fitur yang terdapat pada dataset *GazeCom* tertera pada Tabel 3.1. Fitur *time* merupakan fitur waktu dari data *GazeCom*. Layaknya data *time series* pada umumnya, data *GazeCom* direkam dengan menggunakan *sampling rate* yang tetap, menghasilkan data secara kontinu. Fitur x dan y merupakan fitur koordinat dari gerakan mata yang direkam. Fitur *confidence* merupakan tingkat kepercayaan dari hasil anotasi yang dilakukan secara manual. Nilai *confidence* sebesar 1.0 menandakan bahwa label dianotasi dengan tingkat kepercayaan sebesar 100%, sedangkan nilai *confidence* sebesar 0.0 menandakan bahwa label dianotasi dengan tingkat kepercayaan sebesar 0%. Pada fitur *handlabe-*

ller, terdapat tiga jenis fitur yang berada pada dataset. *handlabeller1* dan *handlabeller2* masing-masing merupakan hasil anotasi yang diberikan oleh *expert system* pertama dan kedua, sedangkan *handlabeller_final* merupakan agregasi dari kedua anotasi yang dilakukan, sehingga dijadikan sebagai label utama pada dataset GazeCom ini. Pada fitur ketiga fitur label, nilai 1.0 menandakan gerakan *fixation*, 2.0 menandakan *saccade*, 3.0 menandakan *smooth pursuit*, dan 4.0 menandakan *noise*.

	time	x	y	confidence	handlabeller1	handlabeller2	handlabeller_final
0	1000.0	590.9	5.2	1.0	4.0	4.0	4.0
1	5000.0	590.9	5.2	1.0	4.0	4.0	4.0
2	9000.0	590.6	5.0	1.0	4.0	4.0	4.0
3	13000.0	590.4	5.0	1.0	4.0	4.0	4.0
4	17000.0	589.8	5.2	1.0	4.0	4.0	4.0
...
5003	20124000.0	707.6	665.2	1.0	1.0	1.0	1.0
5004	20128000.0	706.0	668.1	1.0	1.0	1.0	1.0
5005	20132000.0	706.7	670.2	1.0	1.0	1.0	1.0
5006	20136000.0	707.6	675.2	1.0	1.0	1.0	1.0
5007	20140000.0	708.0	678.6	1.0	1.0	1.0	1.0

5008 rows × 7 columns

Gambar 3.1. Contoh cuplikan dataset GazeCom pada salah satu *file*.

Tabel 3.1. Deskripsi fitur pada dataset GazeCom [2].

Fitur	Deskripsi
<i>time</i>	Waktu dari data direkam. Fitur ini direpresentasikan dalam satuan mikronsekon.
<i>x</i>	Koordinat x (horizontal) dari gerakan mata.
<i>y</i>	Koordinat y (vertikal) dari gerakan mata.
<i>confidence</i>	Tingkat kepercayaan (<i>confidence</i>) pada label yang diberikan.
<i>handlabeller1</i>	Hasil label yang diberikan oleh <i>expert system</i> pertama.
<i>handlabeller2</i>	Hasil label yang diberikan oleh <i>expert system</i> kedua.
<i>handlabeller_final</i>	Hasil label yang didapat dari agregasi kedua <i>expert system</i>

3.2 Metode yang Digunakan

Penelitian ini merupakan penelitian yang bersifat eksperimental, dengan tujuan untuk mengoptimalkan *hyperparameter* pada model TCN yang menjadi *state-of-the-art* pada permasalahan *eye movement classification*. Penelitian dimulai dengan pemahaman data, praproses data, pemodelan arsitektur TCN, proses *hyperparameter tuning*, serta evaluasi hasil model.

3.2.1 Tahapan Penerapan Metode

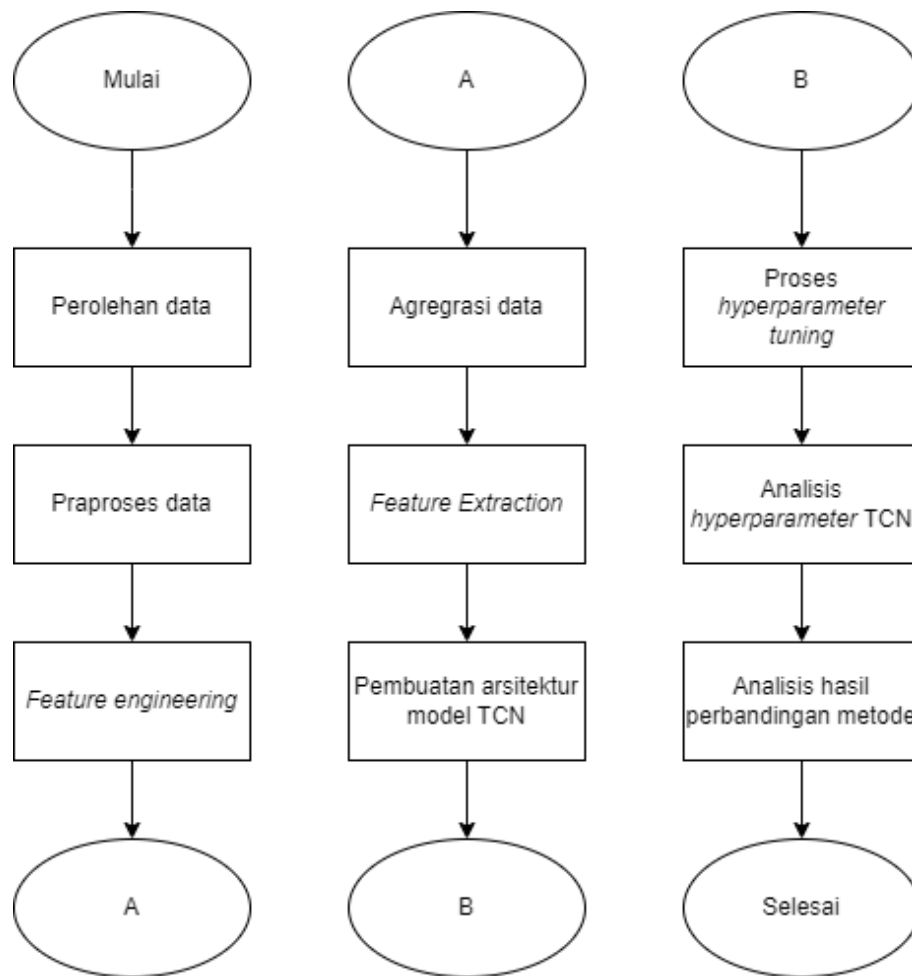
Pada penelitian, analisis akan dilakukan setelah model TCN hasil *tuning* telah selesai dirancang. Berdasarkan tinjauan pustaka pada Bab 2, disebutkan bahwa tujuan pada *eye movement classification* adalah untuk mengklasifikasikan tiga jenis gerakan mata, yakni *fixation*, *saccade*, serta *smooth pursuit*. Model TCN yang dirancang oleh Elmadjian *et al* [23] memiliki sekumpulan nilai *hyperparameter* yang didapat dengan menggunakan metode *Grid Search*. Penelitian ini akan berfokus pada peningkatan performa dan optimalisasi pada pencarian *hyperparameter* tersebut dengan menggunakan metode *Hyperband*, serta analisis pengaruh masing-masing *hyperparameter* pada model arsitektur TCN. Proses evaluasi performa model akan dilakukan dengan dua cara, yaitu teknik *Leave One Video Out* (LOVO) yang merupakan variasi dari teknik LOO serta *K-Fold Cross Validation*.

Dalam penelitian ini, penulis menggunakan bahasa pemrograman Python pada keseluruhan proses penelitian. Penulis menggunakan pustaka Pandas, Numpy serta Sci-py.io pada proses pengolahan serta manipulasi data mula-mula, Matplotlib untuk visualisasi data, Tensorflow dan TCN untuk pembuatan arsitektur TCN, serta untuk proses *hyperparameter tuning*.

Kelebihan dari penggunaan metode *Hyperband* dibandingkan metode yang lain terletak pada alokasi sumber daya serta penghematan waktu. Metode *Hyperband* bersifat lebih cepat untuk konvergen ke nilai yang optimal, dan sangat diunggulkan terutama pada dataset yang berukuran sangat besar, seperti pada dataset GazeCom.

3.3 Alur Tugas Akhir

Alur dari tugas akhir digambarkan pada Gambar 3.2. Penjelasan lengkap dari tiap proses pada alur adalah sebagai berikut.



Gambar 3.2. Alur dari tugas akhir.

3.3.1 Perolehan Data

Pada tahap ini, dilakukan perolehan dataset GazeCom terlebih dahulu sebelum diolah. Dataset GazeCom bersifat *open source*, sehingga dapat diunduh dan diolah dengan mudah.

3.3.2 Praproses Data

Tahap selanjutnya adalah tahap praproses data (*data preprocessing*). Pada tahap ini, mula-mula dataset GazeCom harus dapat dibaca terlebih dahulu. Dataset GazeCom disimpan ke dalam format *.arff* (*Attribute-Relation File Format*). Format ini berisikan dua bagian, *header*, dan *data*. Bagian *header* umumnya berisikan penjelasan atribut serta metadata dari dataset, sedangkan bagian *data* berisikan keseluruhan dari data.

Baik Elmadjian *et al* [23] maupun Startsev *et al* [22] mengolah serta memanipulasi keseluruhan *file* ARFF dengan menggunakan pustaka *liac-arff*, yang dapat digunakan untuk membaca maupun mengolah *file* ARFF. Sehingga keseluruhan tahap praproses hingga pemodelan dilakukan dengan menggunakan data ARFF tersebut. Namun, data

ARFF memiliki tingkat keterbacaan (*readability*) yang tidak terlalu tinggi. Berdasarkan hal tersebut, beserta dengan faktor keleluasaan, penulis memutuskan untuk mengubah keseluruhan *file* menjadi ke dalam format .csv. Format .csv (*Comma-Separated Value*) merupakan format basis data tabular yang sangat umum digunakan terutama pada pemrosesan data. *File* ARFF dapat dibaca dan diolah ke dalam bentuk NumPy Array dengan menggunakan bantuan pustaka Scipy.io, sehingga dapat diubah ke dalam bentuk Pandas DataFrame, dan dapat disimpan ke dalam *format* CSV.

3.3.3 Feature Engineering

Tahap *feature engineering* merupakan salah satu tahap yang krusial pada penelitian ini. Pada tahap ini, ditambahkan fitur-fitur baru pembentuk masing-masing baris data. Fitur-fitur ini dibuat dengan menggunakan serta memanipulasi fitur-fitur yang sudah terdapat pada dataset sebelumnya, seperti waktu (*time*), serta koordinat x dan y. Startsev *et al* [22] menambahkan tiga fitur tambahan pada dataset GazeCom sebelum dimasukkan ke dalam model *deep learning*, yaitu fitur kecepatan (*speed*), percepatan (*acceleration*), serta arah (*direction*). Lanjut, Elmadjian *et al* [23] menambahkan dua fitur tambahan, yaitu deviasi standar (*standard deviation*) serta perpindahan (*distance*).

Kelima fitur tambahan ini diekstrak dengan menggunakan *temporal window size* yang berbeda-beda, yakni 1, 2, 4, 8, 16, 32, 64, dan 128. Pemilihan jumlah *window* disesuaikan pada konfigurasi yang dilakukan pada penelitian Elmadjian *et al*, di mana pemilihan hingga ukuran 128 dimaksudkan agar semakin banyak konteks waktu yang tertangkap oleh tiap fitur data. *Temporal window* ini berfungsi sebagai batas yang digunakan dalam pembuatan fitur. Hal ini bertujuan tiap titik data dapat merepresentasikan hubungan temporal dengan titik-titik data lainnya dalam tingkat fitur. Semakin besar ukuran *temporal window* yang dipilih, maka akan terlihat relasi dari data dalam jangka waktu yang panjang. Karena dataset direkam dengan *sampling rate* sebesar 250 Hz, maka *window size* 128 merepresentasikan jarak waktu sebesar 512 milisekon. Gambar 3.3 menunjukkan hasil dari proses *feature engineering* yang menampilkan beberapa fitur tambahan dengan ukuran *temporal window* yang diekstrak. Dari hasil *feature engineering* fitur dataset bertambah menjadi 47 fitur, dengan tambahan sebanyak 40 fitur. Pada tahap ini beberapa fitur juga dibuang, seperti *confidence*, *handlabeller_1* serta *handlabeller_2* dikarenakan tidak diperlukan lagi. Tabel 3.2 menunjukkan deskripsi dari tiap fitur yang ditambahkan.

	speed_1	direction_1	acceleration_1	std_1	displacement_1	speed_2	direction_2	acceleration_2	std_2	displacement_2	...
0	0.000000	0.000000	0.000000e+00	0.000	0.000000	3962.953444	-1.056345	0.000000	5.400000	15.851814	...
1	3962.953444	-1.056345	9.907384e+05	5.400	15.851814	940.495215	-1.980924	869654.430291	5.092875	7.523962	...
2	3204.001404	2.573068	1.739309e+06	4.425	12.816006	2337.733946	2.416727	694207.337994	5.531321	18.701872	...
3	1590.793827	2.097740	4.829920e+05	2.175	6.363175	2412.500000	2.085444	196676.487219	5.470038	19.300000	...
4	3234.385568	2.079396	4.110296e+05	4.400	12.937542	4046.159444	2.748465	652159.873037	9.236927	32.369276	...
...
5011	79.056942	-1.892547	2.500000e+04	0.100	0.316228	80.039053	-0.896055	20963.137289	0.210753	0.640312	...
5012	134.629120	-0.380506	3.801727e+04	0.175	0.538516	62.500000	0.000000	15934.435980	0.164992	0.500000	...
5013	50.000000	1.570796	4.001953e+04	0.050	0.200000	79.056942	0.321751	6987.712430	0.188562	0.632456	...
5014	150.000000	0.000000	3.952847e+04	0.150	0.600000	170.018381	0.298499	22750.343404	0.359904	1.360147	...
5015	201.556444	0.519146	2.576941e+04	0.275	0.806226	201.556444	0.519146	12884.705080	0.275000	0.806226	...

5016 rows x 40 columns

Gambar 3.3. Contoh dari dataset hasil ekstraksi fitur baru.

Tabel 3.2. Deskripsi dari fitur-fitur hasil proses *feature engineering* pada dataset GazeCom.

Fitur	Deskripsi
<i>Speed_i</i>	Kecepatan yang ditempuh dari dua titik tepi dalam satu <i>temporal window</i>
<i>Acceleration_i</i>	Percepatan yang ditempuh dari dua titik tepi dalam satu <i>temporal window</i>
<i>Direction_i</i>	Arah yang ditempuh dari dua titik tepi dalam satu <i>temporal window</i>
<i>Stddev_i</i>	Devisasi standar dari keseluruhan data dalam satu <i>temporal window</i>
<i>Displacement_i</i>	Perpindahan yang ditempuh dari dua titik tepi dalam satu <i>temporal window</i>

3.3.4 Agregasi Data

Dataset GazeCom baik sebelum dipraproses maupun setelah dipraproses masih terpisah ke dalam beberapa *folder* dan tiap rekaman disimpan ke dalam satu *file* individu. Oleh karena itu, tahap agregasi data ini dilakukan untuk mengumpulkan keseluruhan data hasil *feature engineering* menjadi satu sebelum dimasukkan ke dalam model TCN. Tahap agregasi ini dilakukan dengan menggunakan pustaka NumPy sebagai pustaka utama dalam pemrosesan data.

Sebelum melakukan agregasi, beberapa fitur perlu dikonversi terlebih dahulu agar dapat merepresentasikan data lebih baik. Konversi ini dilakukan dengan membagi fitur koordinat seperti x dan y, fitur kecepatan, akselerasi, serta arah dengan sebuah nilai konstan yang disebut sebagai *Pixel per Degree* (PPD). Nilai PPD ini merupakan nilai yang bersifat unik pada tiap jenis *eye tracker* serta perangkat monitor yang digunakan dalam perekaman gerakan mata. Sehingga proses konversi ini dilakukan agar dataset bersifat general, dan model yang dilatih dengan data tersebut dapat digunakan pada data baru

yang ditangkap dengan menggunakan prosedur yang berbeda.

Algoritme 7 Algoritme Perhitungan PPD

Masukan:

w_{px} = Lebar dari resolusi layar yang digunakan pada proses penangkapan data (dalam satuan piksel)

h_{px} = Tinggi dari resolusi layar yang digunakan pada proses penangkapan data (dalam satuan piksel)

w_{mm} = Lebar dari resolusi layar yang digunakan pada proses penangkapan data (dalam satuan mm)

h_{mm} = Tinggi dari resolusi layar yang digunakan pada proses penangkapan data (dalam satuan mm)

d = Jarak rata-rata dari partisipan terhadap alat *eye tracker* maupun layar yang digunakan

Luaran: ppd = Konstanta *pixel per degree*

Algoritme:

Inisialisasi ppd_x, ppd_y konstanta PPD dalam sumbu x dan $y = 0$

Inisialisasi θ_x, θ_y , sudut yang dibentuk antara layar dan partisipan dalam sumbu x dan $y = 0$

$$\theta_x = 2\arctan^{-1}\left(\frac{w_{mm}}{2d}\right)\frac{180}{\pi}$$

$$\theta_y = 2\arctan^{-1}\left(\frac{h_{mm}}{2d}\right)\frac{180}{\pi}$$

$$ppd_x = \frac{w_{px}}{\theta_x}$$

$$ppd_y = \frac{h_{px}}{\theta_y}$$

$$ppd = \frac{ppd_x + ppd_y}{2}$$

Gambar 3.4. *Pseudocode* dari proses perhitungan *pixel per degree* (PPD).

Setelah dilakukan konversi, keseluruhan *file* akan digabung menjadi satu dan dikelompokkan berdasarkan jenis video (tiap *file* dengan jenis video yang sama sebelumnya dikelompokkan ke dalam satu *folder*). Dikarenakan terdapat 18 video dalam proses penangkapan data, maka akan terdapat 18 matriks pada dimensi pertama data hasil agregasi. Tiap matriks video ini akan diisi dengan keseluruhan *file* yang berada pada video itu. Sebagai contoh, pada video dengan nama *beach* memiliki total 47 *file*, di mana masing-masing *file* berisikan sekitar 5000 baris data dan 47 kolom dari hasil *feature engineering* sebelumnya. Oleh karena itu, data hasil agregasi akan memiliki bentuk multidimensi, di mana dimensi pertama akan merepresentasikan jumlah video, dimensi kedua merepresentasikan jumlah *file* dalam satu video, dan dimensi ketiga merepresentasikan jumlah baris dan kolom dalam satu *file*.

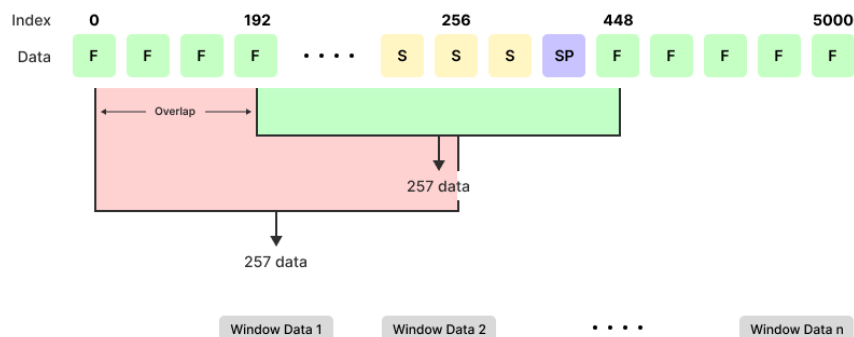
Pada proses ini juga akan dilakukan pemisahan antara data latih (yang hanya beri-

sikan fitur-fitur yang akan dimasukkan ke dalam model) serta label yang akan diprediksi. Dikarenakan fitur label bersifat kategorikal (memiliki lebih dari dua nilai unik yang saling independen), maka dilakukan proses *one hot encoding* terlebih dahulu. *One hot encoding* merupakan teknik yang umum digunakan pada fitur kategorikal untuk memecah fitur kategorikal dengan n jumlah nilai unik ke dalam n fitur independen baru namun dengan nilai biner, di mana 1 merepresentasikan kelas itu sendiri, sedangkan 0 tidak.

Keseluruhan hasil agregasi data latih, termasuk labelnya akan disimpan ke dalam format HDF5, yang umum digunakan dalam menyimpan data yang berukuran besar, dan tidak bersifat tabular.

3.3.5 Feature Extraction

Sebelum dataset GazeCom yang telah diagregasi dimasukkan ke dalam model, perlu dilakukan ekstraksi fitur dengan membentuk ulang (*reshape*) data agar dapat dimasukkan ke dalam model TCN. Sebagaimana yang telah dijelaskan pada tinjauan pustaka di Bab 2 mengenai TCN, arsitektur TCN memiliki lapisan dekonvolusi yang membuat keluaran dari arsitektur dapat memiliki panjang yang sama dengan masukan data. Oleh karena itu, kita ingin memanfaatkan properti dari arsitektur ini dengan melakukan *sampling* pada data menggunakan *temporal window* dengan ukuran tetap.



Gambar 3.5. Proses ekstraksi fitur dengan menggunakan *temporal window* berukuran 257 dengan *overlap* sebesar 192. Kode F menandakan gerakan *fixation*, S menandakan *saccade*, dan SP menandakan *smooth pursuit*.

Temporal window yang berisikan sampel berukuran tetap dari dataset ini yang akan menjadi fitur yang dimasukkan ke dalam model TCN. Ukuran dari *temporal window* ini ditetapkan terlebih dahulu sebelum proses ekstraksi dimulai. Elmadjian *et al.* melakukan percobaan dengan menggunakan tiga ukuran *window*, yakni 256, 385, dan 514 sampel, yang merepresentasikan masing-masing 1, 1.5, serta 2 detik dari data yang diambil. Gambar 3.5 menunjukkan proses ekstraksi fitur yang dilakukan dengan menggunakan *window size* sebesar 256, dengan *overlap* sebesar 192.

3.3.6 Pembuatan Arsitektur Model TCN

Setelah proses ekstraksi fitur selesai, maka pembuatan model TCN dapat dilakukan. Model TCN dibuat dengan menggunakan pustaka Keras-TCN. Arsitektur TCN dibangun dengan menggunakan *optimizer* yang sama dengan rancangan Elmadjian *et al.*, yakni *categorical cross-entropy*, yang cocok digunakan pada kasus klasifikasi multilabel. Model TCN akan menerima masukan dengan ukuran $(None, a, b)$, di mana *None* merepresentasikan jumlah dari data yang dimasukkan (jika diinginkan beberapa prediksi dari beberapa kumpulan data sekaligus), *a* merepresentasikan ukuran dari *window* yang dipakai dalam mengekstraksi data (sebagai contoh 257 sampel), dan *b* merepresentasikan jumlah fitur yang dipilih dari hasil *feature engineering* yang dilakukan sebelumnya. Berdasarkan penelitian yang dilakukan oleh Startsev *et al.*, ditemukan bahwa penggunaan fitur akselerasi pada model justru berpotensi mengurangi performa dari model secara keseluruhan. Lanjut, Elmadjian *et al.* menggunakan empat fitur utama, yakni kecepatan, arah, perpindahan, serta deviasi standar. Berdasarkan hal tersebut, pemodelan dilakukan dengan menggunakan empat fitur utama tersebut sebagai fitur masukan.

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 385, 28)]	0
tcn (TCN)	(None, 385, 128)	2000512
time_distributed (TimeDistributed)	(None, 385, 5)	645
Total params: 2,001,157		
Trainable params: 2,001,157		
Non-trainable params: 0		

Gambar 3.6. Contoh kerangka arsitektur model TCN yang dibuat dengan menggunakan 385 *temporal window* dengan jumlah fitur sebesar 28.

Proses *training* yang dilakukan oleh Elmadjian *et al.* adalah dengan menggunakan *batch size* sebesar 128 serta *epoch* sebanyak 20. Sehingga untuk parameter *epoch* akan diuji dengan menggunakan *epoch* sebanyak 20 juga. Untuk melakukan evaluasi terhadap performa dari model, digunakan mekanisme validasi khusus yang disebut sebagai *Leave One Video Out* (LOVO), yang sebelumnya diajukan oleh Startsev *et al* [22]. Teknik evaluasi ini merupakan teknik di mana proses *training* dilakukan secara iteratif dengan menggunakan keseluruhan data pada 17 dari 18 video, dan melakukan evaluasi pada 1 video terakhir. Hal ini bertujuan agar dapat dihasilkan model yang dapat digeneralisasi pada jenis video yang belum pernah dilihat sebelumnya.

Selain teknik validasi LOVO, akan dicoba juga metode validasi menggunakan

K-Fold Cross Validation, yang merupakan teknik validasi yang juga umum digunakan. Kedua metode validasi akan diukur dengan menggunakan metrik F1 Score, berdasarkan fakta bahwa dataset GazeCom memiliki persebaran label yang tidak seimbang (lebih dari 70% dari dataset memiliki label *fixation*).

3.3.7 Proses *Hyperparameter Tuning*

Proses *hyperparameter tuning* dilakukan pada model TCN yang sudah dibuat. Metode *tuning* yang digunakan adalah metode *Hyperband*. Sebelum melakukan proses *tuning*, perlu didefinisikan beberapa *hyperparameter* yang akan diuji terlebih dahulu beserta dengan rentang nilai yang akan digunakan. Hal ini disebut sebagai *hyperparameter space*. Tabel 3.3 menunjukkan keseluruhan *hyperparameter* yang dipakai dalam proses *tuning* beserta dengan rentang nilai yang digunakan. Pemilihan rentang nilai *hyperparameter* hanya didasarkan pada rentang nilai yang umumnya digunakan pada masing-masing *hyperparameter*. Pemilihan rentang nilai yang lebih spesifik memerlukan *expert system* yang sesuai.

Tabel 3.3. *Hyperparameter space* yang digunakan pada proses *hyperparameter tuning* menggunakan *Hyperband*.

<i>Hyperparameter</i>	Tipe Data	Rentang Nilai
<i>dropout_rate</i>	<i>float</i>	0,1, 0,2, 0,3
<i>number of filters</i>	<i>integer</i>	64, 96, 128
<i>number of stacks</i>	<i>integer</i>	1, 2, 3, 4, 5
<i>kernel size</i>	<i>integer</i>	3, 4, 5, 6, 7, 8
<i>dilations rate</i>	<i>integer</i>	5, 6, 7, 8
<i>batch size</i>	<i>integer</i>	32, 64, 128
<i>activation function</i>	<i>categorical</i>	'relu', 'tanh'

Proses *tuning* dilakukan dengan menspesifikasikan objektif dari metrik yang ingin dimaksimalkan. Pada penelitian ini, penulis ingin memaksimalkan F1 Score pada data validasi dibandingkan dengan metrik akurasi yang digunakan secara *default*. Proses *tuning* dilakukan dengan menggunakan 80% dari dataset dalam proses *training*, dan 20% sisanya digunakan pada proses validasi. Metode *Hyperband* diatur dengan menggunakan jumlah *epoch* maksimal sebesar 20.

3.3.8 Analisis *Hyperparameter* TCN

Selain melakukan *hyperparameter tuning* untuk menghasilkan sekumpulan nilai *hyperparameter* yang terbaik, dilakukan juga analisis pada tiap *hyperparameter* yang

digunakan pada TCN. Proses ini dilakukan dengan melakukan *training* serta evaluasi secara iteratif dengan menggunakan nilai *hyperparameter* yang diuji secara berbeda-beda. Beberapa kondisi yang ingin dilihat, yakni performa pada kedua data *training* maupun *evaluasi*, serta melihat apakah ada indikasi terjadinya *overfitting* atau tidak.

3.3.9 Analisis Hasil Perbandingan Metode

Hasil dari model yang dirancang oleh Elmadjian *et al.* akan dibandingkan dengan model yang sudah di-*tuning* berdasarkan hasil dari penelitian ini. Komparasi dilakukan baik pada F1 Score keseluruhan data, maupun F1 Score dari masing-masing kelas data. F1 Score dipilih dikarenakan cocok digunakan pada dataset yang bersifat *imbalanced*, sehingga penilaian tidak hanya dilakukan berdasarkan benar atau tidaknya hasil prediksi seperti pada metrik akurasi, namun juga berdasarkan *precision* serta *recall* dari hasil prediksi. Elmadjian *et al.* pada risetnya juga menambahkan kelas *noise* sebagai kelas yang dievaluasi, dikarenakan jumlahnya yang cukup banyak. Oleh karena itu, terdapat empat kelas data yang akan dikomparasi. Selain menggunakan teknik validasi LOVO, komparasi juga akan dilakukan dengan menggunakan metode *K-Fold Cross Validation*. Dikarenakan penelitian yang dilakukan oleh Elmadjian *et al.* hanya berfokus pada teknik LOVO sebagai metode evaluasi, maka model konfigurasi TCN tersebut akan di-*training* ulang menggunakan metode evaluasi ini sebelum dikomparasi.

BAB IV

HASIL DAN PEMBAHASAN

4.1 *Hyperparameter Tuning* dengan metode *Hyperband*

Pada tahap ini, penulis melakukan *hyperparameter tuning* menggunakan metode *Hyperband* dengan menggunakan konfigurasi yang sesuai seperti yang dibahas pada Bab 3. Pada implementasinya, metode *Hyperband* akan mula-mula melakukan *training* dengan jumlah *epoch* yang sangat sedikit, yakni 2. Proses ini dilakukan sebagai langkah dari *Hyperband* untuk menyeleksi subset dari *hyperparameter* yang berpotensi untuk kemudian di-*training* lebih lanjut dengan menggunakan jumlah *epoch* yang lebih tinggi. Proses ini bersifat iteratif dengan adanya penambahan jumlah *epoch* yang digunakan pada tiap sesi *tuning*.

Tabel 4.1 menunjukkan perbandingan antara nilai-nilai *hyperparameter* sebelum dan sesudah di-*tuning* dengan menggunakan metode *Hyperband*. Hasil *tuning* menunjukkan terdapat empat parameter utama yang memberikan hasil terbaik pada proses pencarian dengan menggunakan *Hyperband*. Keempat parameter ini yaitu *dropout_rate*, *kernel size*, *number of stacks*, dan *dilations rate*. Baik jumlah filter pada arsitektur, *activation function* yang digunakan serta *batch size* yang dipakai pada proses *training* tidak mengalami perubahan.

Tabel 4.1. Perbandingan nilai *hyperparameter* sebelum dan sesudah di-*tuning* menggunakan metode *Hyperband*.

<i>Hyperparameter</i>	Hasil <i>grid search</i>	Hasil <i>hyperband</i>
<i>dropout_rate</i>	0,3	0,1
<i>number of filters</i>	128	128
<i>number of stacks</i>	1	3
<i>kernel size</i>	8	4
<i>dilations rate</i>	8	5
<i>batch size</i>	128	128
<i>activation function</i>	'tanh'	'tanh'

4.1.0.1 Evaluasi dengan Teknik LOVO

Evaluasi utama pada kasus *eye movement classification* ini dilakukan dengan menggunakan teknik validasi LOVO. Berdasarkan jumlah video yang terdapat pada dataset, maka akan dilakukan 18 iterasi yang menghasilkan 18 model individual yang dilatih dengan menggunakan kumpulan dataset yang berbeda-beda pula. Pada iterasi pertama,

video pertama pada urutan akan disisakan, di mana sisa video akan dijadikan sebagai bahan latih pada model TCN. Setelah proses *training* selesai, dilanjutkan dengan evaluasi pada video yang disisakan sebelumnya.

Tabel 4.2. Hasil evaluasi pada seluruh model hasil *tuning* metode *hyperband* dengan menggunakan teknik evaluasi LOVO.

<i>Evaluate on Video</i>	F1 Fixation	F1 Saccade	F1 SP	F1 Noise	F1 Total
<i>Beach</i>	0,9578	0,9066	0,5862	0,7701	0,8442
<i>Breite_strasse</i>	0,9568	0,9095	0,8432	0,7949	0,901
<i>Bridge_1</i>	0,9751	0,902	0	0,7109	0,7177
<i>Bridge_2</i>	0,8801	0,8822	0,6815	0,75	0,8388
<i>Bumblebee</i>	0,9442	0,8935	0,6286	0,6999	0,8333
<i>Doves</i>	0,9134	0,8919	0,6922	0,6348	0,8266
<i>Ducks_boat</i>	0,9431	0,9072	0,851	0,694	0,8791
<i>Ducks_children</i>	0,9397	0,891	0,8373	0,7389	0,8815
<i>Golf</i>	0,951	0,9183	0,7954	0,8347	0,8999
<i>Holsten_gate</i>	0,9593	0,9069	0,6105	0,6617	0,8277
<i>Koenigstrasse</i>	0,9435	0,9108	0,8836	0,766	0,9008
<i>Puppies</i>	0,9453	0,9129	0,5288	0,7335	0,8242
<i>Roundabout</i>	0,9351	0,9241	0,546	0,7518	0,8315
<i>Sea</i>	0,9275	0,8686	0,3584	0,4693	0,7248
<i>St_petri_gate</i>	0,969	0,8955	0,5001	0,6908	0,8112
<i>St_petri_market</i>	0,9653	0,9004	0,4537	0,7321	0,8103
<i>St_petri_mcdonalds</i>	0,943	0,9096	0,727	0,7187	0,8597
<i>Street</i>	0,9421	0,8784	0,8625	0,7719	0,891
<i>Overall</i>	0,9459	0,9014	0,758	0,7137	0,8638
<i>Overall Accuracy</i>	0,9103				

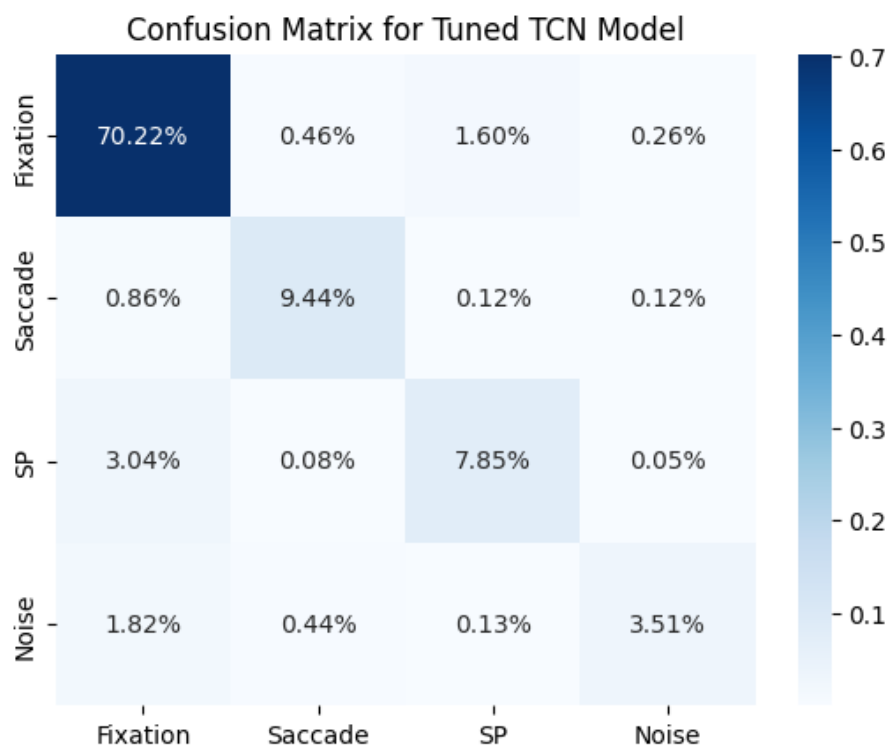
Tabel 4.3. Hasil evaluasi pada seluruh model hasil *grid search* dengan menggunakan teknik evaluasi LOVO.

<i>Evaluate on Video</i>	F1 Fixation	F1 Saccade	F1 SP	F1 Noise	F1 Total
<i>Beach</i>	0,9512	0,8966	0,5596	0,758	0,8331
<i>Breite_strasse</i>	0,9521	0,8952	0,8281	0,7815	0,8914
<i>Bridge_1</i>	0,9722	0,8876	0	0,711	0,7142
<i>Bridge_2</i>	0,884	0,872	0,7164	0,746	0,8437
<i>Bumblebee</i>	0,9572	0,8797	0,7526	0,6783	0,8536
<i>Doves</i>	0,9206	0,8846	0,6853	0,6289	0,8239
<i>Ducks_boat</i>	0,9428	0,8964	0,8574	0,6938	0,8781
<i>Ducks_children</i>	0,9339	0,8839	0,8124	0,7469	0,8754
<i>Golf</i>	0,9504	0,9089	0,7848	0,8131	0,8915
<i>Holsten_gate</i>	0,9592	0,9001	0,5498	0,6822	0,8183
<i>Koenigstrasse</i>	0,9399	0,9028	0,8784	0,7568	0,8956
<i>Puppies</i>	0,9362	0,8981	0,4789	0,7246	0,8076
<i>Roundabout</i>	0,938	0,9143	0,6079	0,7361	0,8393
<i>Sea</i>	0,9221	0,86	0,283	0,4595	0,7049
<i>St_petri_gate</i>	0,9664	0,8979	0,321	0,6867	0,7744
<i>St_petri_market</i>	0,9652	0,8945	0,4694	0,728	0,8114
<i>St_petri_mcdonalds</i>	0,9432	0,9006	0,7479	0,7107	0,8605
<i>Street</i>	0,9382	0,8633	0,858	0,7427	0,8805
<i>Overall</i>	0,9441	0,8921	0,7556	0,7039	0,8591
<i>Overall F1 Accuracy</i>	0,9074				

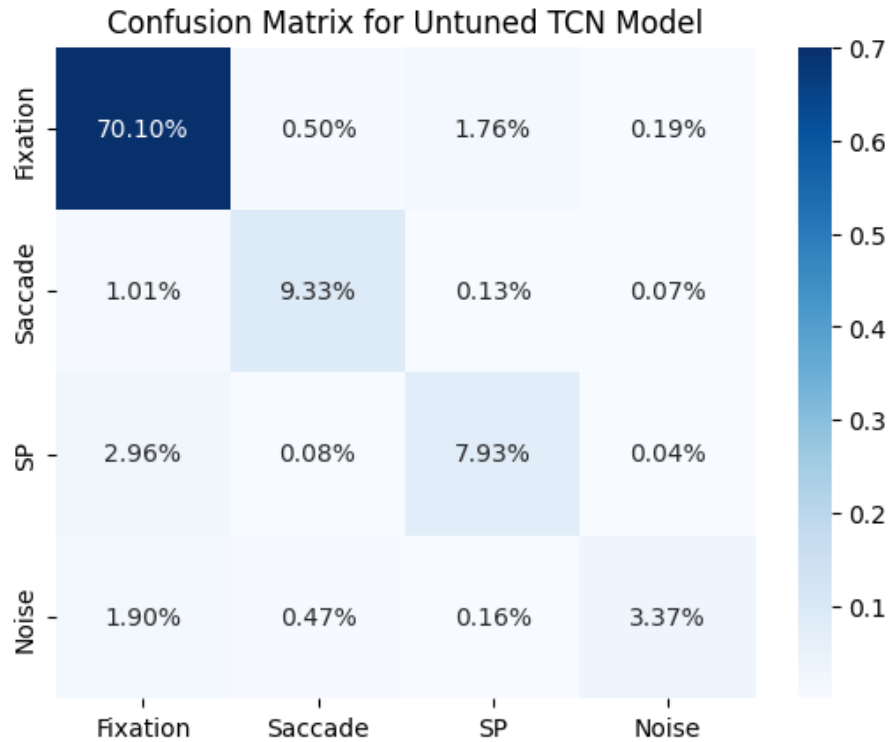
Tabel 4.2 dan 4.3 menunjukkan hasil evaluasi secara lengkap yang dilakukan dengan menggunakan teknik LOVO pada kedua jenis model. Pada F1 score untuk kelas *fixation*, rentang nilai evaluasi berada di antara 88 hingga 97, sedangkan pada kelas *saccade* berada di antara 86 hingga 91. Kelas *smooth pursuit* merupakan kelas yang memiliki persebaran data yang sangat beragam pada tiap video. Hal ini ditunjukkan pada hasil evaluasi di mana rentang nilai yang dihasilkan sangat berbeda jauh antar tiap video. Bahkan dapat dilihat bahwa pada video *bridge_1* menunjukkan tidak terdapat gerakan *smooth pursuit* yang terekam sedikitpun. Hal ini berbeda jauh dengan video seperti *street*, *breite_strasse*, dan *ducks_boat* yang dapat menghasilkan F1 score di atas 84 pada kedua model, menunjukkan banyaknya gerakan *smooth pursuit* yang terdapat pada

masing-masing video.

Gambar 4.1 dan 4.2 menunjukkan plot *confusion matrix* pada hasil evaluasi LO-VO masing-masing model TCN. *Confusion matrix* merupakan plot visualisasi yang menunjukkan performa dari sebuah model *supervised learning*, seperti pada kasus klasifikasi. Visualisasi ini akan menampilkan *true positive*, *true negative*, *false positive* serta *false negative* dari tiap kelas pada hasil klasifikasi. Dari kedua hasil plot dapat kita lihat bahwa 70% dari keseluruhan hasil prediksi merupakan *fixation*. Hal ini wajar mengingat persebaran dari dataset itu sendiri serta tidak adanya proses *undersampling* pada dataset untuk menyeimbangkan jumlah label pada dataset.



Gambar 4.1. Plot *confusion matrix* pada hasil evaluasi model TCN setelah proses *tuning* dengan *hyperband*.



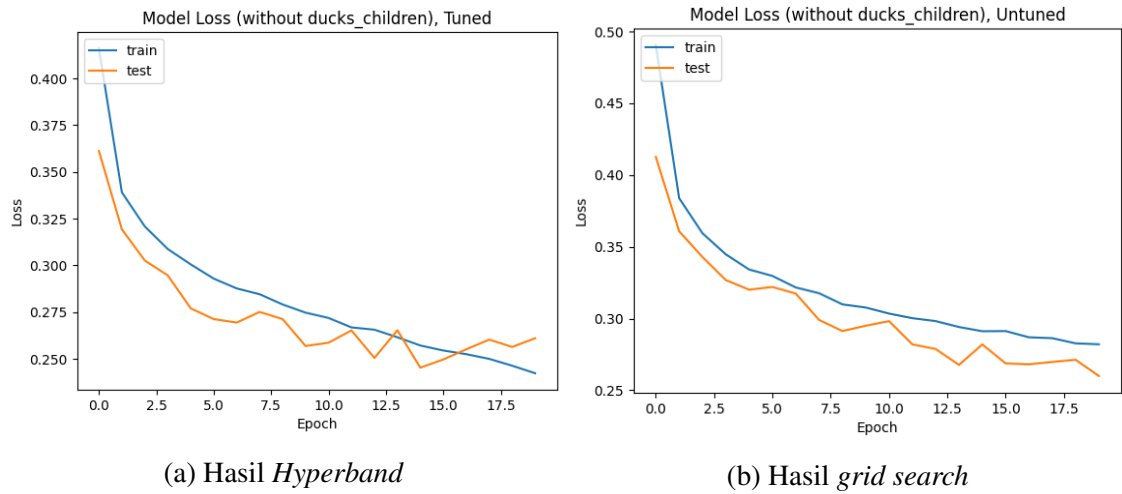
Gambar 4.2. Plot *confusion matrix* pada hasil evaluasi model TCN dengan menggunakan metode *grid search*.

Pada kelas *fixation*, sekitar 92,2% dari hasil evaluasi diprediksi benar sebagai *fixation*, di mana 3,8% diprediksi sebagai *smooth pursuit*, dan sisanya sebagai *noise* serta *saccade*, secara berturut-turut. Hal ini mengindikasikan bahwa gerakan *fixation* dapat secara keliru diidentifikasi sebagai *smooth pursuit*. Beberapa kemungkinan alasan adalah dikarenakan kedua gerakan memiliki kecepatan (*speed*) maupun perubahan arah yang cukup mirip, sedangkan antara *fixation* serta *saccade* memiliki karakteristik yang berbeda jauh, baik dari segi kecepatan, percepatan, maupun atribut yang lain.

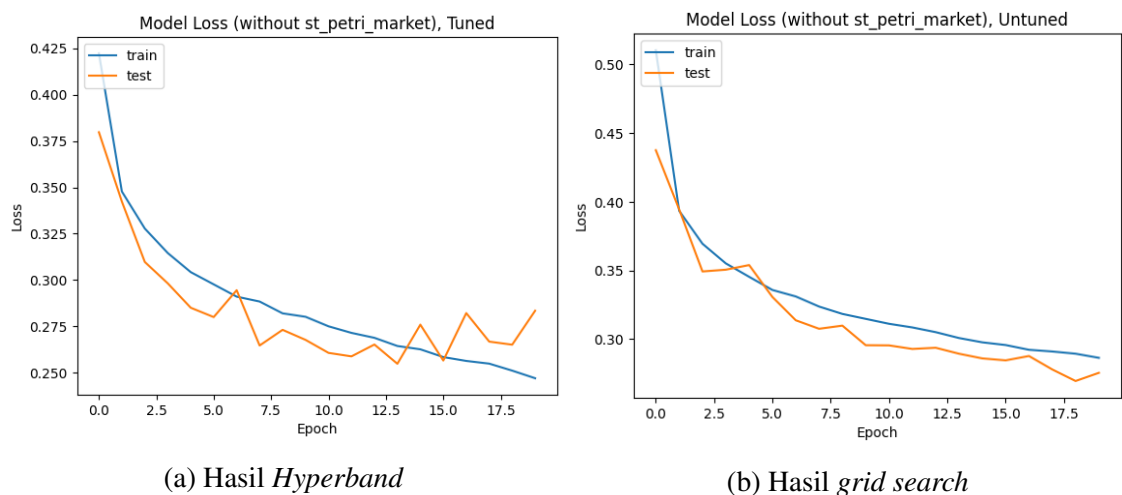
Pada kelas *saccade*, sekitar 91% dari hasil evaluasi diprediksi benar sebagai *saccade*, di mana sekitar masing-masing 4% diprediksi salah sebagai *noise* maupun *fixation*, dan sisanya sebagai *smooth pursuit*. Ini menandakan gerakan *saccade* dapat secara keliru diidentifikasi sebagai *noise*. *Noise* sendiri dapat terjadi saat adanya galat pada mesin, maupun saat mata berkedip. Proses mata berkedip ini umumnya dapat diidentifikasi pada data saat terjadinya perubahan letak koordinat secara tidak lazim, yang menandakan perubahan posisi serta kecepatan yang dihasilkan pula, yang memiliki karakteristik yang sama dengan *saccade*.

Pada kelas *smooth pursuit*, sekitar 80% dari hasil evaluasi diprediksi benar sebagai *smooth pursuit*. layaknya *fixation* yang umumnya paling banyak diidentifikasi salah sebagai *smooth pursuit*, *smooth pursuit* juga demikian, dengan persentase sebesar 17% dari hasil evaluasi salah diidentifikasi sebagai *fixation*. Kedua *noise* serta *saccade* tidak

memiliki persentase yang cukup besar, di mana kedua label hanya terprediksi sebesar 3% secara total.



Gambar 4.3. Perbandingan kedua plot *history* proses *training* pada iterasi video *ducks_children*.



Gambar 4.4. Perbandingan kedua plot *history* proses *training* pada iterasi video *St_petri_market*.

4.1.0.2 Evaluasi dengan Teknik *K-Fold Cross Validation*

Evaluasi kedua dilakukan dengan menggunakan teknik *K-Fold Cross Validation*. Evaluasi dengan teknik LOVO menunjukkan bahwa terdapat inkonsistensi antara masing-masing video, di mana terdapat beberapa video yang memiliki bobot *smooth pursuit* yang besar, serta beberapa video yang memiliki bobot sedikit hingga tidak ada sama sekali. Teknik *K-Fold Cross Validation* akan menggunakan keseluruhan dataset yang dibagi secara acak menjadi 5 bagian. Hal ini agar model dapat lebih belajar dengan menggunakan dataset yang lebih general. Secara penerapan secara *real time*, metode K-Fold lebih

diunggulkan dikarenakan dapat menghasilkan model yang telah mempelajari pola pada keseluruhan jenis video, dan akan lebih mudah digunakan jika ingin dievaluasi pada jenis video yang baru (di luar dari 18 video yang terdapat pada dataset GazeCom).

Tabel 4.4 menunjukkan hasil evaluasi dari model TCN yang menggunakan metode *grid search* dan yang menggunakan metode *hyperband* dengan menggunakan metode evaluasi *K-Fold Cross Validation*. Pada implementasinya, digunakan jumlah *fold* sebanyak 5, yang berarti ada 5 iterasi yang dilakukan dengan menggunakan data latih dan uji yang berbeda-beda tiap iterasinya. Berbeda dengan evaluasi dengan teknik LOVO, evaluasi dengan *K-Fold Cross Validation* menghasilkan nilai evaluasi yang lebih merata, dengan nilai deviasi standar hanya sebesar 0,01 untuk kelas *smooth pursuit*, berbeda jauh dengan deviasi standar yang didapat khususnya pada hasil evaluasi menggunakan metode LOVO pada kelas *smooth pursuit*, di mana didapat nilai sebesar 0,2.

Tabel 4.4. Hasil evaluasi pada seluruh model dengan menggunakan teknik evaluasi *K-Fold Cross Validation*.

Jenis Model	<i>Fold Number</i>	F1 Fixation	F1 Saccade	F1 SP	F1 Noise
TCN + <i>Hyperband</i>	1	0,9524	0,9109	0,7971	0,7234
	2	0,9533	0,9074	0,8293	0,7313
	3	0,9537	0,9072	0,8126	0,7216
	4	0,9554	0,911	0,8191	0,7336
	5	0,9541	0,9028	0,8251	0,74
	Rata-rata	0,9538	0,9079	0,8167	0,73
TCN + <i>grid search</i>	1	0,9457	0,8965	0,7577	0,7096
	2	0,9446	0,8966	0,7458	0,7041
	3	0,9432	0,8922	0,7698	0,6939
	4	0,9459	0,8957	0,7653	0,6999
	5	0,9466	0,8962	0,774	0,726
	Rata-rata	0,9452	0,8955	0,7625	0,7067

4.2 Analisis *Hyperparameter* pada Model TCN

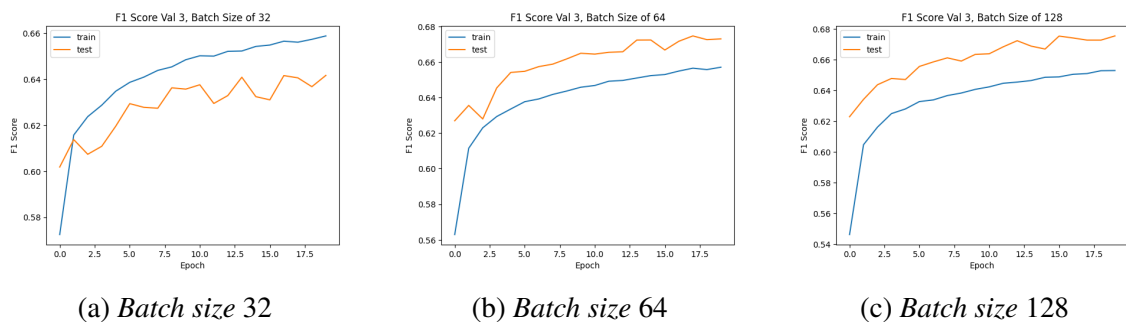
Analisis *hyperparameter* dilakukan untuk melihat pengaruh dari masing-masing *hyperparameter* baik pada arsitektur TCN maupun pada proses *training* itu sendiri. Beberapa *hyperparameter* yang diuji yakni *batch_size*, *dropout_rate*, jumlah *kernel*, jumlah *stack*, *dilation_rate*, serta jumlah filter. Keseluruhan *hyperparameter* dianalisis dengan menggunakan metode evaluasi *K-Fold Cross Validation* dengan menggunakan jumlah *fold* sebesar 5.

4.2.1 Batch Size

Terdapat tiga nilai *batch_size* yang diuji, yakni 32, 64, serta 128. Tabel 4.5 menunjukkan hasil komparasi dari masing-masing nilai pada *batch size* terhadap hasil evaluasi. Terlihat bahwa nilai *batch_size* sebesar 32 menghasilkan nilai evaluasi yang paling tinggi pula dibandingkan kedua nilai yang lain. Sedangkan dari segi waktu, justru *batch_size* dengan ukuran 128 menghasilkan waktu *training* yang lebih cepat dibandingkan dengan 32. Hal ini dikarenakan model akan melatih 128 kelompok data dalam satu sesi, sehingga dapat memangkas waktu lebih cepat.

Tabel 4.5. Perbandingan hasil evaluasi dari nilai-nilai pada *hyperparameter batch size* dalam performa TCN.

<i>Batch Size</i>	F1 Fixation (Rata-rata)	F1 Saccade (Rata-rata)	F1 SP (Rata-rata)	F1 Noise (Rata-rata)	Waktu dalam 1 epoch (detik)
32	0,948	0,9011	0,7811	0,7191	18-19
64	0,9467	0,8992	0,7766	0,719	12-13
128	0,9451	0,8957	0,7711	0,7148	11-12



Gambar 4.5. Perbandingan ketiga plot *history* pada proses *training* dengan ukuran *batch size* yang berbeda-beda

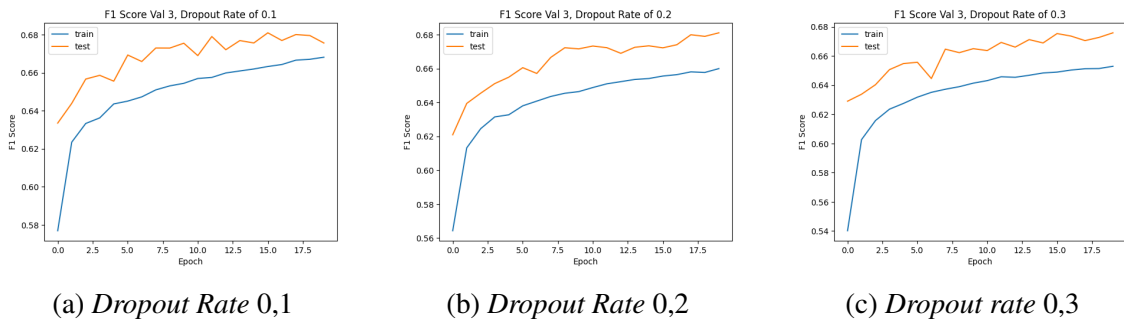
4.2.2 Dropout Rate

Terdapat tiga nilai *dropout rate* yang diuji, yakni 0,1, 0,2, serta 0,3. Tabel 4.6 menunjukkan hasil komparasi dari masing-masing nilai pada *dropout rate* terhadap hasil evaluasi. Terlihat pada tabel bahwa nilai *dropout rate* 0,1 menghasilkan nilai F1 score evaluasi yang paling baik dibandingkan dengan kedua nilai lainnya. Dikarenakan *hyperparameter dropout rate* mengatur seberapa banyak koneksi dari tiap *layer* yang ingin secara acak dihapus dan tidak terlibat dalam jumlah *neuron* ataupun *layer* yang ada, maka jumlah parameter *training* yang dilatih pun tidak berubah. Selain itu, keseluruhan nilai *dropout rate* menghasilkan waktu *training* per *epoch* yang sama.

Tabel 4.6. Perbandingan hasil evaluasi dari nilai-nilai pada *hyperparameter dropout rate* dalam performa TCN.

<i>Dropout Rate</i>	F1 Fixation (Rata-rata)	F1 Saccade (Rata-rata)	F1 SP (Rata-rata)	F1 Noise (Rata-rata)
0,1	0,9494	0,9004	0,7866	0,7257
0,2	0,946	0,8977	0,7781	0,7285
0,3	0,9455	0,8938	0,7687	0,7081

Untuk perbandingan plot *history* pada *hyperparameter dropout rate* dapat dilihat pada Gambar 4.6. Dapat dilihat bahwa ketiga model tidak memiliki potensi terjadinya *overfitting* maupun *underfitting*. Meskipun umumnya semakin kecil *dropout rate* yang digunakan maka peluang *overfitting* yang terjadi semakin besar, pada dataset GazeCom dapat dikatakan nilai *dropout rate* 0,1 masih dikatakan aman untuk digunakan pada pe-modelan.



Gambar 4.6. Perbandingan ketiga plot *history* pada proses *training* dengan ukuran *dropout rate* yang berbeda-beda.

4.2.3 Nb Filters

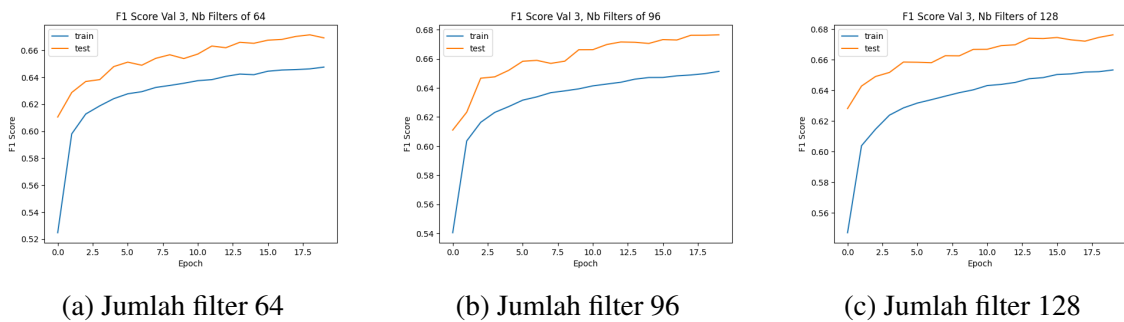
Pada *hyperparameter filter numbers* terdapat tiga nilai yang diuji, yaitu 64, 96, dan 128. Tabel 4.7 menunjukkan hasil evaluasi F1 Score dengan menggunakan nilai *filter numbers* yang berbeda-beda. Dikarenakan parameter ini mengatur jumlah filter yang digunakan pada TCN, yang secara langsung mengatur besarnya ukuran luaran yang akan dimasukkan pada *output layer*, maka jumlah *filter* yang digunakan sangat mempengaruhi jumlah parameter yang digunakan dalam proses *training*, yang menambah kompleksitas dari model serta mempengaruhi waktu pada proses *training* dalam satu *epoch*. Pada tabel, dapat dilihat bahwa hasil F1 score yang didapat sangat beragam, di mana nilai F1 score kelas *fixation* serta *noise* tertinggi didapat pada jumlah filter 128, sedangkan pada kelas *saccade* serta *smooth pursuit* didapat pada model dengan jumlah filter 96. Dikarenakan tingkat kompleksitas model yang berbanding terbalik dengan waktu *training* yang diper-

lukan pada satu *epoch*, model dengan jumlah filter 64 merupakan model dengan waktu *training* tercepat dibandingkan kedua model lainnya.

Tabel 4.7. Perbandingan hasil evaluasi dari nilai-nilai pada *hyperparameter number of filters* dalam performa TCN.

<i>Number of filters</i>	F1 Fixation (Rata-rata)	F1 Saccade (Rata-rata)	F1 SP (Rata-rata)	F1 Noise (Rata-rata)	Waktu/ <i>epoch</i>	Jumlah parameter
64	0,943	0,8898	0,7541	0,7106	6-7	508736
96	0,9442	0,9042	0,7686	0,7119	9	1131744
128	0,9455	0,8955	0,7667	0,724	12	2000512

Selain itu, Gambar 4.7 juga menunjukkan plot *history* perbandingan ketiga hasil *training* dengan menggunakan jumlah filter yang berbeda-beda. Dari hasil plot yang diperoleh dapat ditangkap bahwa ketiga model tidak memiliki indikasi terjadinya *overfitting*, terutama pada model dengan jumlah filter 128, yang memiliki kompleksitas tertinggi. Sehingga pada dataset GazeCom, *overfitting* tetap dapat dihindarkan meskipun menggunakan jumlah filter tertinggi.



Gambar 4.7. Perbandingan ketiga plot *history* pada proses *training* dengan jumlah filter yang berbeda-beda.

4.2.4 Nb Stacks

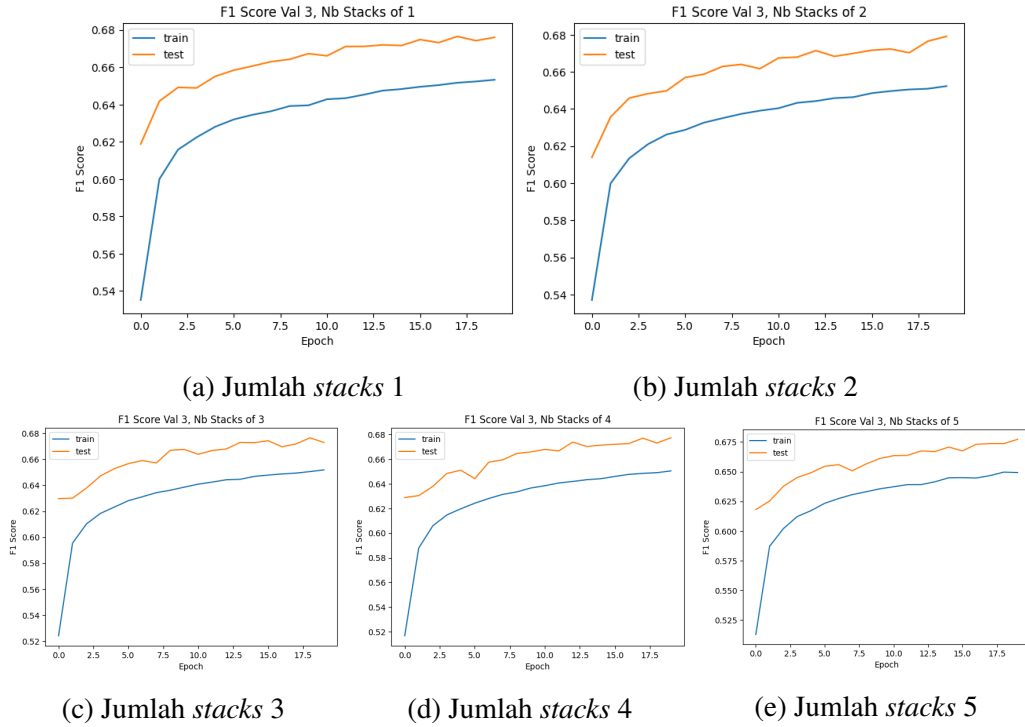
Pada *hyperparameter* jumlah *stack*, terdapat lima nilai yang diuji, yaitu 1, 2, 3, 4, dan 5. Tabel 4.9 menunjukkan perbandingan hasil evaluasi pada nilai-nilai *stacks number* yang diuji. Pada tabel dapat dilihat bahwa keseluruhan nilai memberikan hasil yang tidak berbeda jauh. Beberapa nilai parameter dapat menghasilkan hasil yang sama, di mana pada kelas *fixation* serta *saccade*, nilai yang tertinggi didapat oleh model dengan jumlah *stack* 1 dan 3, sedangkan pada kelas *smooth pursuit* didapat oleh jumlah *stack* 2, dan pada *noise* didapat oleh jumlah *stack* 1. Berbeda dengan *hyperparameter nb filters* yang umumnya didominasi oleh jumlah filter terbesar dengan kompleksitas yang lebih tinggi, hal yang berbeda terjadi pada jumlah *stacks*, setidaknya pada set parameter yang

diuji. Layaknya jumlah filter, jumlah *stack* juga mempengaruhi kompleksitas dari model, mengakibatkan waktu *training* yang lebih lama, hingga mencapai 52 detik untuk satu *epoch*.

Tabel 4.8. Perbandingan hasil evaluasi dari nilai-nilai pada *hyperparameter number of stacks* dalam performa TCN.

<i>Number of stacks</i>	F1 Fixation (Rata-rata)	F1 Saccade (Rata-rata)	F1 SP (Rata-rata)	F1 Noise (Rata-rata)	Waktu/ <i>epoch</i>	Jumlah parameter
1	0,945	0,8954	0,7694	0,7123	12	2000512
2	0,9447	0,8949	0,7711	0,7117	22	4099712
3	0,945	0,8954	0,7641	0,7109	32	6198912
4	0,9446	0,8924	0,7691	0,7097	42	8298112
5	0,9444	0,8916	0,762	0,7064	52	10397312

Selain itu, Gambar 4.8 juga menunjukkan plot *history* perbandingan kelima hasil *training* dengan menggunakan jumlah *stack* yang berbeda-beda. Sama halnya dengan jumlah *filter*, dari hasil plot yang diperoleh dapat ditangkap bahwa kelima model tidak memiliki indikasi terjadinya *overfitting*, terutama pada model dengan jumlah *stack* sebesar 5, yang memiliki kompleksitas tertinggi. Sehingga pada dataset GazeCom, *overfitting* juga tetap dapat dihindarkan meskipun menggunakan jumlah *stack* tertinggi, walaupun catatan utama terdapat pada *trade off* yang terjadi antara kompleksitas model dengan waktu *training* yang sangat besar.



Gambar 4.8. Perbandingan kelima plot *history* pada proses *training* dengan ukuran jumlah *stacks* yang berbeda-beda.

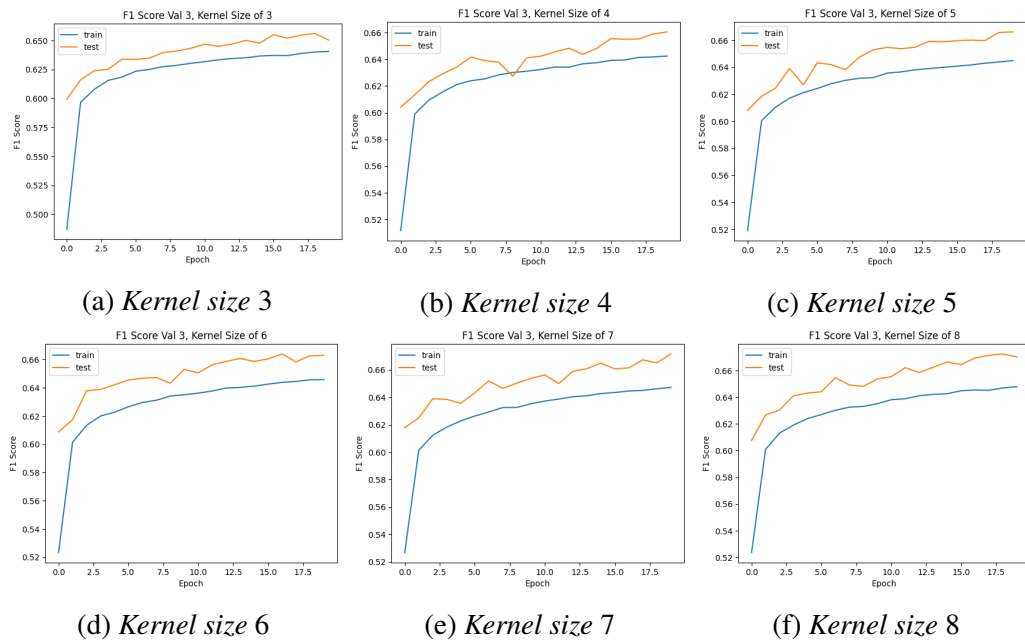
4.2.5 Kernel Size

Pada *hyperparameter kernel size*, terdapat 6 nilai yang diuji, yaitu 3, 4, 5, 6, 7, dan 8. Tabel 4.9 menunjukkan komparasi hasil evaluasi dari keenam ukuran kernel yang digunakan pada model. Satu hal yang perlu diperhatikan adalah meskipun ukuran dari kernel mempengaruhi kompleksitas dari model dengan menambah jumlah dari parameter *training*, pada proses *training* yang dilakukan tidak terdapat perubahan pada waktu *training* dalam satu *epoch*, yang berbeda dengan *hyperparameter* lain sebelumnya. Pada tabel dapat dilihat bahwa nilai F1 score tertinggi didominasi oleh model dengan ukuran kernel yaitu 8, dengan jumlah parameter *training* tertinggi.

Tabel 4.9. Perbandingan hasil evaluasi dari nilai-nilai pada *hyperparameter kernel size* dalam performa TCN.

<i>Kernel Size</i>	F1 Fixation (Rata-rata)	F1 Saccade (Rata-rata)	F1 SP (Rata-rata)	F1 Noise (Rata-rata)	Jumlah parameter
3	0,94	0,888	0,7184	0,7031	192576
4	0,9397	0,8877	0,7344	0,704	255808
5	0,9418	0,8906	0,7474	0,7078	319040
6	0,9424	0,8899	0,7532	0,7023	382272
7	0,9417	0,8906	0,7552	0,7085	445504
8	0,9435	0,8912	0,7547	0,7083	508736

Selain itu, Gambar 4.9 menunjukkan perbandingan plot *history* dari keenam model dengan menggunakan ukuran kernel yang berbeda-beda. Sama halnya dengan jumlah *filter* maupun *stacks*, dari hasil plot yang diperoleh dapat ditangkap bahwa keenam model tidak memiliki indikasi terjadinya *overfitting*.



Gambar 4.9. Perbandingan keenam plot *history* pada proses *training* dengan ukuran *kernel* yang berbeda-beda.

4.2.6 Dilation Rate

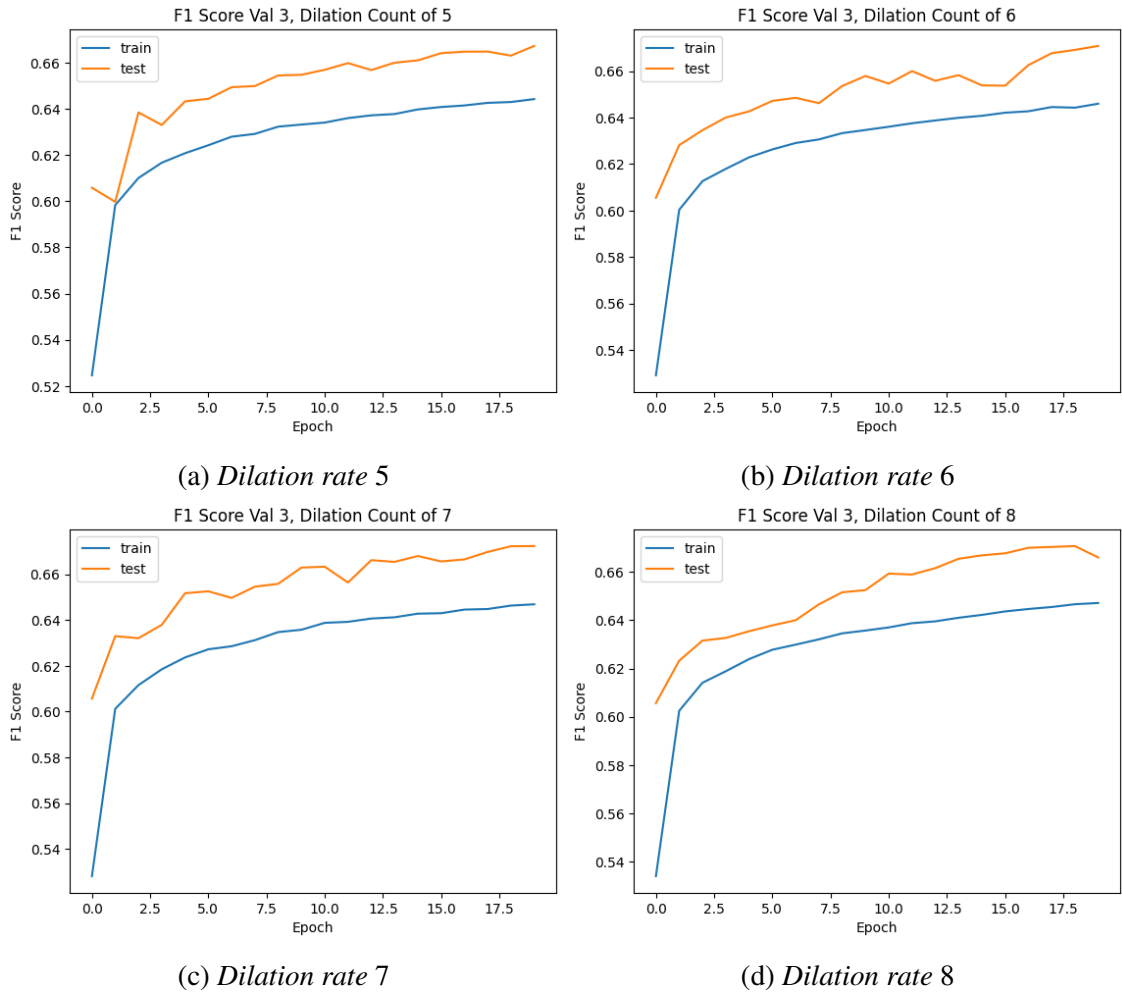
Hyperparameter terakhir yang dianalisis pada penelitian adalah *dilation rate*. Pada *hyperparameter* ini, terdapat 4 nilai *dilation rate* yang diuji, yaitu 5, 6, 7, dan 8. Tabel 4.10 menunjukkan perbandingan hasil evaluasi antara keempat nilai *dilation rate* yang

diuji. Pada *dilation rate*, waktu *training* dalam satu *epoch* bergantung pada kompleksitas model yang ditentukan oleh *dilation rate* yang digunakan. Pada tabel dapat dilihat bahwa *dilation rate* dengan nilai 7 dan 8 umumnya menghasilkan nilai evaluasi tertinggi. Namun, tidak terdapat perbedaan yang jauh antara keempat nilai.

Tabel 4.10. Perbandingan hasil evaluasi dari nilai-nilai pada *hyperparameter dilation rate* dalam performa TCN.

<i>Dilation Rate</i>	F1 Fixation (Rata-rata)	F1 Saccade (Rata-rata)	F1 SP (Rata-rata)	F1 Noise (Rata-rata)	Waktu/ <i>epoch</i>	Jumlah parameter
5	0,9417	0,8904	0,7477	0,7072	4	311744
6	0,942	0,8898	0,7589	0,705	5	377408
7	0,9414	0,8908	0,7535	0,7113	5	443072
8	0,9428	0,9016	0,7549	0,7076	6	508736

Selain itu, Gambar 4.10 menunjukkan perbandingan plot *history* dari keempat model dengan menggunakan nilai *dilation rate* yang berbeda-beda. Sama halnya dengan parameter-parameter yang lain, dari hasil plot yang diperoleh dapat ditangkap bahwa keempat model tidak memiliki indikasi terjadinya *overfitting*.



Gambar 4.10. Perbandingan keempat plot *history* pada proses *training* dengan *dilation rate* yang berbeda-beda.

4.3 Perbandingan dengan Penelitian Sebelumnya

Penulis akan membandingkan hasil penelitian dengan penelitian Elmadjian *et al.* yang menggunakan model TCN dengan metode *grid search*. Tabel 4.11 menunjukkan hasil akhir komparasi dari kedua model TCN. Model pertama yaitu model TCN yang menggunakan nilai-nilai parameter serta konfigurasi yang ditentukan oleh Elmadjian *et al.* dalam penelitiannya. Dengan menggunakan jumlah fitur yang sama, model TCN hasil *tuning* dengan metode *hyperband* mampu mengalahkan hasil evaluasi F1 score pada model milik Elmadjian *et al.* terutama pada kelas *saccade*, dengan peningkatan sebesar 1%. Pada publikasi yang dibuat oleh Elmadjian *et al.*, disebutkan bahwa pada kelas *smooth pursuit* dengan konfigurasi yang sama dapat menghasilkan F1 score sebesar 0,762. Namun, model tersebut dilabeli dengan catatan yang menandakan bahwa performa model tersebut didapat dari proses *training* yang dilakukan secara berulang-ulang hingga mendapatkan nilai tertinggi.

Lanjut, Elmadjian *et al.* juga menambahkan model lain dengan konfigurasi yang

sama namun dengan menggunakan mekanisme *output filtering* dengan mengasumsikan bahwa jika terdapat sekumpulan titik berdurasi kurang dari 12 milidetik dan memiliki label yang berbeda dengan mayoritas label dari titik sebelum dan sesudahnya, maka titik tersebut akan dilabeli ulang mengikuti label mayoritas tersebut. Proses *filtering* ini dapat menghasilkan F1 score pada kelas *smooth pursuit* mencapai 0,764. Namun, proses ini secara praktis tidak dapat diterapkan terutama pada kasus *real time*, sehingga penggunaan *filter* tersebut tidak digunakan pada penelitian ini, dan hanya berfokus pada luaran dari model secara umum.

Tabel 4.11. Perbandingan hasil evaluasi dari kedua model pada metode evaluasi LOVO dan *K-Fold Cross Validation*.

	F1 Fixation	F1 Saccade	F1 SP	F1 Noise	F1 Total
LOVO TCN + <i>Hyperband</i>	0,9459	0,9014	0,758	0,7137	0,8638
LOVO TCN + <i>grid search</i>	0,9441	0,8921	0,7556	0,7039	0,8591
KFold TCN + <i>Hyperband</i>	0,9538	0,9079	0,8167	0,73	-
KFold TCN + <i>grid search</i>	0,9452	0,8955	0,7625	0,7067	-

Hal yang menarik terdapat pada hasil evaluasi dengan menggunakan metode *K-Fold Cross Validation*, di mana hasil evaluasi F1 score pada keseluruhan kelas mengalami peningkatan yang cukup tinggi, terutama pada kelas *smooth pursuit* yang mencapai 0,8167. Pada kelas *fixation*, peningkatan terjadi sebesar 0,9%, sedangkan *saccade* sebesar 1,3%, *noise* sebesar 3,2%, dan *smooth pursuit* sebesar 6,6%. Hal ini menandakan bahwa model TCN hasil *tuning* dengan menggunakan metode *Hyperband* dapat mempelajari pola pada dataset latih lebih baik, terutama untuk gerakan *smooth pursuit*.

Selain itu, dari hasil penelitian ditemukan bahwa metode *Hyperband* dalam proses pencarian nilai-nilai *hyperparameter* yang optimal memerlukan waktu sekitar 3 jam, dengan jumlah iterasi yang dilakukan sebanyak 30, Berbeda dengan metode *grid search* yang digunakan oleh Elmadjian *et al.* yang menggunakan hanya subset dari dataset GazeCom, pencarian nilai-nilai *hyperparameter* dengan *Hyperband* dilakukan dengan menggunakan keseluruhan dari dataset, yang menandakan bahwa selain bersifat ramah daya komputasi, metode *Hyperband* juga lebih cepat dibandingkan dengan *grid search*. *Grid search* pada implementasinya dapat menghabiskan waktu yang sangat lama dikarenakan proses pencarian yang dilakukan pada seluruh kombinasi nilai-nilai yang ada pada *hyperparameter space*. Sebagai acuan, dengan menggunakan rentang nilai *hyperparameter* yang sama dengan rentang nilai yang digunakan pada penelitian ini, metode *grid search* akan melakukan iterasi sebanyak 6480 kali, dengan masing-masing iterasi dilakukan dengan jumlah *epoch* yang sama, yakni 20.

Investigasi yang dilakukan pada masing-masing *hyperparameter* juga menun-

jukkan bahwa *hyperparameter batch size* sangat mempengaruhi performa dari model, mulai dari waktu *training* dalam satu *epoch*, hingga adanya potensi *overfitting* yang terjadi. Sedangkan dari *hyperparameter* yang lain dapat dikatakan bahwa beberapa *hyperparameter* berpengaruh baik terhadap kompleksitas model maupun pada hasil evaluasi. Namun dikarenakan keseluruhan *hyperparameter* diuji dengan menggunakan set *hyperparameter* yang sama (sebagai contoh, pada proses analisis perbandingan nilai *hyperparameter dilation rate*, maka nilai-nilai *hyperparameter* yang lain akan diatur dengan menggunakan nilai yang digunakan pada model *untuned*). Hal ini mengindikasikan penelitian lebih lanjut agar dapat berfokus pada analisis kaitan antara dua atau lebih *hyperparameter* pada model TCN ini.

Penelitian ini memiliki beberapa kelebihan, seperti penggunaan metode *hyperband* yang lebih optimal dibandingkan dengan *grid search* pada proses pencarian *hyperparameter*. Selain itu, model yang parameternya didapatkan dari proses pencarian dengan metode *hyperband* ini dapat menghasilkan hasil evaluasi yang lebih baik dibandingkan dengan model sebelumnya, baik dengan menggunakan teknik evaluasi LOVO maupun *K-Fold Cross Validation*. Sedangkan kelemahan yang terdapat pada penelitian ini seperti model TCN baik yang dirancang oleh Elmadjian *et al* maupun hasil penelitian ini yang masih bersifat *offline* dengan sistem non-kausal. Selain itu model TCN yang dirancang masih menggunakan tingkat abstraksi yang tinggi, sehingga belum dianalisis lebih lanjut kerangka dari TCN secara mendalam.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil penelitian yang dilakukan, dapat diambil kesimpulan sebagai berikut.

- Model TCN hasil *hyperparameter tuning* dapat menghasilkan model dengan peningkatan skor evaluasi dengan menggunakan metode LOVO, dengan peningkatan yang lebih tinggi dengan menggunakan metode *K-Fold Cross Validation*. Metode TCN terbaru dapat memberikan peningkatan sebesar 1% pada kelas *saccade* dengan menggunakan metode evaluasi LOVO, dan dapat memberikan peningkatan sebesar 0,9%, 1,3%, 6,6%, dan 3,2% pada kelas *fixation*, *saccade*, *smooth pursuit*, serta *noise* dengan menggunakan metode evaluasi *K-Fold Cross Validation*.
- *Hyperparameter batch_size* memiliki pengaruh yang sangat besar pada performa model TCN, di mana parameter ini dapat mempengaruhi waktu dari proses *training* serta ukuran *fitness* dari sebuah model, baik apakah model *overfitting* atau tidak.

5.2 Saran

Saran untuk penelitian selanjutnya adalah sebagai berikut.

- Proses *training* pada model hasil *tuning* dapat dilakukan selama beberapa kali untuk memaksimalkan hasil evaluasi yang didapat. Selain itu, pengaruh proses *output filter* pada model dapat diinvestigasi lebih lanjut.
- Analisis *hyperparameter* pada model TCN dapat dilakukan secara lebih mendalam, yakni dengan melihat korelasi antar dua atau lebih *hyperparameter* serta pengaruhnya pada performa dari model itu sendiri.
- Model arsitektur TCN yang dibangun dengan menggunakan pustaka Keras-TCN memiliki tingkat abstraksi yang sangat tinggi. Pengembangan selanjutnya dapat berfokus pada konfigurasi arsitektur lebih lanjut maupun penambahan arsitektur lain pada TCN.

DAFTAR PUSTAKA

- [1] K. Holmqvist and R. Andersson, *Eye tracking: a comprehensive guide to methods, paradigms, and measures*, 2nd ed. Lund, Sweden: Lund Eye-Tracking Research Institute, 2017.
- [2] M. Dorr, T. Martinetz, K. R. Gegenfurtner, and E. Barth, “Variability of eye movements when viewing dynamic natural scenes,” *Journal of Vision*, vol. 10, no. 10, pp. 28–28, Aug. 2010. [Online]. Available: <http://jov.arvojournals.org/Article.aspx?doi=10.1167/10.10.28>
- [3] O. V. Komogortsev and A. Karpov, “Automated classification and scoring of smooth pursuit eye movements in the presence of fixations and saccades,” *Behavior Research Methods*, vol. 45, no. 1, pp. 203–215, Mar. 2013. [Online]. Available: <http://link.springer.com/10.3758/s13428-012-0234-9>
- [4] N. M. Arar, “Robust Eye Tracking Based on Adaptive Fusion of Multiple Cameras,” PhD Thesis, Sep. 2017.
- [5] A. W. Kiwelekar, G. S. Mahamunkar, L. D. Netak, and V. B. Nikam, “Deep Learning Techniques for Geospatial Data Analysis,” in *Machine Learning Paradigms*, G. A. Tsihrintzis and L. C. Jain, Eds. Cham: Springer International Publishing, 2020, vol. 18, pp. 63–81, series Title: Learning and Analytics in Intelligent Systems. [Online]. Available: http://link.springer.com/10.1007/978-3-030-49724-8_3
- [6] R. C. Staudemeyer and E. R. Morris, “Understanding LSTM – a tutorial into Long Short-Term Memory Recurrent Neural Networks,” 2019, publisher: arXiv Version Number: 1. [Online]. Available: <https://arxiv.org/abs/1909.09586>
- [7] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” 2014, publisher: arXiv Version Number: 9. [Online]. Available: <https://arxiv.org/abs/1412.6980>
- [8] A. Hidaka and T. Kurita, “Consecutive Dimensionality Reduction by Canonical Correlation Analysis for Visualization of Convolutional Neural Networks,” in *Proceedings of the ISCIE International Symposium on Stochastic Systems Theory and its Applications*, vol. 2017, Dec. 2017, pp. 160–167.
- [9] J. Yan, L. Mu, L. Wang, R. Ranjan, and A. Y. Zomaya, “Temporal Convolutional Networks for the Advance Prediction of ENSO,” *Scientific Reports*, vol. 10, no. 1, p. 8055, May 2020. [Online]. Available: <https://www.nature.com/articles/s41598-020-65070-5>
- [10] Z. Shang, B. Zhang, W. Li, S. Qian, and J. Zhang, “Machine remaining life prediction based on multi-layer self-attention and temporal convolution network,” *Complex & Intelligent Systems*, vol. 8, Dec. 2021.
- [11] J. Ashfaq Minstp Maat and A. Iqbal, “Introduction to Support Vector Machines and Kernel Methods,” Apr. 2019.

- [12] B. W. Matthews, “Comparison of the predicted and observed secondary structure of T4 phage lysozyme,” *Biochimica et Biophysica Acta (BBA)-Protein Structure*, vol. 405, no. 2, pp. 442–451, 1975, publisher: Elsevier.
- [13] B. Alojaiman, “Technological Modernizations in the Industry 5.0 Era: A Descriptive Analysis and Future Research Directions,” *Processes*, vol. 11, no. 5, p. 1318, Apr. 2023. [Online]. Available: <https://www.mdpi.com/2227-9717/11/5/1318>
- [14] X. Xu, Y. Lu, B. Vogel-Heuser, and L. Wang, “Industry 4.0 and Industry 5.0—Inception, conception and perception,” *Journal of Manufacturing Systems*, vol. 61, pp. 530–535, Oct. 2021. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S0278612521002119>
- [15] H. O. Edughele, Y. Zhang, F. Muhammad-Sukki, Q.-T. Vien, H. Morris-Cafiero, and M. Opoku Agyeman, “Eye-Tracking Assistive Technologies for Individuals With Amyotrophic Lateral Sclerosis,” *IEEE Access*, vol. 10, pp. 41 952–41 972, 2022. [Online]. Available: <https://ieeexplore.ieee.org/document/9745906/>
- [16] C. Ware and H. H. Mikaelian, “An evaluation of an eye tracker as a device for computer input2,” *ACM SIGCHI Bulletin*, vol. 17, no. SI, pp. 183–188, May 1986. [Online]. Available: <https://dl.acm.org/doi/10.1145/30851.275627>
- [17] L. E. Sibert and R. J. K. Jacob, “Evaluation of eye gaze interaction,” in *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. The Hague The Netherlands: ACM, Apr. 2000, pp. 281–288. [Online]. Available: <https://dl.acm.org/doi/10.1145/332040.332445>
- [18] D. H. Koh, S. Munikrishne Gowda, and O. V. Komogortsev, “Real time eye movement identification protocol,” in *CHI '10 Extended Abstracts on Human Factors in Computing Systems*. Atlanta Georgia USA: ACM, Apr. 2010, pp. 3499–3504. [Online]. Available: <https://dl.acm.org/doi/10.1145/1753846.1754008>
- [19] M. Vidal, A. Bulling, and H. Gellersen, “Detection of smooth pursuits using eye movement shape features,” in *Proceedings of the Symposium on Eye Tracking Research and Applications*. Santa Barbara California: ACM, Mar. 2012, pp. 177–180. [Online]. Available: <https://dl.acm.org/doi/10.1145/2168556.2168586>
- [20] R. Zemblys, D. C. Niehorster, O. Komogortsev, and K. Holmqvist, “Using machine learning to detect events in eye-tracking data,” *Behavior Research Methods*, vol. 50, no. 1, pp. 160–181, Feb. 2018. [Online]. Available: <http://link.springer.com/10.3758/s13428-017-0860-3>
- [21] S. Hoppe and A. Bulling, “End-to-End Eye Movement Detection Using Convolutional Neural Networks,” 2016, publisher: arXiv Version Number: 1. [Online]. Available: <https://arxiv.org/abs/1609.02452>
- [22] M. Startsev, I. Agtzidis, and M. Dorr, “1D CNN with BLSTM for automated classification of fixations, saccades, and smooth pursuits,” *Behavior Research Methods*, vol. 51, no. 2, pp. 556–572, Apr. 2019. [Online]. Available: <http://link.springer.com/10.3758/s13428-018-1144-2>

- [23] C. Elmadjian, C. Gonzales, and C. H. Morimoto, “Eye Movement Classification with Temporal Convolutional Networks,” in *Pattern Recognition. ICPR International Workshops and Challenges*, A. Del Bimbo, R. Cucchiara, S. Sclaroff, G. M. Farinella, T. Mei, M. Bertini, H. J. Escalante, and R. Vezzani, Eds. Cham: Springer International Publishing, 2021, vol. 12663, pp. 390–404, series Title: Lecture Notes in Computer Science. [Online]. Available: http://link.springer.com/10.1007/978-3-030-68796-0_28
- [24] T. Yu and H. Zhu, “Hyper-Parameter Optimization: A Review of Algorithms and Applications,” 2020, publisher: arXiv Version Number: 1. [Online]. Available: <https://arxiv.org/abs/2003.05689>
- [25] G. Montavon, G. Orr, and K.-R. Müller, Eds., *Neural networks: tricks of the trade*, 2nd ed., ser. Lecture notes in computer science. Heidelberg: Springer, 2012, no. 7700, oCLC: ocn828098376.
- [26] K. Eggenberger, “Towards an Empirical Foundation for Assessing Bayesian Optimization of Hyperparameters,” 2013.
- [27] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, “Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization,” Jun. 2018, arXiv:1603.06560 [cs, stat]. [Online]. Available: <http://arxiv.org/abs/1603.06560>
- [28] R. J. Leigh and D. S. Zee, *The Neurology of Eye Movements*, 5th ed. Oxford University Press, Jun. 2015. [Online]. Available: <https://academic.oup.com/book/25280>
- [29] T. Santini, W. Fuhl, T. Kübler, and E. Kasneci, “Bayesian identification of fixations, saccades, and smooth pursuits,” in *Proceedings of the Ninth Biennial ACM Symposium on Eye Tracking Research & Applications*. Charleston South Carolina: ACM, Mar. 2016, pp. 163–170. [Online]. Available: <https://dl.acm.org/doi/10.1145/2857491.2857512>
- [30] A. T. Bahill, A. Brockenbrough, and B. T. Troost, “Variability and development of a normative data base for saccadic eye movements,” *Investigative Ophthalmology & Visual Science*, vol. 21, no. 1 Pt 1, pp. 116–125, Jul. 1981.
- [31] D. D. Salvucci and J. H. Goldberg, “Identifying fixations and saccades in eye-tracking protocols,” in *Proceedings of the symposium on Eye tracking research & applications - ETRA '00*. Palm Beach Gardens, Florida, United States: ACM Press, 2000, pp. 71–78. [Online]. Available: <http://portal.acm.org/citation.cfm?doid=355017.355028>
- [32] S. Wibirama, S. Murnani, N. A. Setiawan, and H. Nurlatifa, “A Survey of Event Detection Methods for Eye Movements Classification in Smooth-Pursuit-Based Interactive Applications,” in *2020 International Symposium on Community-centric Systems (Ccs)*. Hachioji, Tokyo, Japan: IEEE, Sep. 2020, pp. 1–6. [Online]. Available: <https://ieeexplore.ieee.org/document/9231384/>
- [33] J. Pekkanen and O. Lappi, “A new and general approach to signal denoising and eye movement classification based on segmented linear regression,”

- Scientific Reports*, vol. 7, no. 1, p. 17726, Dec. 2017. [Online]. Available: <https://www.nature.com/articles/s41598-017-17983-x>
- [34] J. M. Bernardo and A. F. M. Smith, *Bayesian theory*, ser. Wiley series in probability and mathematical statistics. Chichester, Eng. ; New York: Wiley, 1994.
- [35] R. Zemblys, “Eye-movement event detection meets machine learning,” *BIOMEDICAL ENGINEERING 2016*, vol. 20, no. 1, 2017.
- [36] S. Bai, J. Z. Kolter, and V. Koltun, “An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling,” 2018, publisher: arXiv Version Number: 2. [Online]. Available: <https://arxiv.org/abs/1803.01271>
- [37] A. Darwish, A. E. Hassanien, and S. Das, “A survey of swarm and evolutionary computing approaches for deep learning,” *Artificial Intelligence Review*, vol. 53, no. 3, pp. 1767–1812, Mar. 2020. [Online]. Available: <http://link.springer.com/10.1007/s10462-019-09719-2>
- [38] J. A. Vilensky, W. Robertson, and C. A. Suárez-Quian, Eds., *The clinical anatomy of the cranial nerves: the nerves of an old Olympus towering top*. Ames, Iowa: Wiley Blackwell, 2015.
- [39] J. N. Stember, H. Celik, D. Gutman, N. Swinburne, R. Young, S. Eskreis-Winkler, A. Holodny, S. Jambawalikar, B. J. Wood, P. D. Chang, E. Krupinski, and U. Bagci, “Integrating Eye Tracking and Speech Recognition Accurately Annotates MR Brain Images for Deep Learning: Proof of Principle,” *Radiology: Artificial Intelligence*, vol. 3, no. 1, p. e200047, Jan. 2021. [Online]. Available: <http://pubs.rsna.org/doi/10.1148/ryai.2020200047>
- [40] P. A. Punde, M. E. Jadhav, and R. R. Manza, “A study of eye tracking technology and its applications,” in *2017 1st International Conference on Intelligent Systems and Information Management (ICISIM)*. Aurangabad: IEEE, Oct. 2017, pp. 86–90. [Online]. Available: <http://ieeexplore.ieee.org/document/8122153/>
- [41] J. M. Bland and D. G. Altman, “Statistics notes: Measurement error,” *BMJ*, vol. 312, no. 7047, pp. 1654–1654, Jun. 1996. [Online]. Available: <https://www.bmj.com/lookup/doi/10.1136/bmj.312.7047.1654>
- [42] T. F. Budinger and A. Brahme, Eds., *Comprehensive biomedical physics. Volume 1, Nuclear medicine and molecular imaging*. Amsterdam: Elsevier, 2014, oCLC: 886540666.
- [43] A. Graves, G. Wayne, and I. Danihelka, “Neural Turing Machines,” 2014, publisher: arXiv Version Number: 2. [Online]. Available: <https://arxiv.org/abs/1410.5401>
- [44] T. Hastie, R. Tibshirani, and J. H. Friedman, *The elements of statistical learning: data mining, inference, and prediction: with 200 full-color illustrations*, ser. Springer series in statistics. New York: Springer, 2001.
- [45] S. Ruder, “An overview of gradient descent optimization algorithms,” Jun. 2017, arXiv:1609.04747 [cs]. [Online]. Available: <http://arxiv.org/abs/1609.04747>

- [46] O. H. Rodriguez and J. M. Lopez Fernandez, "A semiotic reflection on the didactics of the Chain rule," *The Mathematics Enthusiast*, vol. 7, no. 2-3, pp. 321–332, Jul. 2010. [Online]. Available: <https://scholarworks.umt.edu/tme/vol7/iss2/10>
- [47] K. Janocha and W. M. Czarnecki, "On Loss Functions for Deep Neural Networks in Classification," 2017, publisher: arXiv Version Number: 1. [Online]. Available: <https://arxiv.org/abs/1702.05659>
- [48] S. Jadon, "A survey of loss functions for semantic segmentation," in *2020 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*. Via del Mar, Chile: IEEE, Oct. 2020, pp. 1–7. [Online]. Available: <https://ieeexplore.ieee.org/document/9277638/>
- [49] J. Wang, D. R. Salem, and R. K. Sani, "Two new exopolysaccharides from a thermophilic bacterium *Geobacillus* sp. WSUCF1: Characterization and bioactivities," *New Biotechnology*, vol. 61, pp. 29–39, Mar. 2021. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S1871678420301928>
- [50] C. M. Bishop, *Pattern Recognition and Machine Learning*, softcover reprint of the original 1st edition 2006 (corrected at 8th printing 2009) ed., ser. Information Science and Statistics. New York, NY: Springer New York, 2016.
- [51] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, ser. Springer Series in Statistics. New York, NY: Springer New York, 2009. [Online]. Available: <http://link.springer.com/10.1007/978-0-387-84858-7>
- [52] Y. Tian, D. Su, S. Lauria, and X. Liu, "Recent advances on loss functions in deep learning for computer vision," *Neurocomputing*, vol. 497, pp. 129–158, Aug. 2022. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S0925231222005239>
- [53] U. R. Dr.A, "Binary cross entropy with deep learning technique for Image classification," *International Journal of Advanced Trends in Computer Science and Engineering*, vol. 9, no. 4, pp. 5393–5397, Aug. 2020. [Online]. Available: <http://www.warse.org/IJATCSE/static/pdf/file/ijatcse175942020.pdf>
- [54] L. Deng, J. Li, J.-T. Huang, K. Yao, D. Yu, F. Seide, M. Seltzer, G. Zweig, X. He, J. Williams, Y. Gong, and A. Acero, "Recent advances in deep learning for speech research at Microsoft," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. Vancouver, BC, Canada: IEEE, May 2013, pp. 8604–8608. [Online]. Available: <http://ieeexplore.ieee.org/document/6639345/>
- [55] K. Fukushima, "A neural network model for selective attention in visual pattern recognition," *Biological Cybernetics*, vol. 55, no. 1, pp. 5–15, Oct. 1986. [Online]. Available: <http://link.springer.com/10.1007/BF00363973>
- [56] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015. [Online]. Available: <https://www.nature.com/articles/nature14539>
- [57] E. Lewinson, *Python for finance cookbook: over 80 powerful recipes for effective financial data analysis*, second edition ed. Birmingham Mumbai: Packt Publishing, 2022.

- [58] R. M. Schmidt, “Recurrent Neural Networks (RNNs): A gentle Introduction and Overview,” 2019, publisher: arXiv Version Number: 1. [Online]. Available: <https://arxiv.org/abs/1912.05911>
- [59] Q. Qiu, Z. Xie, L. Wu, and W. Li, “DGeoSegmenter: A dictionary-based Chinese word segmenter for the geoscience domain,” *Computers & Geosciences*, vol. 121, pp. 1–11, Dec. 2018. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S0098300418300852>
- [60] C. Lea, M. D. Flynn, R. Vidal, A. Reiter, and G. D. Hager, “Temporal Convolutional Networks for Action Segmentation and Detection,” 2016, publisher: arXiv Version Number: 1. [Online]. Available: <https://arxiv.org/abs/1611.05267>
- [61] F. Yu and V. Koltun, “Multi-Scale Context Aggregation by Dilated Convolutions,” Apr. 2016, arXiv:1511.07122 [cs]. [Online]. Available: <http://arxiv.org/abs/1511.07122>
- [62] C. Sun, J. Sharma, and M. Maiti, “Investigating the Relationship Between Dropout Regularization and Model Complexity in Neural Networks,” Aug. 2021, arXiv:2108.06628 [cs]. [Online]. Available: <http://arxiv.org/abs/2108.06628>
- [63] S. Salman and X. Liu, “Overfitting Mechanism and Avoidance in Deep Neural Networks,” Jan. 2019, arXiv:1901.06566 [cs, stat]. [Online]. Available: <http://arxiv.org/abs/1901.06566>
- [64] J. Lederer, “Activation Functions in Artificial Neural Networks: A Systematic Overview,” 2021, publisher: arXiv Version Number: 1. [Online]. Available: <https://arxiv.org/abs/2101.09957>
- [65] S. R. Dubey, S. K. Singh, and B. B. Chaudhuri, “Activation Functions in Deep Learning: A Comprehensive Survey and Benchmark,” 2021, publisher: arXiv Version Number: 3. [Online]. Available: <https://arxiv.org/abs/2109.14545>
- [66] P. M. Radiuk, “Impact of Training Set Batch Size on the Performance of Convolutional Neural Networks for Diverse Datasets,” *Information Technology and Management Science*, vol. 20, no. 1, Jan. 2017. [Online]. Available: <https://itms-journals.rtu.lv/article/view/itms-2017-0003>
- [67] S. Ioffe and C. Szegedy, “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift,” 2015, publisher: arXiv Version Number: 3. [Online]. Available: <https://arxiv.org/abs/1502.03167>
- [68] S. Afaq and S. Rao, “Significance Of Epochs On Training A Neural Network,” *International Journal of Scientific & Technology Research*, vol. 9, pp. 485–488, 2020.
- [69] M.-A. Zöller and M. F. Huber, “Benchmark and Survey of Automated Machine Learning Frameworks,” 2019, publisher: arXiv Version Number: 5. [Online]. Available: <https://arxiv.org/abs/1904.12054>
- [70] R. Elshaw, M. Maher, and S. Sakr, “Automated Machine Learning: State-of-The-Art and Open Challenges,” 2019, publisher: arXiv Version Number: 2. [Online]. Available: <https://arxiv.org/abs/1906.02287>

- [71] N. DeCastro-García, L. Muñoz Castañeda, D. Escudero García, and M. V. Carriegos, “Effect of the Sampling of a Dataset in the Hyperparameter Optimization Phase over the Efficiency of a Machine Learning Algorithm,” *Complexity*, vol. 2019, pp. 1–16, Feb. 2019. [Online]. Available: <https://www.hindawi.com/journals/complexity/2019/6278908/>
- [72] S. S. Olof, “A Comparative Study of Black-box Optimization Algorithms for Tuning of Hyper-parameters in Deep Neural Networks,” 2018.
- [73] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, “Algorithms for Hyper-Parameter Optimization,” in *Advances in Neural Information Processing Systems*, J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Q. Weinberger, Eds., vol. 24. Curran Associates, Inc., 2011. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2011/file/86e8f7ab32cfd12577bc2619bc635690-Paper.pdf
- [74] J. Bergstra and Y. Bengio, “Random Search for Hyper-Parameter Optimization,” *J. Mach. Learn. Res.*, vol. 13, no. null, pp. 281–305, Feb. 2012, publisher: JMLR.org.
- [75] J. Wang, J. Xu, and X. Wang, “Combination of Hyperband and Bayesian Optimization for Hyperparameter Optimization in Deep Learning,” 2018, publisher: arXiv Version Number: 1. [Online]. Available: <https://arxiv.org/abs/1801.01596>
- [76] M. Sokolova, N. Japkowicz, and S. Szpakowicz, “Beyond Accuracy, F-Score and ROC: A Family of Discriminant Measures for Performance Evaluation,” in *AI 2006: Advances in Artificial Intelligence*, D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, G. Weikum, A. Sattar, and B.-h. Kang, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, vol. 4304, pp. 1015–1021, series Title: Lecture Notes in Computer Science. [Online]. Available: http://link.springer.com/10.1007/11941439_114
- [77] H. Dalianis, “Evaluation Metrics and Evaluation,” in *Clinical Text Mining*. Cham: Springer International Publishing, 2018, pp. 45–53. [Online]. Available: http://link.springer.com/10.1007/978-3-319-78503-5_6
- [78] S. Yadav and S. Shukla, “Analysis of k-Fold Cross-Validation over Hold-Out Validation on Colossal Datasets for Quality Classification,” in *2016 IEEE 6th International Conference on Advanced Computing (IACC)*. Bhimavaram, India: IEEE, Feb. 2016, pp. 78–83. [Online]. Available: <http://ieeexplore.ieee.org/document/7544814/>
- [79] R. Kohavi, “A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection,” vol. 14, Mar. 2001.
- [80] G. I. Webb, C. Sammut, C. Perlich, T. Horváth, S. Wrobel, K. B. Korb, W. S. Noble, C. Leslie, M. G. Lagoudakis, N. Quadrianto, W. L. Buntine, N. Quadrianto, W. L. Buntine, L. Getoor, G. Namata, L. Getoor, X. J. Han, Jiawei, J.-A. Ting, S. Vijayakumar, S. Schaal, and L. D. Raedt, “Leave-One-Out Cross-Validation,” in *Encyclopedia of Machine Learning*, C. Sammut and G. I. Webb, Eds. Boston, MA: Springer US, 2011, pp. 600–601. [Online]. Available: http://link.springer.com/10.1007/978-0-387-30164-8_469

- [81] T.-T. Wong, "Performance evaluation of classification algorithms by k-fold and leave-one-out cross validation," *Pattern Recognition*, vol. 48, no. 9, pp. 2839–2846, Sep. 2015. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S0031320315000989>