DETEKSI *SLEEP APNEA* BERDASARKAN SINYAL *ELECTROCARDIOGRAM* (ECG) DENGAN *ONE DIMENSIONAL CONVOLUTIONAL NEURAL NETWORK* (1DCNN)

DETECTION OF SLEEP APNEA BASED ON ELECTROCARDIOGRAM SIGNAL (ECG) USING ONE DIMENSIONAL CONVOLUTIONAL NEURAL NETWORK (1DCNN)

LAPORAN TUGAS AKHIR

Diajukan untuk memenuhi persyaratan Sarjana Strata Satu (S-1)
Program Studi Teknik Elektro
Fakultas Teknik
Universitas Kristen Maranatha

Disusun Oleh:

CHRISTIAN FELIX

1822041

PROGRAM STUDI TEKNIK ELEKTRO
FAKULTAS TEKNIK
UNIVERSITAS KRISTEN MARANATHA
BANDUNG
2022

LEMBAR PENGESAHAN

DETEKSI *SLEEP APNEA* BERDASARKAN *SINYAL ELECTROCARDIOGRAM* (ECG) DENGAN *ONE DIMENSIONAL CONVOLUTIONAL NEURAL NETWORK* (1DCNN)

DETECTION OF SLEEP ANEA BASED ON ELECTROCARDIOGRAM SIGNAL (ECG) USING ONE DIMENSIONAL CONVOLUTIONAL NEURAL NETWORK (1DCNN)

LAPORAN TUGAS AKHIR

Disusun Oleh:

CHRISTIAN FELIX

1822041

Laporan Tugas Akhir ini telah diterima dan disahkan
untuk memenuhi persyaratan Sarjana Strata Satu (S-1)
Program Studi Teknik Elektro
Fakultas Teknik
Universitas Kristen Maranatha

Bandung, Januari 2022

Disahkan oleh :	Disetujui oleh :
Ketua Program Studi Teknik Elektro	Pembimbing
	Novie Theresia Pasaribu 2022.01.24 21:55:27 +07'00'
Heri Andrianto, S.T., M.T.	Novie Theresia Br. Pasaribu S.T., M.T.
NIK : 220835	NIK : 220924

PERNYATAAN ORISINALITAS LAPORAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama	: Christian Felix
NRP	: 1822041
Fakultas / Program Studi	: Teknik / Teknik Elektro

menyatakan dengan sesungguhnya bahwa Tugas Akhir ini adalah hasil karya saya sendiri dan bukan duplikasi hasil karya orang lain.

Apabila kelak terbukti pernyataan ini tidak benar, maka saya bersedia menerima sanksi yang diberikan dengan segala konsekuensinya.

Demikian pernyataan saya.

Bandung, Januari 2022

Yang menyatakan,

Christian Felix

NRP : 1822041

PERNYATAAN PERSETUJUAN PUBLIKASI LAPORAN TUGAS AKHIR

Saya yang bertanda tangan di bawah ini:

Nama	: Christian Felix
NRP	: 1822041
Fakultas / Program Studi	: Teknik / Teknik Elektro

Dengan ini, saya menyatakan bahwa:

1. Demi perkembangan ilmu pengetahuan, saya menyetujui untuk memberikan kepada Universitas Kristen Maranatha Hak Bebas Royalti non eksklusif (*Non Exclusive Royalty-Free Right*) atas laporan Tugas Akhir saya yang berjudul "**Deteksi *Sleep Apnea* berdasarkan Sinyal *Electrocardiogram* (ECG) dengan *One Dimensional Neural Network* (1DCNN)**".

2. Universitas Kristen Maranatha Bandung berhak menyimpan, mengalihmediakan/mengalihformatkan, mengelola dalam bentuk pangkalan data (*database*), mendistribusikannya, serta menampilkannya dalam bentuk *softcopy* untuk kepentingan akademis tanpa perlu meminta izin dari saya selama tetap mencantumkan nama saya sebagai penulis/pencipta.

3. Saya bersedia untuk menanggung secara pribadi, tanpa melibatkan pihak Universitas Kristen Maranatha Bandung, segala bentuk tuntutan hukum yang timbul atas pelanggaran hak cipta dalam karya ilmiah saya ini.

Demikian pernyataan ini saya buat dengan sebenarnya dan untuk dapat dipergunakan sebagaimana mestinya.

Bandung, Januari 2022

Yang menyatakan,

Christian Felix
NRP : 1822041

KATA PENGANTAR

Puji dan syukur penulis panjatkan kepada Tuhan Yang Maha Kuasa, karena atas rahmat-Nya penulis dapat menyelesaikan laporan Tugas Akhir yang berjudul "**Deteksi *Sleep Apnea* berdasarkan Sinyal *Electrocardiogram* (ECG) dengan *One Dimensional Neural Network* (1DCNN)"** untuk memenuhi persyaratan kelulusan Program Studi Sarjana Strata Satu (S1) Program Studi Teknik Elektro, Fakultas Teknik, Universitas Kristen Maranatha, Bandung.

Telah terselesaikannya laporan Tugas Akhir ini tidak lepas dari bantuan, motivasi, bimbingan, dan doa dari semua pihak baik secara langsung maupun tidak langsung. Penulis ingin menyapaikan ucapan terimakasih kepada :

1. Ibu Novie Theresia Br. Pasaribu, S.T., M.T. selaku dosen pembimbing yang telah memberikan dukungan moral selama masa Tugas Akhir sehingga Penulis dapat menyelesaikannya dengan tepat waktu.
2. Keluarga yang tiada henti-hentinya memberikan dukungan baik moral dan materiil kepada Penulis, sehingga Penulis tetap dapat menyelesaikan Tugas Akhir dengan tepat waktu.
3. Keluarga rohani dari Army of God – Gereja Mawar Sharon Bandung, terutama keluarga Team Leader - West Youth yang selalu memberikan semangat dan dukungan moral kepada Penulis, sehingga Tugas Akhir dapat terselesaikan dengan baik.
4. Ketua Program Studi Teknik Elektro Universitas Kristen Maranatha beserta seluruh jajaran dosen pengajar yang telah mengajar dan mendidik penulis selama menempuh pendidikan S-1 Teknik Elektro.
5. Seluruh teman-teman Angkatan 2018, terutama delapan teman Angkatan 2018 yang juga menyelesaikan masa studinya pada waktu yang bersamaan dengan penulis yang selalu memberi dukungan sehingga dapat menyelesaikan Tugas Akhir.

Akhir kata, penyusun berharap Tugas Akhir ini dapat memberikan sumbangan nyata bagi kemajuan Teknik Elektro pada khususnya, dan bagi pihak yang memerlukan.

Bandung, 10 Januari 2022

Penyusun,

Christian Felix
NRP : 1822041

DETEKSI *SLEEP APNEA* BERDASARKAN SINYAL ELECTROCARDIOGRAM (ECG) DENGAN ONE DIMENSIONAL CONVOLUTIONAL NEURAL NETWORK (1DCNN)

Christian Felix

NRP : 1822041

e-mail : 1822041@eng.maranatha.edu

ABSTRAK

Sleep Apnea merupakan penyakit gangguan tidur yang ditandai dengan adanya halangan terhadap saluran pernapasan bagian atas sehingga udara sulit masuk ke paru-paru dan berakibat pada desaturasi oksigen. *Sleep Apnea* yang tidak diberikan tindakan medis lebih lanjut dapat meningkatkan resiko terjangkitnya penyakit, terutama pada bidang *caridiovascular*. Studi menunjukkan, pasien dengan *Sleep Apnea*, 2-4 kali lebih rentan terhadap aritmia, gagal jantung, serta penyakit jantung koroner, oleh karena itu dibutuhkan prediksi yang akurat untuk menentukan pasien mengalami *Sleep Apnea* atau tidak, sehingga mampu diberikan tindakan medis yang sesuai. Deteksi *Sleep Apnea* dari dataset *Physionet Apnea-ECG Database (PAED)* dilakukan berdasarkan sinyal *Electrocardiogram* menggunakan metoda *Deep Learning* tipe *supervised learning*, yaitu *One Dimensional Convoltuional Neural Network* (1DCNN). Data sinyal dari PAED didekomposisi menjadi beberapa komponen sinyal sederhana berupa Intrinsic Mode Function (IMF) dengan metode *Empirical Mode Decomposition* (EMD), selanjutnya akan dipilih IMF sebagai masukan model 1DCNN. Hasil deteksi terbaik diperoleh dengan masukan berupa IMF 1 dengan nilai akurasi sebesar 93.24%.

Kata kunci: Deteksi *Sleep Apnea , Deep Learning, One Dimensional Convolutional Neural Netowrk (1DCNN), Empirical Mode Decomposition (EMD)*

DETECTION OF SLEEP ANEA EVENTS BASED ON ELECTROCARDIOGRAM SIGNAL (ECG) USING ONE DIMENSIONAL CONVOLUTIONAL NEURAL NETWORK (1DCNN)

Christian Felix

NRP : 1822041

e-mail : 1822041@eng.maranatha.edu

ABSTRACT

Sleep Apnea is a sleeping disorder which caused by obstruction in the upper respiratory tract and as a result, oxygen could barely enter the lungs and thus causing oxygen desaturation. In some cases, individuals with Sleep Apnea whom have not received medical treatment may increase risk of cardiovascular diseases. Studies have shown, patients with Sleep Apnea, are 2-4 times more susceptible to arrythmia, heart failure and coronary heart, therefore an accurate prediction is needed to detect patients undergo Sleep Apnea, and as a result could receive proper medical treatments accordingly. Sleep Apnea Detection from Physionet Apnea-ECG Database (PAED) based on Electrocardiogram signal is achieved using Deep Learning method, supervised learning type, called One Dimensional Convolutional Neural Network (1DCNN). Signal datas from PAED are decomposed into several simple signal components, called Intrinsic Mode Function (IMF) using Empirical Mode Decomposition method (EMD), and therefore IMF will be chosen as 1DCNN model input. The best detection result is achieved using IMF 1 as an input with accuracy value of 93.24%.

***Keywords**: Sleep Apnea Detection , Deep Learning, One Dimensional Convolutional Neural Netowrk (1DCNN), Empirical Mode Decomposition (EMD)*

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

BAB I
PENDAHULUAN

Pada bab ini dijelaskan mengenai latar belakang, rumusan masalah, tujuan, batasan masalah, dan sistematika penulisan.

I.1 Latar Belakang

Manusia menghabiskan sepertiga waktu setiap harinya untuk tidur, sehingga tidur adalah kegiatan kesehatan yang penting, mampu menghilangkan kelelahan, mengembalikan energi dan meningkatkan imunitas serta menunda penuaan[1]. Menurut survei yang dilakukan oleh *World Health Organization*, sebesar 27% manusia di seluruh dunia memiliki kualitas tidur yang buruk yang ditandai dengan mendengkur, sering terbangun di malam hari, dan kantuk di siang hari yang mampu menyebabkan gangguan *cardiovascular*[1]. Salah satu gangguan tidur yang umum terjadi dikenal sebagai *Sleep Apnea*. *Sleep Apnea* merupakan gangguan tidur yang ditandai dengan seringnya terjadi halangan terhadap saluran pernapasan bagian atas sehingga udara sulit masuk ke paru-paru yang berakibat pada desaturasi oksigen[2].

Pada umumnya, penyakit *Sleep Apnea* diderita oleh lansia akibat berkurangnya tingkat kontraksi otot polos pada saluran pernapasan bagian atas. Namun, menurut Dr. Mark S. Gaylord, *University of Tennessee Medical Center*, sekitar 80% bayi dengan berat di bawah dua kilogram juga dapat menderita *Sleep Apnea*, yang ditandai dengan bayi tidak bernapas selama 15 detik atau lebih dan terjadi sebanyak 5-10 kali dalam satu jam[3]. *Sleep Apnea* (keadaan penderita tidak bernafas dalam interval waktu tertentu) yang terburuk, dapat terjadi hingga lebih dari sekali dalam jangka waktu dua menit[4]. *Sleep Apnea* yang tidak diberi tindakan medis mampu meningkatkan resiko gangguan *cardiovascular* dan aritmia, studi menunjukkan, pasien dengan *Sleep Apnea*, 2-4 kali lebih rentan terhadap aritmia serta meningkatkan resiko gagal jantung sebesar 140% dan penyakit jantung koroner sebesar 30%[5]. Hal ini menunjukkan bahwa *Sleep Apnea* tidak mengenal usia, serta mampu meningkatkan kemungkinan terjangkitnya gangguan

cardiovascular lebih lanjut, sehingga salah satu cara mendeteksi *Sleep Apnea* adalah dengan menganalisa sinyal *Electrocardigram* (ECG *Signal*).

Sinyal ECG menjelaskan aktivitas jantung dalam bentuk impuls yang memiliki karakteristik, amplituda kecil, sebesar 0.5mV sampai 300mV, dan memiliki rentang frekuensi dari 0.05 – 100 Hz[6]. Setiap sinyal ECG terdiri atas kumpulan gelombang (P, Q, R, S, T, U) dan beberapa interval diantaranya (S-T, Q-T, P-R, R-R), interval tersebut digunakan untuk menghitung periode dan amplituda sebagai dasar untuk melakukan klasifikasi aktivitas jantung oleh para ahli[6]. Praktiknya di lapangan, rumah sakit menggunakan uji *polysomnography* (PSG) dalam mendiagnosa gangguan tidur, termasuk *Sleep Apnea*[7]. Namun di sisi lain, PSG memiliki beberapa kelemahan antara lain: (i) dibutuhkan pengawasan langsung selama pemeriksaan karena perlu mengenakan berbagai perangkat (contoh, sensor), (ii) PSG membutuhkan peralatan sistem perekam medis canggih, dan (iii) biaya PSG berkisar antara 3000-6000 USD, berawal dari kelemahan tersebut, maka dipilih sinyal ECG yang lebih hemat biaya sebagai salah satu cara untuk mendeteksi penyakit *Sleep Apnea*[7]. Melalui aktivitas jantung yang digambarkan oleh sinyal ECG, mampu dilakukan analisa untuk mendeteksi penyakit *Sleep Apnea.*

Masukan yang berupa sinyal ECG rentan terhadap derau (*noise*), diantaranya, *Powerline Interference Noise* dan *Baseline Wander Noise*[8]. *Noise* tersebut timbul pada rentang frekuensi dibawah 0.5 Hz dan di atas 150 Hz sehingga membutuhkan tahap *signal preprocessing* sehingga hasil *training* dapat lebih akurat[6]. Upaya yang dapat dilakukan untuk mengurangi *noise* adalah dengan tahap *signal preprocessing* yang meliputi penggunaan *filter*, untuk mengurangi Powerline Interference Noise dapat digunakan *Low Pass Filter* dengan frekuensi *cut-off* (f_c=50Hz)[6], sedangkan untuk mengurangi *Baseline Wander Noise* digunakan *IIR Notch* Filter dengan frekuensi *cut-off* (f_c=0.5Hz)[8]. Penggunaan *Low Pass Filter* serta *IIR Notch Filter* dalam mengurangi *noise* pada sinyal ECG sebagai tahap *signal preprocessing* telah dilakukan oleh Sheta, dkk. dalam melakukan diagnosa penyakita *Sleep Apnea* dengan menggunakan metode *Machine Learning* berupa *K-Nearest Neighbor* (KNN) memperoleh hasil akurasi sebesar 76.5%[6]. Selain penggunaan *filter*, tahap *signal preprocessing* untuk sinyal ECG

dapat dilakukan dengan metode *Empirical Mode Decomposition* (EMD)[9]. Metode *Empirical Mode Decomposition* akan mendekomposisi sinyal ECG menjadi komponen osilasi sederhana, sehingga dapat mengurangi *noise* pada sinyal ECG[10]. Dengan menerapkan metode EMD pada tahap *signal preprocessing*, Corthout, dkk. berhasil memperoleh nilai akurasi sebesar 89.45% dalam mendiagnosa penyakit *Sleep Apnea* dengan menggunakan metode *Machine Learning* berupa *Linear Discriminant* (LD)[11].

Convolutional Neural Network (CNN) merupakan salah satu algoritma dari *Deep Learning*, CNN umumnya digunakan dalam mengatasi data *input* berupa citra, termasuk didalamnya klasifikasi citra, dan deteksi objek dalam citra [12]. Beberapa penelitian sebelumnya telah digunakan metode CNN dalam mendeteksi penyakit *Sleep Apnea* berdasarkan analisa sinyal ECG, Ajit,dkk. mencapai nilai akurasi sebesar 86.22%, Yunxiang,dkk. berhasil memperoleh nilai akurasi sebesar 88% dalam mengklasifikasikan jenis *Sleep Apnea* dengan menggunakan dataset Apnea-ECG Database yang dipublikasikan oleh PhysioNet pada tahun 2000[13], dan Hassan,dkk. juga menerapkan CNN namun sebelumnya sinyal ECG awal diproses terlebih dahulu dengan *Tunable-Q factor Wavelet Transform* dan berhasil memperoleh nilai akurasi sebesar 87.33% dengan dataset *Apnea-ECG Database*[14]. Dalam mendeteksi penyakit *Sleep Apnea* dengan analisa sinyal ECG, pada Tugas Akhir dilakukan dengan salah satu pengembangan dari metode *Convolutional Neural Network* (CNN), yaitu *One Dimensional Convolutional Neural Network* (1DCNN). Berdasarkan *input* yang digunakan, yakni sinyal ECG yang bersifat satu dimensi (*time series*), Hung-Yu Chang, dkk. telah menggunakan metode *One Dimensional Convolutional Neural Network* (1DCNN) dalam mendiagnosa penyakit *Sleep Apnea* dengan memperoleh nilai akurasi sebesar 87.9%[15].

Dengan demikian, pada Tugas Akhir akan dilakukan perancangan dan implementasi deteksi *Sleep Apnea* berdasarkan sinyal ECG dengan menggunakan 1DCNN.

I.2 Rumusan Masalah

Berdasarkan uraian diatas, rumusan masalah dalam Tugas Akhir adalah bagaimana hasil mendeteksi *Sleep Apnea*/Normal berdasarkan analisa sinyal ECG dengan menggunakan metode *One Dimensional Convolutional Neural Network* (1DCNN)?

I.3 Tujuan

Tujuan Tugas Akhir adalah mendeteksi *Sleep Apnea* berdasarkan hasil analisa sinyal ECG dengan menggunakan metode *One Dimensional Convolutional Neural Network* (1DCNN).

I.4 Batasan Masalah

Batasan masalah dalam Tugas Akhir adalah sebagai berikut :

1. Dataset yang digunakan adalah dataset Apnea-ECG Database yang dipublikasikan oleh PhysioNet pada tahun 2000, dalam rangkaian kompetisi CinC (Computer in Cardiac Challenge 2000), data disediakan oleh Dr. Thomas Penzel - Philips University, Germany. Dataset terdiri atas 35 rekaman untuk training dan *testing*, dengan tiap rekaman memiliki durasi yang bervariasi antara 401- 587 menit [16].
2. Metode Deep Learning yang digunakan merupakan One Dimensional Convolutional Neural Network (1DCNN) dengan model Hung-Yu Chang, dkk. dan pengubahan pada arsitektur tersebut [16].
3. Deteksi *Sleep Apnea* dilakukan dengan klasifikasi biner, kelas Apnea (A) dan Normal (N).
4. *Metrics* yang digunakan dalam mengukur performa model pada data *testing*, antara lain: skor F-1, akurasi, presisi, sensitivitas, dan spesifisitas.
5. Model *Deep Learning* dan proses *signal preprocessing* yang dilakukan dirancang dengan menggunakan *library* dari Sci-kit Learn, Py-EMD, dan Keras API dari Tensorflow.

I.5 Sistematika Penulisan

Dalam laporan Tugas Akhir dibagi menjadi lima bab utama, referensi dan lampiran sebagai pendukung laporan tugas akhir ini. Berikut pembahasan masing-masing bab sebagai berikut :

BAB I : PENDAHULUAN

Pada bab ini, akan dijelaskan mengenai latar belakang, rumusan masalah, tujuan, batasan masalah, dan sistematika penulisan.

BAB II : LANDASAN TEORI

Pada bab ini, akan dijelaskan mengenai teori-teori penunjang Tugas Akhir. Adapun teori-teori penunjang tersebut mengenai : *Sleep Apnea*, Sinyal *Electrocardiogram* (ECG), *Empirical Mode Decomposition* (EMD), *Apnea-ECG Database*, *Deep Learning*, *Convolutional Neural Network* (CNN), *One Dimensional Convolutional Neural Network* (1DCNN) dan *metrics.*

BAB III : PERANCANGAN SISTEM

Pada bab ini, akan dijelaskan proses perancangan arsitektur *One Dimensional Convolutional Neural Network* (1DCNN) untuk mendeteksi *Sleep Apnea*. Dalam Tugas Akhir ini, masukan berupa sinyal ECG dari *Apnea-ECG Database*. Masukan akan dilakukan tahap *signal preprocessing* sebelum dilatih pada tahap *training and testing*. Tahap *signal preprocessing* meliputi tahap normalisasi sinyal, tahap *signal filtering*, dan tahap *Empirical Mode Decomposition* (EMD). Tahap *training and testing* akan dirancang arsitektur 1DCNN dan variasinya.

BAB IV : DATA PENGAMATAN DAN ANALISIS

Pada bab ini, akan dijelaskan data pengamatan dan analisis terhadap seluruh kombinasi arsitektur 1DCNN dan *input* yang diberikan berupa IMF 1 dan IMF 2. Performa yang terbaik dinilai dari *metrics* akurasi dan skor F-1 yang dihasilkan dari tiap kombinasi yang ada. *Metrics* berupa sensitivitas dan spesifisitas

digunakan untuk membandingkan performa model dalam memprediksi Apnea/Normal.

BAB V : SIMPULAN DAN SARAN

Bab ini menjelaskan mengenai simpulan dan saran dari bab-bab yang telah dibahas sebelumnya.

BAB II
LANDASAN TEORI

Pada bab ini dijelaskan mengenai teori-teori penunjang Tugas Akhir. Adapun teori-teori penunjang tersebut mengenai : *Sleep Apnea*, Sinyal *Electrocardiogram* (ECG), *Empirical Mode Decomposition* (EMD), *Apnea-ECG Database*, *Deep Learning*, *Convolutional Neural Network* (CNN), *One Dimensional Convolutional Neural Network* (1DCNN) dan *metrics.*

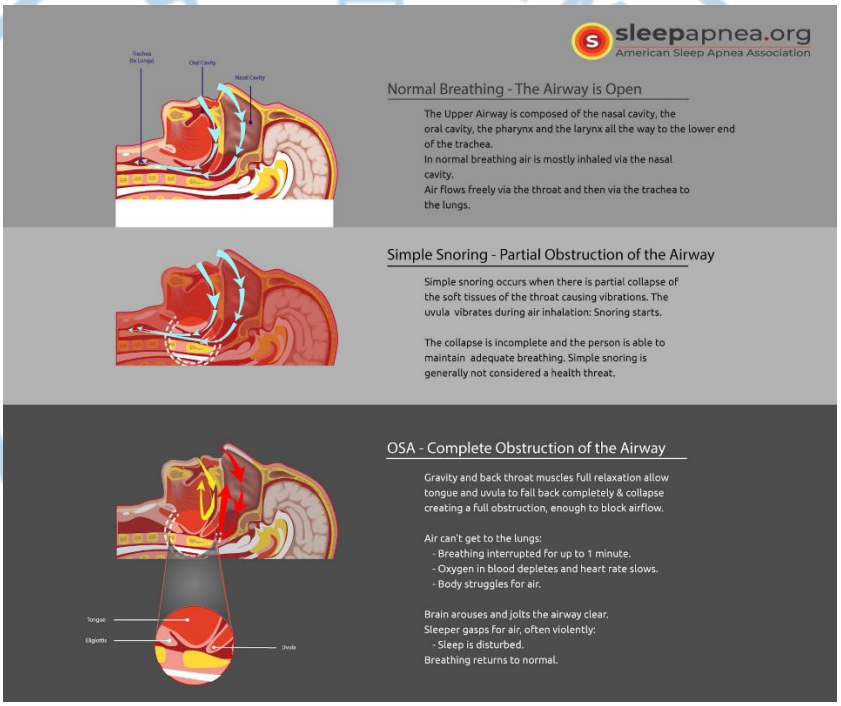
II.1 *Sleep Apnea*

Sleep Apnea merupakan gangguan tidur yang menyebabkan seseorang mengalami keadaan sesak napas hingga pada beberapa kasus khusus, seseorang berhenti bernapas untuk sementara waktu[3]. *Sleep Apnea* umum diderita oleh manusia paruh baya dan lanjut usia (manula) serta tingkat keterjangkitan *Sleep Apnea* terhadap populasi manusia di tiap negara berkisar antara 9% - 38%[17]. Penelitian oleh Frost dan Sullivan (2016) memperkirakan biaya perawatan medis per tahunnya untuk penderita *Sleep Apnea* yang tidak terdiagnosa di kalangan orang dewasa di Amerika Serikat mencapai 149.6 miliar USD. Namun, menurut penelitian yang telah dilakukan oleh Dr. Mark S. Gaylord, *the Director of Neonatology at the University of Tennessee Medical Center*, sekitar 80% bayi dengan berat di bawah dua kilogram juga mengalami Sleep Apnea selama 15 detik, sebanyak 5-10 kali dalam satu jam[3]. Oleh karena itu, *Sleep Apnea* tidak mengenal batasan usia, bahkan bayi juga dapat mengalami *Sleep Apnea*, terlebih lagi gangguan ini dapat berakibat fatal jika diderita bayi karena sistem pernapasannya yang belum berkembang sempurna sehingga mampu menyebabkan pola pernapasan yang tidak menentu[3].

Terdapat tiga jenis *Sleep Apnea* dan digolongkan berdasarkan penyebab terjadinya, antara lain: *Obstructive Sleep Apnea* (OSA), *Central Sleep Apnea* (CSA), dan *Mixed Sleep Apnea* (MSA). *Obstructive Sleep Apnea* disebabkan akibat

seringnya terjadi halangan terhadap saluran pernapasan bagian atas, sehingga mampu menimbulkan desaturasi oksigen, *Central Sleep Apnea* (CSA) terjadi ketika otak sesekali gagal memerintahkan otot-otot pernapasan pada bagian dada, dan *Mixed Sleep Apnea* (MSA) terjadi akibat kombinasi OSA dan CSA[2]. Dari ketiganya, OSA lebih umum terjadi dibanding tipe CSA dan MSA.

Pada kebanyakan kasus, penderita *Sleep Apnea* bahkan tidak menyadari bahwa tubuhnya mengalami *Sleep Apnea*, yaitu terjadinya penghentian pernapasan untuk beberapa waktu karena tersebut tidak menyebabkan tubuh penderita untuk bangun dari tidurnya[2]. Gambar II.1 menunjukan ilustrasi *Obstructive Sleep Apnea* dan perbandingannya dengan mendengkur (*simple snoring*), dan pernapasan normal.



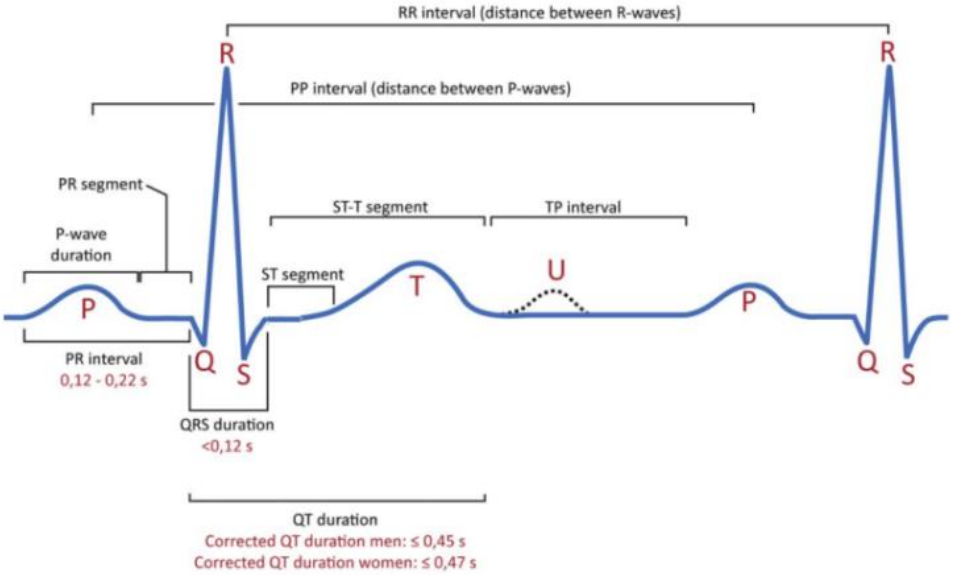
Gambar II.1 Ilustrasi *Obstructive Sleep Apnea* (OSA)
Sumber: American Sleep Apnea Association, URL: https://www.sleepapnea.org/learn/sleep-apnea/obstructive-sleep-apnea/

Dalam perkembangannya, terdapat beberapa metode yang dilakukan dalam mendiagnosa *Sleep Apnea*, antara lain dengan menggunakan metode uji *Polysomnography* (PSG) dan dengan menganalisa sinyal *Electrocardiogram* (ECG). Uji *Polysomnography* dinilai sebagai standar terbaik dalam menganalisa penyakit tidur, tidak terkecuali dalam mendiagnosa *Sleep Apnea*[2]. Uji

Polysomnography (PSG) dapat mendiagnosa *Sleep Apnea* melalui beberapa faktor, antara lain: aliran udara, bernapas, dengkuran, saturasi oksigen dalam darah (SpO2), *electrooculography* (EOG), *electroencephalography* (EEG), dan *electrocardiography* (ECG)[6]. Pada praktiknya, uji PSG dinilai kurang efektif, dikarenakan untuk mendapatkan hasil uji PSG, pasien harus tidur di laboratorium tidur khusus semalam dengan 22 kabel ditempel ke bagian tubuh untuk merekam jejak medis pada tubuh pasien selama tidur[2], selain itu, uji PSG juga sulit ditemukan pada rumah sakit di daerah-daerah kurang berkembang, sehingga sulit diakses semua golongan masyarakat. Dapat disimpulkan bahwa uji PSG memiliki kekurangan dalam mendiagnosa *Sleep Apnea*, (i) dibutuhkan pengawasan langsung selama pemeriksaan karena perlu mengenakan berbagai perangkat (contoh, sensor), (ii) PSG membutuhkan perlatan sistem perekam medis canggih, dan (iii) biaya PSG berkisar antara 3000-6000 USD[6]. Oleh karena beberapa alasan tersebut, dipilih metode kedua, yakni dengan menganalisa sinyal ECG dalam melakukan diagnosa *Sleep Apnea*.

II.2 Sinyal *Electrocardiogram* (ECG)

Sinyal *Electrocardiogram* (ECG) menggambarkan aktivitas jantung dalam bentuk impuls, setiap sinyal ECG terdiri atas kumpulan gelombang (P,Q,R,S,T,U) dan beberapa interval gelombang diantaranya (S-T, Q-T, P-R, R-R) yang juga digunakan untuk klasifikasi aktivitas jantung[6]. Gambar II.2 menjelaskan berbagai jenis gelombang dan interval gelombang pada sinyal ECG, dan pada Tabel II.1 dijelaskan berbagai jenis gelombang pada sinyal ECG dan fungsinya.



Gambar II.2 Struktur Sinyal ECG[6]

Tabel II.1 Struktur Sinyal ECG[6]

Fitur ECG	Deskripsi
Gelombang P	Kontraksi pulsa dari sistol atrium
Gelombang Q	Defleksi turunan langsung setelah gelombang P
Gelombang R	Kontraksi bilik (ventrikel)
Gelombang S	Defleksi turunan langsung setelah gelombang R
Gelombang T	Menggambarkan pemulihan ventrikel (*ventricular recovery*)
Gelombang U	Mengikuti setelah gelombang T (umumnya diabaikan)
Interval P-R	Waktu impuls listrik berangkat dari titik sinusoidal ke titik *Atrioventricular* (AV)
Interval R-R	Segmen diantara dua titik puncak R pertama dan selanjutnya
QRS *Complex*	Daerah yang melambangkan kontraksi ventrikel dan depolarisasi (disebut juga QRS *complex*)
Interval S-T	Segmen berupa isoelektrik dan dimulai setelah QRS *complex*
Interval Q-T	Jarak dari awal QRS *complex* sampai ke ujung akhir dari gelombang T

Pada Tabel II.2 dijelaskan karakteristik umum fitur dari sinyal ECG dilihat dari nilai amplituda sinyal (mV) dan periode setiap gelombang.

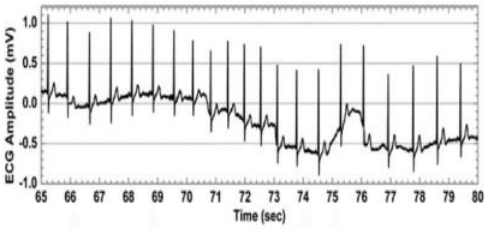
Tabel II.2 Karakteristik Fitur dari Sinyal ECG[6]

Fitur ECG	Periode (s)	Amplituda (mV)
Gelombang P	0.08 – 0.1	0.25
Gelombang T	0.16 – 0.2	>0
QRS *Complex*	0.08 – 0.1	Q < 0, R > 0, S < 0
Interval R-R	0.6 – 1.2	-
Interval P-R	0.12 – 0.22	R > 0
Interval S-T	0.2 – 0.32	Isoelektrik
Interval Q-T	0.35 – 0.45	-

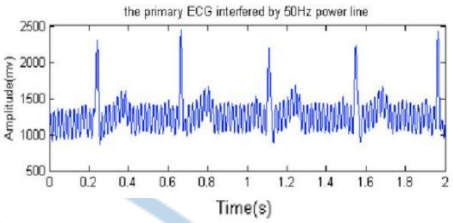
Mengacu pada nilai-nilai umum dari fitur sinyal ECG pada Tabel II.2, terdapat dua pendekatan yang dapat digunakan untuk melakukan diagnosa *Sleep Apnea*, antara lain, metode *Heart Rate Variability* (HRV) dan *ECG-Derived Respiration* (EDR)[18]. Metode *Heart Rate Variability* memanfaatkan interval R-R dari sinyal ECG, yang menandakan interval antar denyut jantung pada sinyal ECG dan secara fisiologis terbukti terkait dengan peristiwa *Sleep Apnea*. Ketika terjadi peristiwa *Sleep Apnea* dan pernapasan terhenti beberapa saat, aktivitas saraf vagus meningkat, menyebabkan denyut jantung melambat (*bradycardia*), diikuti dengan peningkatan nyata terhadap denyut jantung (*tachycardia*)[18]. Metode *ECG-Derived Respiration* (EDR) mampu mendiagnosa peristiwa *Sleep Apnea* melalui pergerakan posisi elektroda alat ECG yang ditempel di dada pasien. Selama siklus bernapas, elektroda tersebut akan bergerak akibat perubahan impedansi listrik di dada akibat paru-paru terisi udara dan membuang udara[18]. Kedua metode tersebut, HRV dan EDR membutuhkan algoritma pendeteksi QRS *complex* yang baik karena deteksi titik maksimum R salah ditempatkan maka akan sangat berpengaruh terhadap akurasi dari fitur sinyal ECG yang diekstrak[15].

Praktiknya, sinyal ECG hasil rekaman medis dari alat *Electrocardiograph* mengandung *noise,* diantaranya, *baseline wander* dan *powerline interference*. *Baseline wander* merupakan *noise* pada rentang frekuensi 0.5 Hz yang menyebabkan sinyal naik dan turun, sehingga menyebabkan sinyal bergeser dari sumbu x aslinya. Hal ini disebabkan oleh peletakan elektroda alat *Electrocardiograph* yang tidak sesuai, dan dapat disebabkan juga oleh pergerakan pasien selama rekam medis[8]. *Powerline Interference* merupakan *noise* yang sering muncul pada sinyal ECG, disebabkan oleh gelombang elektromagnetik yang dihasilkan dari listrik jala-jala. *Noise* ini muncul pada rentang frekuensi listirk jala-

jala, yakni pada rentang frekuensi 50-60 Hz[6], akibatnya muncul *noise* berupa interferensi sinusoidal beserta harmoniknya, sehingga bentuk gelombang pada amplituda rendah menjadi sulit dianalisa, contohnya, gelombang P dan gelombang T[8]. Gambar II.3 dan Gambar II.4 menggambarkan sinyal ECG yang mengandung *noise* berupa *baseline wander* dan *power line interference*.



Gambar II.3 Pengaruh *Baseline Wander Noise* pada Sinyal ECG[17]



Gambar II.4 Pengaruh *Powerline Interference Noise* pada Sinyal ECG [17]

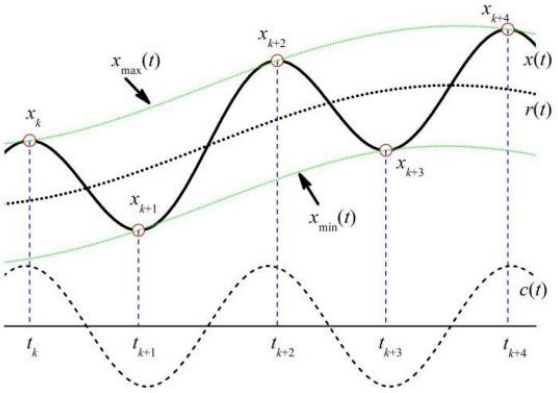
Oleh karena hal tersebut dibutuhkan *filter* dari sinyal ECG sehingga dihasilkan sinyal yang bebas *noise*, pada Tugas Akhir ini digunakan teknik *Empirical Mode Decomposition* (EMD) untuk mengurangi *noise* pada sinyal ECG.

II.3 *Empirical Mode Decomposition*

Empirical Mode Decomposition (EMD) merupakan teknik dekomposisi satu dimensi yang mengekstrak sinyal menjadi komponen osilasi sederhana berupa fungi AM-FM, yang disebut *Intrinsic Mode Function* melalui proses yang dikenal sebagai *sifting process*[19]. Hasil dari *sifting process* dapat disebut IMF jika memenuhi 2 kondisi:

1. Jumlah titik ekstrem harus sama atau berbeda dengan perbedaan maksimal 1.
2. Nilai rata-rata dari selubung yang dibentuk dari titik maksimum dan minimum harus nol.

Prinsip EMD yang digagas oleh Huang, memperlakukan sinyal sebagai "osilasi cepat yang ditumpangkan pada osilasi lambat", sehingga sinyal dapat didekomposisi menjadi beberapa komponen osilasi sederhana[10]. Proses EMD hingga dihasilkan komponen osilasi sederhana berupa IMF, antara lain sebagai berikut[10]:



Gambar II.5 Sinyal $x(t)$ yang sedang Dianalisa [10]

1. Merujuk pada Gambar II.5, asumsi sinyal $x(t)$ adalah sinyal yang akan dianalisa. Sinyal $x(t)$ terdiri atas komponen osilasi $c(t)$ dengan nilai rerata nol dan sebuah variabel residual (sinyal dasar/*baseline signal*) $r(t)$, sehingga $x(t)$ dapat didefinisikan sebagai berikut.

   $x(t) = c(t) + r(t)$ .....(II.1)

2. Karena $c(t)$ adalah fungsi osilasi dengan nilai rerata nol, nilai integral pada interval kedua titik ekstrim lokal t_k dan t_{k+2} total nilainya adalah nol, sehingga dari persamaan (II.1), didapat nilai sebagai berikut.

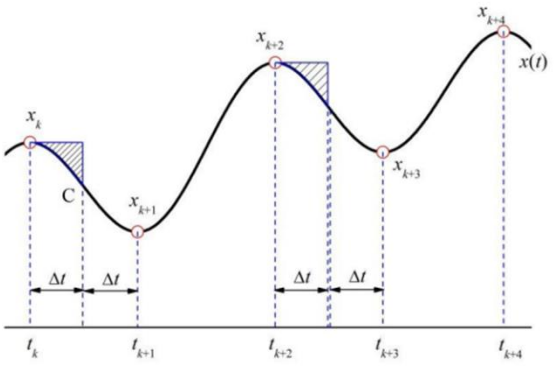
   $$\int_{t_k}^{t_{k+2}} x(t)\mathrm{d}t = \int_{t_k}^{t_{k+2}} c(t)\mathrm{d}t + \int_{t_k}^{t_{k+2}} r(t)\mathrm{d}t = \int_{t_k}^{t_{k+2}} r(t)\mathrm{d}t$$
   $$= r_o(t_{k+2}) - r_o(t_k) \text{.....(II.2)}$$

3. Menurut *Lagrange differential theorem of mean*, diketahui setidaknya ada satu titik t_l di antara titik t_k dan t_{k+2} , sehingga persamaaan II.2 dapat ditulis kembali menjadi,

   $\int_{t_k}^{t_{k+2}} x(t)\mathrm{d}t = (t_{k+2} - t_k)r(t_l)$.....(II.3)

   dengan cara yang sama, dapat diketahui juga terdapat sebuah titik t_m di antara titik t_{k+1} dan titik t_{k+3} sehingga didapat persamaan sebagai berikut.

   $\int_{t_{k+1}}^{t_{k+3}} x(t)\mathrm{d}t = (t_{k+3} - t_{k+1})r(t_m)$.....(II.4)



Gambar II.6 Proses Mencari Nilai Residual[10]

4. Merujuk pada Gambar II.6, dipilih sebuah titik C yang terletak pada titik tengah t_k hingga t_{k+1}, sehingga dapat membentuk hubungan antara persamaan II.3 dan II.4 membentuk persamaan II.5 dan II.6 sebagai berikut.

$$\int_{t_k}^{t_{k+2}} x(t)\mathrm{d}t = \int_{t_k}^{t_k+\Delta t} x(t)\mathrm{d}t + \int_{t_k+\Delta t}^{t_{k+2}+\Delta t} x(t)\mathrm{d}t - \int_{t_{k+2}}^{t_{k+2}+\Delta t} x(t)\mathrm{d}t$$

$$= (t_{k+2} - t_k)r(t_l) \text{.....(II.5)}$$

$$\int_{t_{k+1}}^{t_{k+3}} x(t)\mathrm{d}t = -\int_{t_{k+1}-\Delta t}^{t_{k+1}} x(t)\mathrm{d}t + \int_{t_{k+1}-\Delta t}^{t_{k+3}-\Delta t} x(t)\mathrm{d}t + \int_{t_{k+3}-\Delta t}^{t_{k+3}} x(t)\mathrm{d}t$$

$$= (t_{k+3} - t_{k+1})r(t_m) \text{.....(II.6)}$$

dengan $\Delta t = (t_{k+1} - t_k)/2$

5. Langkah pertama hingga keempat diilustrasikan dengan Gambar II.6. Sinyal $x(t)$ digambarkan dengan warna hitam, residu $r(t)$ dengan garis hitam putus-putus di tengah-tengah selubung $x_{max}(t)$ dan $x_{min}(t)$, serta komponen osilasi $c(t)$ digambarkan dengan garis putus-putus.
6. Dalam memperkenalkan nilai titik ekstrim pada keempat titik $t_k, t_{k+1}, t_{k+2}, t_{k+3}$, dapat digunakan hubungan pendekatan sebagai berikut.

$$\int_{t_k}^{t_k+\Delta t} x(t)\mathrm{d}t - \int_{t_{k+2}}^{t_{k+2}+\Delta t} x(t)\mathrm{d}t \approx -\Delta t(x_{k+2} - x_k) \text{ .....(II.7)}$$

$$-\int_{t_{k+1}-\Delta t}^{t_{k+1}} x(t)\mathrm{d}t + \int_{t_{k+3}-\Delta t}^{t_{k+3}} x(t)\mathrm{d}t \approx \Delta t(x_{k+3} - x_{k+1}) \text{.....(II.8)}$$

$\frac{1}{t_{k+2}-t_k}\int_{t_k+\Delta t}^{t_{k+2}+\Delta t} x(t)\mathrm{d}t \approx \frac{1}{t_{k+3}-t_{k+1}}\int_{t_{k+1}-\Delta t}^{t_{k+3}-\Delta t} x(t)\mathrm{d}t$……..…….(II.9)

$t_m - t_l \approx 2\Delta t$……..……..…..……..…..……..……(II.10)

Dengan penyederhanaan, didapat persamaan II.11 sebagai berikut.

$\frac{1}{2\Delta t}[r(t_m) - r(t_l)] \approx \frac{1}{2}\left(\frac{x_{k+2}-x_k}{t_{k+2}-t_k} + \frac{x_{k+3}-x_{k+1}}{t_{k+3}-t_{k+1}}\right)$……(II.11)

7. Pada persamaan II.11, persamaan di sebelah kiri (LHS) merupakan nilai pendekatan dari turunan fungsi residu $r(t)$, suku pertama dan suku kedua pada persamaan di sebelah kanan (RHS), secara berurutan, menunjukkan nilai turunan dari kurva $x_{max}(t)$ di antara dua titik maksimum lokal, dan nilai turunan dari kurva $x_{min}(t)$ di antara dua titik minimum lokal. Merujuk pada karakteristik fungsi $r(t), x_{max}(t)$, dan $x_{min}(t)$ yang secara seragam bervariasi terhadap interval $[t_k, t_{k+3}]$, sehingga didapat persamaan II.12 sebagai berikut.

   $r'(t) \approx \frac{[x'_{max}(t)+x'_{min}(t)]}{2}$ $(t_l \le t \le t_m)$ ……..…..…..(II.12)

   maka didapat:

   $r(t) \approx \frac{[x_{max}(t)+x_{min}(t)]}{2}(t_l \le t \le t_m)$……..…..… (II.13)

   Merujuk pada Gambar II.6, nilai $r(t)$ merupakan nilai rerata antara selubung atas $x_{max}(t)$ dan selubung bawah $x_{min}(t)$.

8. Setelah mendapat persamaan fungsi ekstrim berturut-turut dari sinyal $x(t)$, titik maksimum dan minimum lokal dihubungkan satu dengan lain dengan metode *cubic spline interpolation*, titik maksimum lokal yang terhubung akan menjadi selubung atas, $x_{max}(t)$, dan titik minimum lokal yang terhubung akan menjadi selubung bawah, $x_{min}(t)$.

9. Kedua selubung digunakan untuk menghitung nilai rerata dari fungsi waktu $m_1(t)$.

10. Kandidat nilai IMF pertama didefinisikan sebagai $h_1(t)$, yang menggambarkan perbedaan nilai sinyal $x(t)$ dengan nilai rerata $m_1(t)$, secara matematis didefinisikan sebagai berikut.

    $h_1(t) = x(t) - m_1(t)$ ……..…….(II.14)

11. Nilai $h_1(t)$ akan didefinisikan sebagai nilai IMF pertama $H_1(t)$ apabila memenuhi kedua kriteria *sifting process* yang dijelaskan sebelumnya. Apabila tidak memenuhi kriteria tersebut, maka nilai $h_1(t)$ akan diperlakukan seperti sinyal baru, kemudian dengan cara yang sama, ditentukan selubung atas dan bawah, dan nilai rerata $m_2(t)$, sehingga dapat didefinisikan persamaan baru yakni, $h_2(t) = h_1(t) - m_2(t)$. Iterasi ini akan terus berlanjut hingga $h_k(t)$ memenuhi kriteria *sifting process* sehingga kemudian dapat didefinisikan sebagai IMF pertama, $H_1(t)$.
12. Nilai residu pertama dari hasil perhitungan IMF pertama, didefinisikan sebagai berikut.

    $d_1(t) = x(t) - H_1(t)$ ……(II.15)

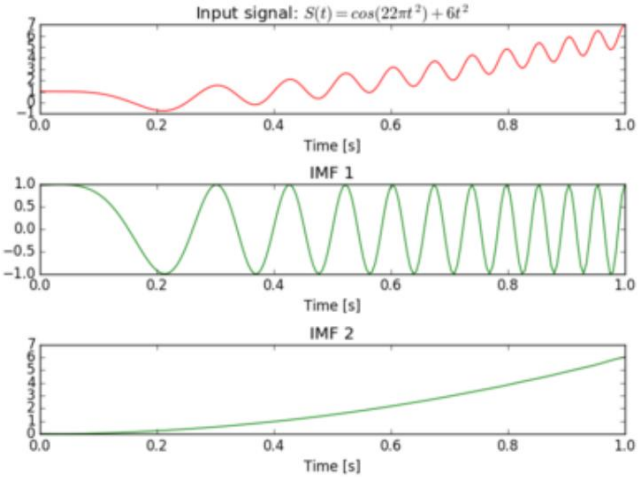
    Nilai residu $d_1(t)$ kemudian akan dianalisa kembali seperti pada langkah 8-11 untuk mendapatkan nilai IMF kedua, $H_2(t)$. Proses ini dikenal sebagai *sifting process* dan akan berlanjut hingga residu $d_k(t)$ tidak memberikan variasi signifikan dengan nilai residu sebelumnya, $d_{k-1}(t)$.

Adapun, kriteria *sifting process* dapat didefinisikan secara matematis sebagai berikut.

$$\sum_t \frac{(h_1(t) - x(t))^2}{x(t)^2} < \epsilon \quad \text{……(II.16)}$$

dengan nilai $\epsilon = 0.2$ sampai dengan 0.3.

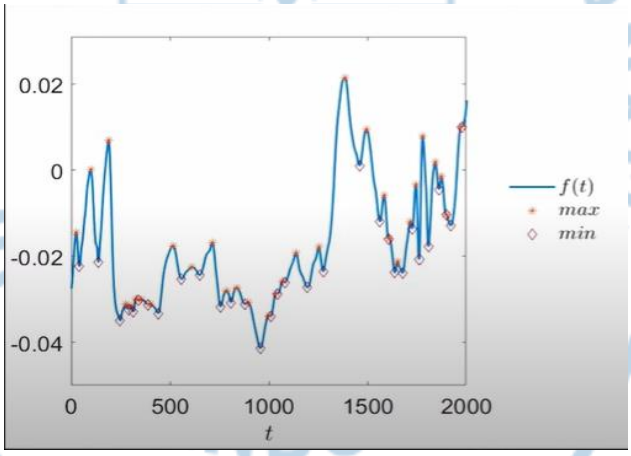
Berikut ini adalah gambar yang menjelaskan hasil EMD dari sinyal $S(t)$. Terlihat pada gambar, dari sinyal *input* $S(t) = \cos(22\pi t^2) + 6t^2$ didapatkan dua IMF, yakni IMF 1 dan IMF 2. Didapatkan hasil 2 IMF menandakan bahwa nilai residu $d_2(t)$ tidak memberikan variasi signifikan dengan nilai residu sebelumnya, $d_1(t)$, sehingga IMF 2 merupakan komponen osilasi paling sederhana dari sinyal $S(t)$.



Gambar II.7 Hasil EMD dari Sinyal $S(t)$ [20]

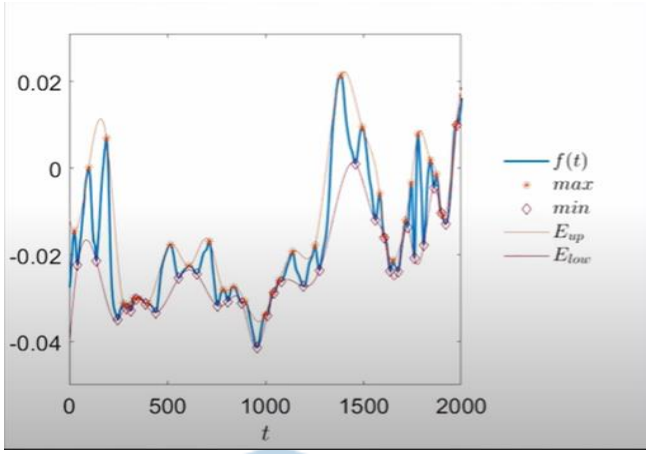
Adapun, visualisasi proses EMD dari kedua belas langkah yang telah dijelaskan sebelumnya disajikan melalui gambar berikut.

1. Mencari titik ekstrim lokal maksimum dan minimum.



Gambar II.8 Visualisasi Langkah EMD Pertama

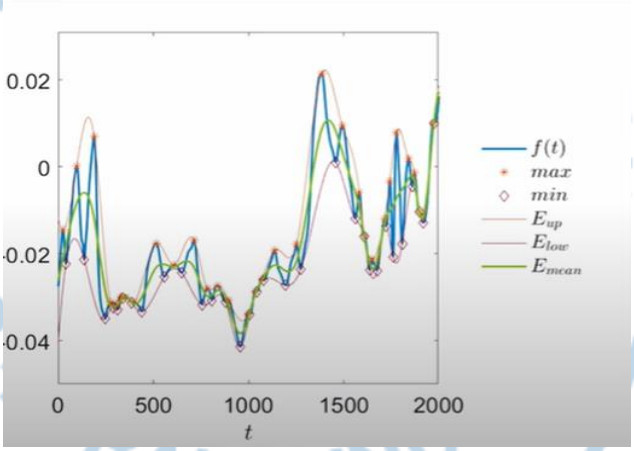
2. Membentuk selubung atas dari titik ekstrim maksimum $E_{up}(t)$ dan selubung bawah dari titik ekstrim minimum $E_{low}(t)$.



Gambar II.9 Visualisasi Langkah EMD Kedua

3. Menentukan nilai rerata dari selubung atas dan bawah berupa sinyal $E_{mean}(t)$ dari selubung atas dan bawah dengan persamaan berikut.

$$E_{mean}(t) = E_{up}(t) - E_{low}(t) \quad \text{(II.17)}$$

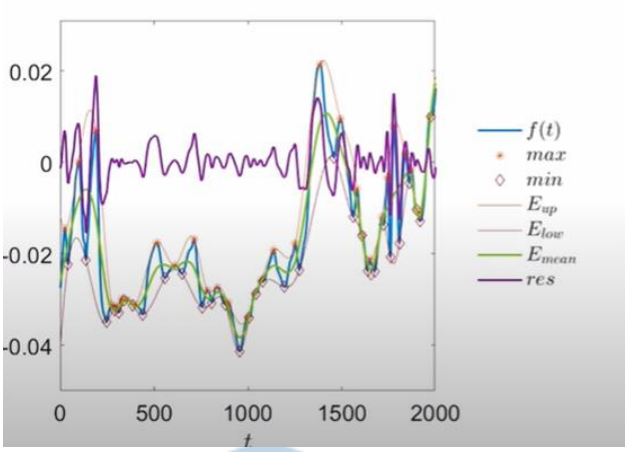


Gambar II.10 Visualisasi Langkah EMD Ketiga

4. Kandidat IMF berupa sinyal $res(t)$ didefinisikan pada persamaan II.18, kemudian dicek dengan *stopping criterion* seperti pada persamaan II.16.

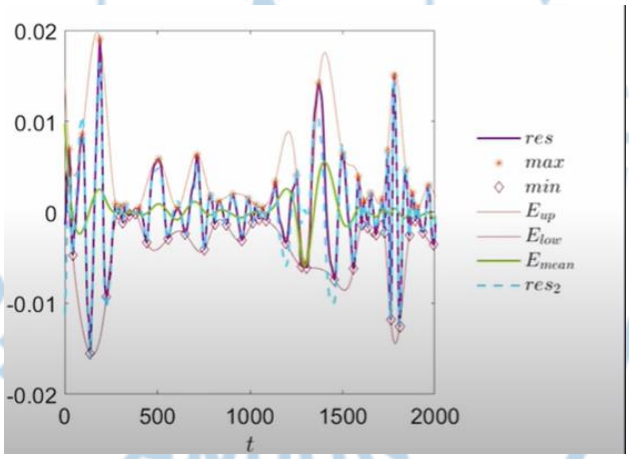
$$res(t) = f(t) - E_{mean}(t) \quad \text{(II.18)}$$

Pada contoh visualisasi ini, $res(t)$ tidak memenuhi *stopping criterion*.



Gambar II.11 Visualisasi Langkah EMD Keempat

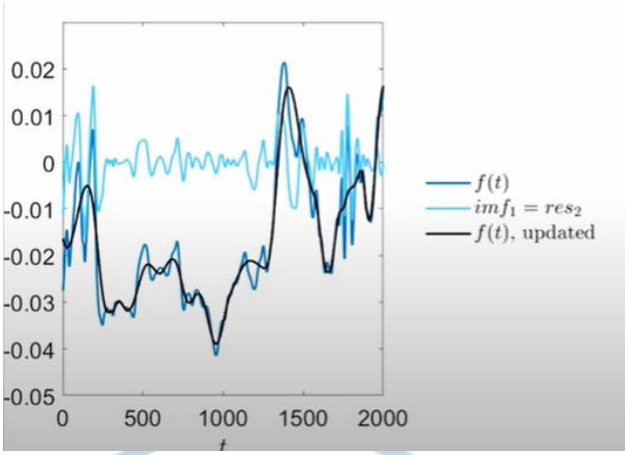
5. Apabila tidak memenuhi *stopping criterion*, maka sinyal $res(t)$ diperlakukan sebagai *input* sehingga didapatkan sinyal $res_2(t)$ yang kemudian juga akan dicek dengan syarat *stopping criterion*.



Gambar II.12 Visualisasi Langkah EMD Kelima

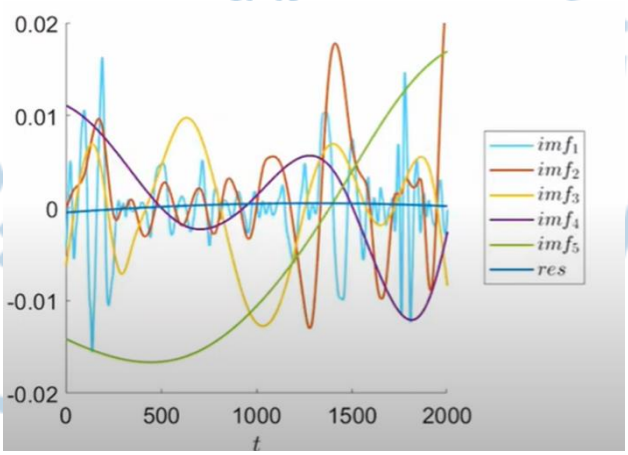
6. Jika memenuhi syarat, maka nilai $res_2(t) = imf_1(t)$. Proses iterasi langkah pertama hingga keenam akan berulang hingga didapatkan komponen osilasi paling sederhana yang tidak dapat didekomposisi lebih lanjut. Proses iterasi akan diulang dengan adanya perubahan pada *input* sinyal $f(t)$, yang didefinisikan pada persamaan II.19.

$f(t) = f(t) - imf_1(t)$……...……..…..(II.19)



Gambar II.13 Visualisasi Langkah EMD Keenam

7. Hasil iterasi dari proses EMD pada langkah pertama hingga keenam akan dihasilkan beberapa IMF dan sebuah sinyal residu berupa *monotonic function*, dengan sinyal residu tidak dapat didekomposisi lebih lanjut. Jumlah seluruh IMF dan sinyal residu adalah sinyal asli sebelum didekomposisi dengan EMD.



Gambar II.14 Visualisasi Hasil Akhir dari EMD

II.4 *Apnea-ECG Database*

MIT PhysioNet Apnea-ECG Database merupakan *database* yang dipublikasikan oleh PhysioNet pada tahun 2000 dalam rangkaian kompetisi CinC (*Computer in Cardiac Challenge* 2000)[16]. Apnea-ECG Database disediakan oleh Dr. Thomas Penzel, *Department of Internal Medicine, Philipps-University*,

Marburg, Germany. *Database* terdiri atas *dataset* dari 35 rekaman yang dipublikasikan secara umum dan 35 rekaman yang tidak dipublikasikan secara umum, dengan *sampling frequency* 100 Hz. Panjang rekaman bervariasi dari 401 sampai 587 menit, dengan setiap rekaman berisi sinyal ECG beserta anotasinya. Pemberian anotasi mengikuti standar yang telah ditetapkan oleh American Academy of Sleep Medicine (AASM) yang dijelaskan pada *AASM Manual for Scoring of Sleep and Associated Events*[20].

Dalam membentuk Apnea-ECG Database diagnosis *Sleep Apnea* dilakukan oleh ahli di sebuah laboratorium tidur (*sleep laboratory*) dengan menggunakan *cardiorespiratory polysomnography*[16], dimana prosedur tersebut membutuhkan rekaman EEG, EOG, dan EMG untuk menentukan tahapan tidur, aliran udara oronasal, pergerakan dinding dada dan dinding perut untuk menentukan usaha pernapasan, saturasi oksigen untuk memantau efek pernapasan,dan ECG untuk memantau detak jantung dan deteksi aritimia[16]. Dalam memberikan penilaian untuk menentukan apnea atau tidak, amplituda saat bernapas harus berkurang lebih dari 50% dari batas dasar (*baseline*), tingkat desaturasi oksigen lebih dari 3% dan terjadi selama paling tidak 10 detik. Hasil dari ketiga faktor tersebut dirangkum menjadi sebuah Apnea-Hypopnea Index (AHI) untuk durasi satu jam tidur[16]. Nilai AHI sampai dengan 5 dianggap normal, nilai AHI 5-15 dianggap sindrom *Obstructive Sleep Apnea-Hypopnea Syndrome* (OSAHS) ringan, nilai AHI 15-30 dianggap OSAHS sedang, dan AHI di atas 30 dianggap OSAHS berat[15].

Hasil analisa sinyal ECG tidak dapat menghasilkan AHI, karena AHI dihasilkan dari penilaian terhadap aliran udara oronasal, usaha pernapasan, dan saturasi oksigen; namun sinyal ECG mampu memberikan penilaian terhadap gangguan pernapasan (*breathing disosder*) yang sesuai dengan standar penilaian apnea[16]. Apnea-ECG Database dibentuk untuk mengenali penyakit gangguan tidur berdasarkan rekaman sinyal ECG tunggal, penilaian dilakukan oleh seorang ahli yang tidak lagi membedakan peristiwa Apnea dan *hypopnea* (terjadinya desaturasi oksigen akibat halangan secara parsial terhadap saluran pernapasan bagian atas), tetapi lebih ditujukan untuk mengenali peristiwa gangguan pernapasan. Peristiwa gangguan pernapasan dapat mengandung lebih dari satu peristiwa Apnea ataupun *hypopnea*, hasil penilaian tersebut dipetakan untuk

rekaman sinyal ECG berdurasi satu menit[16]. Setiap satu menit pada rekaman sinyal ECG memiliki anotasi berupa ‘A’ untuk Apnea, dan ‘N’ untuk Normal, seluruh peristiwa Apnea yang diberi anotasi ‘A’ bertipe *obstructive* (OSA) atau *mixed* (MSA)[15].

Rekaman ECG pada Apnea-ECG Database terbagi menjadi tiga kelas, kelas A (Apnea), kelas B (*Borderline*), dan kelas C (*Control*). Rekaman ECG pada kelas A harus mengandung paling sedikit satu jam dengan *apnea hypopnea index* (AHI) 10 atau lebih, kelas B sebanyak 5 atau lebih. *Dataset* yang dipublikasikan secara umum terdiri atas 20 rekaman kelas A, 5 rekaman untuk kelas B, dan 10 rekaman untuk kelas C.

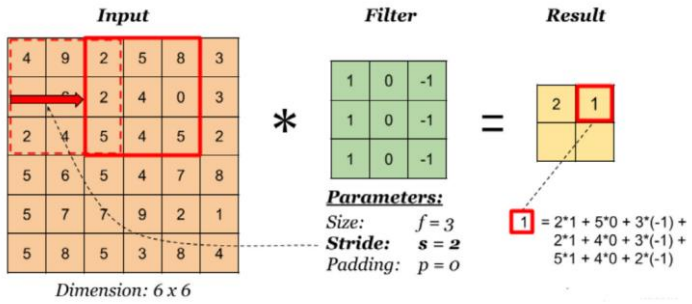
II.5 *Convolutional Neural Network* (CNN)

Convolutional Neural Network (CNN) merupakan salah satu metode *Deep Learning* yang berawal dari fungsi *visual cortex* pada otak yang digunakan untuk mengenali ciri dari data digital berupa sinyal dan citra[21]. Proses pengenalan ciri (*feature extraction*) pada CNN dilakukan dengan cara menerapkan prinsip konvolusi, hasil dari proses pengenalan ciri berupa *feature map* yang kemudian digunakan sebagai dasar untuk melakukan klasifikasi *input* ke dalam kelas tertentu pada *output layer*. *One Dimensional Convolutional Neural Network* merupakan salah satu aplikasi metode CNN dengan *input* berupa matriks satu dimensi, contoh: sinyal, arsitektur 1DCNN juga menyerupai arsitektur CNN pada umumnya yang terdiri atas *input layer*, *hidden layer*, dan *output layer*[21]. Pada Tugas Akhir digunakan 1DCNN karena *input* berupa sinyal ECG yang berdimensi satu atua bersifat *time series*.

II.5.1 *Convolutional Layers*

Convolutional Layer merupakan *layer* yang membedakan arsitektur CNN dengan arsitektur *Deep Learning* lainnya. Sesuai dengan namanya, *convolutional layer* terjadi proses konvolusi sebagia cara model untuk mengenali ciri dari *input*[21]. Proses konvolusi dilakukan dengan cara melakukan perkalian antar

elemen matriks *input* dengan *filter* (atau *kernel*) yang kemudian hasil perkalian akan ditambahkan sebagai hasil konvolusi untuk membentuk *feature map*[22]. Filter atau Kernel merupakan matriks yang berisi nilai pembobotan *input* untuk membentuk *feature map* dari *input* sebagai ciri yang dikenali. Proses konvolusi dijelaskan pada Gambar II.15.



Gambar II.15 Proses Konvolusi pada *Convolutional Layer*[24]

Merujuk pada Gambar II.15, terdapat tiga parameter *filter*/*kernel*, antara lain *size*, *stride*, dan *padding*. *Size* menentukan ukuran matriks *filter* yang digunakan, pada contoh ini digunakan *size* 3 sehingga ukuran matriks *filter* berukuran 3x3. *Stride* menuntukan pergeseran *filter*, pada contoh ini, dipilih nilai *stride* 2, sehingga *filter* bergeser dari kolom pertama ke kolom ketiga dari matriks *input*. Secara matematis, proses konvolusi dapat definisikan sebagai berikut[22].

$$z_{i,j,k} = b_k + \sum_{u=0}^{f_h-1} \sum_{v=0}^{f_w-1} \sum_{k'=0}^{f_{n'}-1} x_{i',j',k'} . w_{u,v,k',k} \quad \text{.....(II.17)}$$

dengan

$$\begin{cases} i' = i \times s_h + u \\ j' = j \times s_w + v \end{cases} \quad \text{.....(II.18)}$$

Keterangan:

$z_{i,j,k}$: neuron *output* yang terletak pada baris i, kolom j, pada *feature map* k.

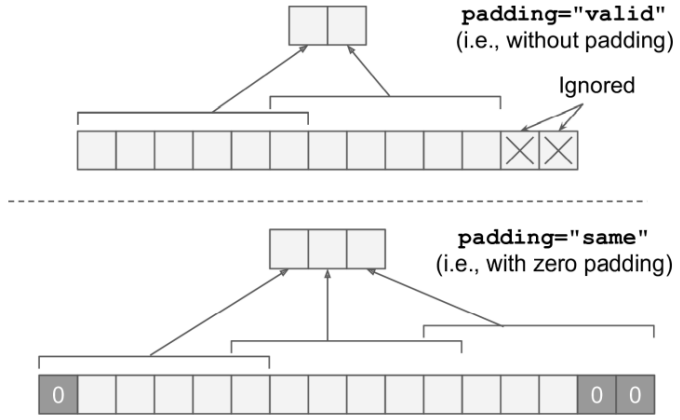
s_h, s_w, f_h, f_w , $f_{n'}$ secara berturut-turut adalah *stride* horizontal dan vertikal, tinggi dan lebar *field*, dan *feature map* pada *layer* sebelumnya (*layer* l – 1).

$x_{i',j',k'}$: neuron *output* pada *layer* l – 1, baris i', kolom j', pada *feature map* k'.

b_k: variabel *bias* untuk *feature map* k.

$w_{u,v,k',k}$: pembobotan neuron pada *feature map* k dari *layer* l dan *input*nya terletak pada baris u, kolom v dan *feature map* k'.

Padding menentukkan dimensi dari *input*, parameter *padding* memiliki dua nilai, antara lain "*same*" dan "*valid*". Apabila dipilih "*valid*", maka *convolutional layer* akan mengabaikan beberapa elemen matriks, sehingga tidak terdapat penambahan elemen pada sisi-sisi matriks bernilai nol. *Padding* "*same*" akan menambahkan nilai nol secara merata pada sisi-sisi matriks. Gambar II.16 menunjukkan perbedaan penggunaan *padding* "*same*" dengan "*valid*".



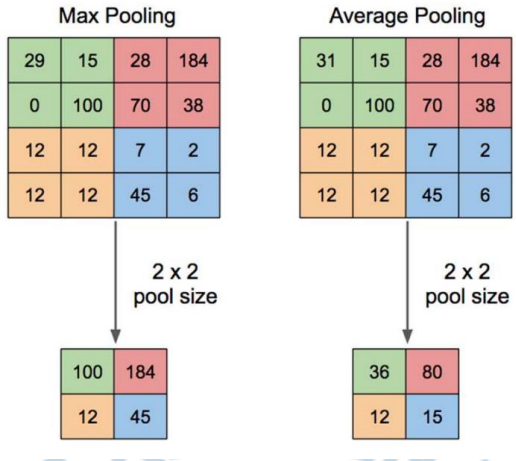
Gambar II.16 Perbedaan *Padding* "same" dan "valid"[22]

Merujuk pada Gambar II.16 terdapat 13 neuron *input*, ukuran *filter* = 6, dan *stride* = 5. Dengan *padding* "*same*", dimensi *output* dari *convolutional layer* didapat dari jumlah neuron *input* dibagi dengan *stride*, yang hasilnya dibulatkan ke atas. Dari contoh, didapat ukuran *output* sebesar 3, sehingga ditambahkan elemen pada sisi-sisi matriks tambahan bernilai 0 (*zeros*) secara merata. Apabila digunakan *padding* "valid", tidak terdapat penambahan elemen matriks, sehingga beberapa elemen matriks akan diabaikan seperti pada contoh.

II.5.2 *Pooling Layers*

Pooling layer berfungsi untuk mengecilkan dimensi *output* dari *convolutional layer* sehingga dapat mengurangi beban komputasi, penggunaan memori, dan jumlah parameter yang digunakan[22]. Terdapat dua jenis *pooling*, anatara lain *max pooling* dan *average pooling*. *Max pooling* akan menggunakan nilai terbesar (maksimum) dengan dimensi *pooling* yang digunakan terhadap nilai

pada matriks *input*, sedangkan *average pooling* akan menggunakan nilai rata-rata dari dimensi *pooling* yang digunakan terhadap nilai pada matriks *input*[22]. Gambar II.17 menunjukkan perbedaan *max pooling* dengan *average pooling*.



Gambar II.17 Perbedaan *Max Pooling* (Kiri) dan *Average Pooling* (Kanan) [22]

II.5.3 *Regularization*

Regularization merupakan metode yang digunakan untuk mengurangi *overfitting*, yaitu kondisi ketika pelatihan memberikan nilai *error* yang kecil, tetapi pada saat pengujian memberikan nilai *error* yang besar[22]. Teknik *regularization* antara lain: *Batch Normalization* dan *Dropout*.

II.5.3.1 *Batch Normalization*

Batch Normalization merupakan salah satu bentuk metode *regularization* yang digunakan untuk mengatasi masalah *exploding gradient* dan *vanishing gradient*[22]. *Exploding gradient* terjadi ketika nilai pembobotan terlalu tinggi, sedangkan *vanishing gradient* terjadi ketika nilai pembobotan berangsur-angsur menjadi rendah. *Batch Normalization* dilakukan dengan cara *zero-centering* dan normalisasi tiap input sehingga dihasilkan nilai pada rentang 0, kemudian *batch normalization* akan mengkoordinasi perubahan bobot agar tidak terjadi *exploding/vanishing gradient*. Perubahan bobot akan diatur melalui dua parameter vektor tiap *layer*, satu untuk penskalaan (*scaling*), dan satu untuk pergeseran

(*shifting*). Berikut adalah algoritma perhitungan *Batch Normalization* disajikan pada persamaan II.20 hingga persamaan II.23.

$$\boldsymbol{\mu}_B = \frac{1}{m_B}\sum_{i=1}^{m_B} \mathbf{x}^{(i)} \quad \text{(II.20)}$$

$$\boldsymbol{\sigma}_B^2 = \frac{1}{m_B}\sum_{i=1}^{m_B} \left(\mathbf{x}^{(i)} - \boldsymbol{\mu}_B\right)^2 \quad \text{(II.21)}$$

$$\hat{\mathbf{x}}^{(i)} = \frac{\mathbf{x}^{(i)} - \boldsymbol{\mu}_B}{\sqrt{\boldsymbol{\sigma}_B^2 + \varepsilon}} \quad \text{(II.22)}$$

$$\mathbf{z}^{(i)} = \gamma \otimes \hat{\mathbf{x}}^{(i)} + \boldsymbol{\beta} \quad \text{(II.23)}$$

Keterangan:

$\boldsymbol{\mu}_B$: vektor dari rerata *input*, dihitung dari seluruh *mini-batch* B (satu nilai rerata tiap *input*).

$\boldsymbol{\sigma_B}$: vektor dari standar deviasi *input*, dihitung dari seluruh *mini-batch* B (satu nilai standar deviasi tiap *input*).

$\boldsymbol{m_B}$: banyaknya sampel pada *mini-batch* B.

$\hat{\mathbf{x}}^{(i)}$: vektor yang terdiri atas nilai *input* setelah dilakukan proses *zero-centering* dan normalisasi.

$\boldsymbol{\gamma}$: *output* dari parameter yang digunakan untuk penskalaan.

$\otimes$: melambangkan proses perkalian tiap elemen pada vektor.

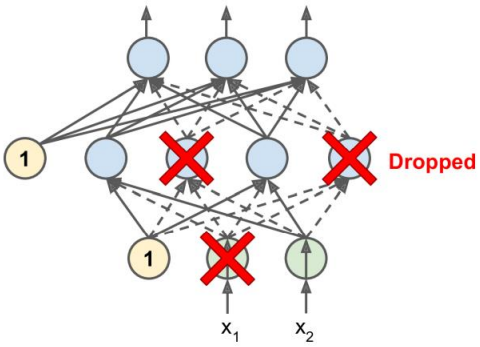
$\boldsymbol{\beta}$: parameter pergeseran *output* (*offset*) pada vektor.

$\boldsymbol{\varepsilon}$: nilai yang kecil untuk menghindari pembagian dengan nol (umumnya $\boldsymbol{\varepsilon}$ sebesar 10^{-5}).

$\mathbf{z}^{(i)}$: *output* dari operasi *Batch normalization*.

II.5.3.2 *Dropout*

Dropout merupakan metode *regularization* dengan cara "mengurangi", "membuang", atau "mengabaikan" sejumlah neuron pada saat pelatihan secara acak[22]. Berkurangnya jumlah neuron akan menekan kemungkinan terjadinya *overfitting*. Banyaknya neuron yang diabaikan pada *dropout* diatur pada parameter *dropout rate*, pada arsitektur CNN umumnya digunakan nilai *dropout rate* sebesar 40-50% [22]. Metode *dropout* dijelaskan pada Gambar II.18.



Gambar II.18 *Dropout*[23]

Merujuk pada Gambar II.18, garis putus-putus menunjukkan neuron yang diabaikan, sehingga nilai *output* dari neuron tersebut bernilai nol.

II.5.4 *Crossentropy Loss Function*

Dalam menilai performa model yang dirancang, digunakan fungsi yang mengukur kemampuan model dalam memprediksi sampel, disebut dengan *loss function*[22]. Jenis *loss function* yang digunakan pada Tugas Akhir adalah *crossentropy loss function*, dikarenakan akan dilakukan klasifikasi biner dalam menyelesaikan Tugas Akhir. *Crossentropy* juga dikenal dengan *logarithmic loss* atau *logistic loss*. Setiap nilai probabilitas dari kelas yang diprediksi dibandingkan dengan kelas sebenarnya sehingga dihasilkan skor *loss* yang dihitung dari seberapa jauh dengan hasil yang sebenarnya. Berikut adalah persamaan *Crossentropy Loss Function*.

$$L_{\mathrm{CE}} = -\sum_{i=1}^{n} t_i \log\left(p_i\right) \quad \text{(II.24)}$$

Keterangan:

n: banyak kelas

t_i: nilai kelas yang benar (*truth label*)

p_i: probabilitas *Softmax* untuk kelas ke- i.

II.5.5 *Activation Function*

Activation Function bertugas untuk mengenalkan nonlinearitas pada hasil *output* dari neuron[22]. *Activation Function* akan mengubah hasil dari jumlah pembebanan *input* sesuai dengan karakteristiknya, pada Tugas Akhir digunakan *Rectified Linear Unit* (ReLU) dan *sigmoid*.

II.5.5.1 *Rectified Linear Unit* (ReLU)

ReLU merupakan jenis *activation function* yang umum digunakan pada arsitektur *neural network* pada umumnya[22]. ReLU bersifat kontinu tetapi tidak dapat diturunkan pada nilai 0, dan nilai turunannya bernilai 0 untuk nilai *input* di bawah 0. Secara matematis, ReLU didefinisikan sebagai berikut.

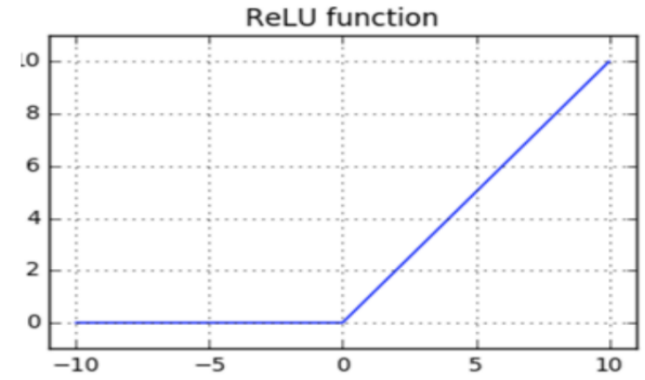
$$ReLU\ (z) = max(0, z) \quad \text{(II.25)}$$

Keterangan:

z: nilai *input*

max: fungsi maksimum untuk mencari nilai tertinggi

Pada Gambar II.19 disajikan *ReLU activation function*.



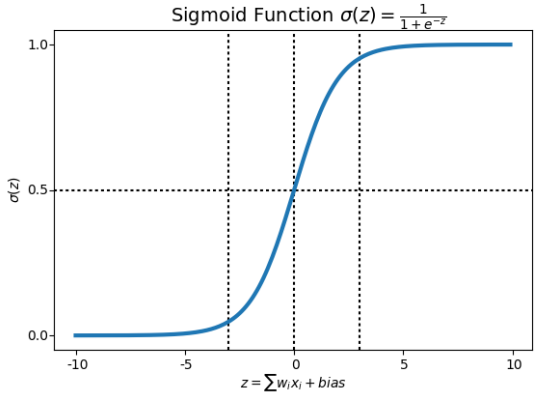
Gambar II.19 *ReLU Activation Function*[22]

Merujuk pada Gambar II.19, maka *ReLU* hanya akan meloloskan/mengaktivasi neuron yang bernilai positif, sedangkan neuron bernilai negatif akan dinonaktivkan (diabaikan).

II.5.5.2 *Sigmoid*

Sigmoid activation function merupakan fungsi aktivasi yang memiliki nilai turunannya nol untuk seluruh daerah[22]. Sigmoid akan memberikan nilai *output* berupa probabilitas untuk dua kelas, sehingga cocok digunakan untuk melakukan klasifikasi biner yang hanya terdapat dua kelas (kelas 0 dan 1). Adapun fungsi aktivasi Sigmoid dapat didefinisikan sebagai berikut.

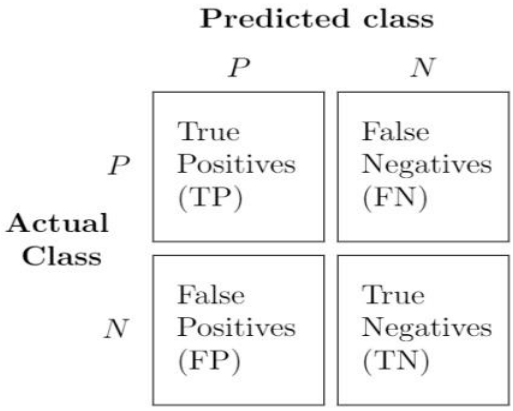
$$\sigma(z) = \frac{1}{1+e^{-z}} \quad \text{.....(II.25)}$$



Gambar II.20 *Sigmoid Activation Function*[22]

II. 6 *Metrics*

Dalam mengukur performa model dari hasil pelatihan dan ujinya, dibutuhkan *metrics* sebagai evaluasi terhadap model yang telah dibentuk. Performa model akan dinilai berdasarkan skor yang dihasilkan terhadap *metrics* yang digunakan. Dalam Tugas Akhir ini, digunakan sebuah *confusion matrix* sebagai satu cara merangkum performa model klasifikasi biner[23]. *Confusion matrix* terdiri atas bars dan kolom yang menandakan berapa jumlah masing-masing kelas sebenarnya (*actual class*) dan kelas prediksi (*predicted class*). Gambar II.22 menunjukkan sebuah *confusion matrix*, label P menandakan label 'Positif' atau kelas 1, sedangkan label N menandakan label 'Negatif'.



Gambar II.21 *Confusion Matrix*[26]

Merujuk pada Gambar II.22, terdapat empat kelas dalam sebuah *confusion matrix*, yaitu: *True Positive, True Negative, False Negative,* dan *False Positive*. Nilai yang terdapat pada empat kelas tersebut, didapatkan dari hasil perbandingan prediksi sampel pada kelas prediksi (*predicted class*) terhadap *ground truth* atau nilai sebenarnya yang ada pada kelas sebenarnya (*actual class*) sehingga dapat diklasifikasikan ke dalam empat kelas tersebut. Tabel II.3 menjelaskan maksud dari tiap kelas yang ada dalam sebuah *confusion matrix*.

Tabel II.3 Kelas pada *Confusion Matrix*[26]

Kelas	Deskripsi
True Positive (TP)	Hasil prediksi positif dan pada kenyataannya data positif., sehingga prediksi benar.
True Negative (TN)	Hasil prediksi negatif dan pada kenyataannya data negatif, sehingga prediksi benar.
Falase Positive (FP)	Hasil prediksi positif, tetapi pada kenyataannya data negatif, sehingga prediksi salah.
False Negative (FN)	Hasil prediksi negatif, tetapi pada kenyataannya data positif, sehingga prediksi salah.

Dari Tabel II.3 menunjukkan bahwa kelas *True Negative* dan *True Positive* menandakan jumlah prediksi negatif dan positif yang diprediksi benar, sedangkan kelas *False Negative* dan *False Positive* menandakan jumlah prediksi negatif dan positif yang diprediksi salah, atau tidak sesuai dengan kenyatannya[24]. *Metrics*

yang akan digunakan dalam Tugas Akhir ini antara lain: akurasi, sensitivitas (*recall*), spesifisitas, presisi, dan skor F-1, nilai-nilai *metrics* tersebut didapat dari keempat kelas pada *confusion matrix*.

II.6.1 Akurasi

Akurasi merupakan *metric* yang memberikan gambaran umum banyaknya sampel yang benar terhadap sampel yang salah digolongkan[23]. Nilai akurasi didapat dari perbandingan prediksi yang benar terhadap seluruh sampel yang dievaluasi, akurasi membandingkan kelas *True Positive* dan *True Negative* terhadap keempat kelas yang ada pada *confusion matrix*. Secara matematis, nilai akurasi dapat didefinisikan sebagai berikut.

$$Akurasi = \frac{TP+TN}{TP+FP+TN+FN} \quad \text{(II.16)}$$

Berdasarkan persamaan II.16, nilai dari akurasi memberikan gambaran performa model dalam memprediksi sampel yang nilainya sesuai dengan *ground truth*.

II.6.2 Presisi, Sensitivitas (*Recall*), dan Skor F-1

Dalam menilai *metrics* akurasi dibutuhkan *metrics* yang lebih bermakna, dibutuhkan *metrics* presisi, sensitivitas dan skor F-1 dalam memahami nilai akurasi yang telah didapatkan. Presisi digunakan untuk mengukur banyaknya sampel positif yang berhasil diprediksi benar terhadap total prediksi pada kelas positif. Sensitivitas (*Recall*) menunjukkan perbandingan antara sampel positif yang diprediksi benar terhadap seluruh sampel positif pada keadaan sebenarnya, sensitivtas juga dikenal sebagai *True Positive Rate* (TPR). Nilai *metrics* presisi dan sensitivitas umumnya diringkas menjadi nilai *metrics* skor F-1 sehingga dapat dengan mudah memahami relasi antara presisi dan sensitivitas. Skor F-1 didefinisikan sebagai nilai rerata harmonik dari presisi dan sensitivitas. Berbeda dengan nilai rerata pada umumnya yang memberikan bobot yang sama untuk semua nilai, rerata harmonik memberikan bobot lebih besar kepada nilai yang lebih rendah, sehingga nilai skor F-1 akan tinggi jika nilai presisi dan sensitivitas

tinggi[22]. Secara matematis, presisi, sensitivitas, dan skor F-1 didefinisikan sebagai berikut.

$$Presisi = \frac{TP}{TP+FP} \quad \text{(II.17)}$$

$$Sensitivitas = \frac{TP}{TP+FN} \quad \text{(II.18)}$$

$$Skor\ F-1 = 2 \times \frac{Presisi \times Sensitivitas}{Presisi + Sensitivitas} \quad \text{(II.19)}$$

Dalam praktiknya, skor F-1 lebih umum digunakan sebagai indikator pembeda yang baik untuk permasalahan klasifikasi biner.

II.6.3 Spesifisitas

Spesifisitas merupakan *metric* yang menunjukkan perbandingan antara sampel negatif yang diprediksi benar terhadap seluruh sampel negatif pada keadaan sebenarnya. Melalui nilai spesifisitas, dapat ditunjukkan performa model dalam memprediksi sampel negatif yang pada keadaan sebenarnya juga negatif, dengan kata lain, nilai spesifisitas merupakan *True Negative Rate* (TNR). Nilai spesifisitas dan sensitivitas dapat melengkapi satu sama lain, menunjukkan bagaimana performa model dalam memprediksi sampel positif dan negatif secara benar sesuai dengan keadaan sebenarnya. Secara matematis, nilai spesifitas dapat didefinisikan sebagai berikut.

$$Spesifisitas = \frac{TN}{TN+FP} \quad \text{(II.20)}$$

BAB III
PERANCANGAN SISTEM

Pada bab ini dijelaskan proses perancangan sistem dan perancangan arsitektur *One Dimensional Convolutional Neural Network* (1DCNN) untuk mendeteksi *Sleep Apnea*. Dalam Tugas Akhir ini, masukan berupa sinyal ECG dari *Apnea-ECG Database*. Masukan akan dilakukan tahap *signal preprocessing* sebelum dilatih pada tahap *training and testing.* Tahap *signal preprocessing* meliputi tahap normalisasi sinyal, tahap *signal filtering,* dan tahap *Empirical Mode Decomposition* (EMD).

III.1 Perancangan Sistem

Pada tahap ini dilakukan perancangan sistem proses deteksi Apnea dimulai dari masukan, tahap *signal preprocessing* dan tahap *training and testing*. Masukan tahap normalisasi berupa sinyal ECG awal dari *Apnea-ECG Database,* kemudian keluaran dari tahap normalisasi berupa sinyal ECG yang sudah dinormalisasi akan menjadi masukan dari tahap *signal filtering*. Tahap *signal filtering* akan dilakukan *filtering* pada sinyal ECG yang telah dinormalisasi dengan menggunakan IIR Notch Filter orde 2 dengan frekuensi *cutoff* (f_c=0.5 Hz). Sinyal ECG keluaran hasil tahap *signal filtering* akan menjadi masukan dari tahap EMD. Tahap EMD akan medekomposisi sinyal ECG hasil *filter* dan normalisasi hingga menjadi komponen osilasi sederhana berupa *Intrinsic Mode Function* (IMF). Hasil keluaran tahap EMD yang berupa IMF akan menjadi masukan pada tahap *training and testing* yang terlebih dahulu dilakukan pembagian sampel data menjadi beberapa bagian untuk *training*, *testing*, dan validasi dengan perbandingan 80:10:10.

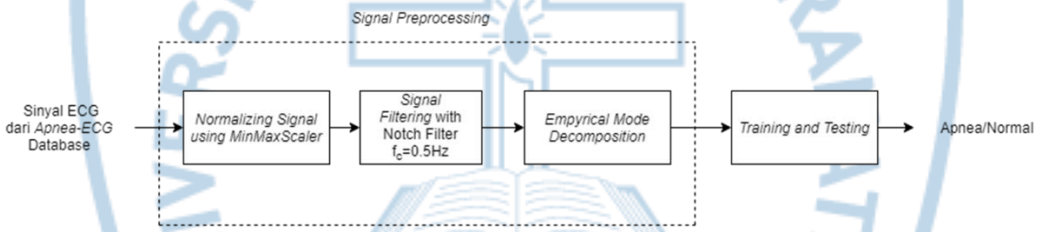
Data yang telah terbagi akan dilakukan tahap *data training* dengan menggunakan arsitektur 1DCNN yang dirancang oleh Hung-Yu Chang pada jurnal *"A Sleep Apnea Detection System Based on a One-Dimensional Deep Convolutional Neural Network Using Single-Lead Electrocardiogram"*[15]. Dalam

penelitiannya dengan *dataset* dari *Apnea-ECG Database*, arsitektur 1DCNN yang dirancangnya berhasil memberikan nilai *metrics* akurasi 87.9%, sensitivitas 81.1%, dan spesifisitas 92%.

Dari arsitektur tersebut akan didapatkan hasil keluaran berupa deteksi peristiwa *Sleep Apnea*. Hasil tersebut akan mendapatkan dua kelas keluaran, yaitu

1. Kelas 0 merupakan bukan peristiwa Apnea (Normal).
2. Kelas 1 merupakan peristiwa Apnea.

Dalam menguji performa arsitektur digunakan *confusion matrix* dalam merangkum informasi hasil uji model. Nilai *metrics* berupa akurasi, sensitivitas (*recall*), spesifisitas, presisi, dan skor F-1 akan digunakan dalam menguji performa arsitektur dalam mendeteksi persitiwa *Sleep Apnea* ke dalam kelas Normal atau Apnea. Diagram blok proses deteksi peristiwa *Sleep Apnea* pada Tugas Akhir ini ditunjukkan pada Gambar III.1.



Gambar III.1 Diagram Blok Proses Deteksi Apnea

III.1.1 Masukan (*Input*)

Masukan dari *Apnea-ECG Database* berupa 35 rekaman sinyal ECG, 20 rekaman sinyal (a01-a20) kelas A (Apnea), 5 rekaman sinyal (b01-b05) kelas B (*Borderline*), dan 10 rekaman sinyal (c01-c10) kelas C (*Control*), masing-masing memiliki durasi yang bervariasi dari 401 sampai 587 menit dengan setiap rekaman berisi sinyal ECG beserta anotasinya. Anotasi diberikan setiap satu menit dalam rekaman sinyal ECG. Dari variasi durasi rekaman tersebut, untuk tiap rekaman dipotong menjadi rekaman dengan durasi satu jam, dengan maksimal hasil potongan berjumlah delapan rekaman berdurasi satu jam. Dari total 35 rekaman tersebut kemudian dihasilkan 260 rekaman berdurasi satu jam dari a01_s1 hingga c01_s7 beserta data anotasinya.

Data rekaman sinyal berupa *array* berukuran 1x360000 untuk rekaman bedurasi satu jam, nilai yang terdapat dalam *array* merupakan amplituda sinyal yang di*sampling* dengan frekuensi *sampling* ($f_s = 100\ Hz$), sehingga sinyal ECG memiliki periode 0.01 detik, oleh karena itu, untuk rekaman berdurasi satu jam memiliki 360000 nilai amplituda. Karena data berdurasi satu jam, sehingga terdapat data 60 anotasi yang diberikan setiap menit dari rekaman sinyal ECG, dengan label 'A' untuk peristiwa Apnea dan label 'N' untuk peristiwa Normal. Gambar III.2 menunjukkan ukuran dari data rekaman a01_s1 dan data anotasinya. Gambar III.3 menunjukkan isi dari data rekaman a01_s01 dan data anotasi untuk rekaman a01_s01.

```
Record a01_s1 value shape:  (1, 360000)
Record a01_s1 annotation shape:  (60, 1)
```

Gambar III.2 Ukuran Data Rekaman a01_s1 dan Data Anotasi a01_s1

```
Isi dari record a01_s1:
[[-12 -13 -12 ... -42  -9 -21]]
Isi dari anotasi record a01_s1:
0     N
1     N
2     N
3     N
4     N
5     N
6     N
7     N
```

Gambar III.3 Isi Data Rekaman a01_s1 dan Data Anotasi a01_s1

Merujuk pada Gambar III.3, isi dari data rekaman a01_s1 yang ditampilkan hanya tiga data pertama dan tiga data terakhir, dan isi dari data anotasi rekaman a01_s1 ditampilkan 8 data anotasi pertama dari *index* 0 sampai ke *index* 7. Data rekaman kemudian akan digunakan sebagai masukan untuk tahap *signal prerocessing*.

III.1.2 *Signal Preprocessing*

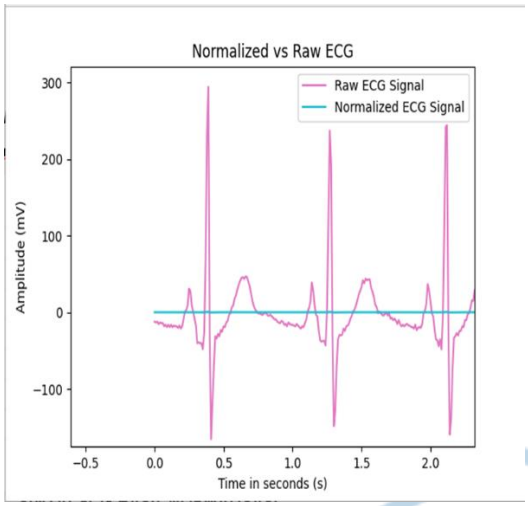
Tahap *signal preprocessing* terbagi menjadi tiga tahap, yaitu: tahap normalisasi, tahap *signal filtering*, dan tahap *Empirical Mode Decomposition*. Masukan dari tahap *signal preprocessing* berupa 260 data rekaman berdurasi satu jam yang merupakan *array* berukuran 1x360000.

III.1.2.1 Normalisasi

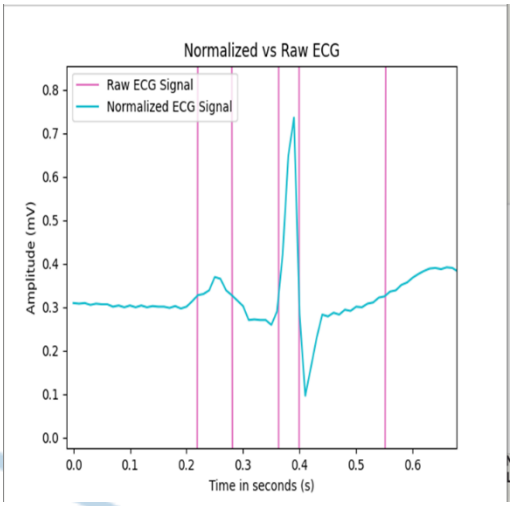
Tahap normalisasi sinyal merupakan upaya untuk menyeragamkan nilai *input* pada tahap *data training*. Sinyal ECG dari Apnea-ECG Database memiliki jangkauan nilai amplituda (dalam mV) dalam bilangan bulat yang bervariasi dari -120mV sampai 480 mV, sehingga dilakukan normalisasi dengan metode MinMaxScaler yang akan menskalakan nilai sinyal ECG menjadi dari rentang 0 hingga 1 untuk menyeragamkan nilai sinyal ECG. Kode program yang digunakan untuk melakukan normalisasi sinyal adalah sebagai berikut.

```
from scipy.io import loadmat
from sklearn.preprocessing import MinMaxScaler
import numpy as np
import pandas as pd


def normalization (record):
    global val
    'Read record values'
    record = loadmat("cleaned data/ECG2/a01_s1.mat") #read from
a01_s1 matlab file
    val = record['val']
    val = np.array(val)

    'Scaler/MinMax Normalize only'
    scaler = MinMaxScaler()
    norm_val = scaler.fit_transform(val)

    return norm_val
```

Setelah dilakukan normalisasi, Gambar III.4 dan Gambar III.5 menunjukkan perbandingan sinyal ECG sebelum dan sesudah dilakukan normalisasi. Terlihat bahwa sinyal ECG ternormalisasi memiliki rentang nilai amplituda dari 0 hingga 1. Keluaran dari tahap normalisasi berupa sinyal ECG ternormalisasi dan digunakan sebagai masukan dari tahap *signal filtering*.



Gambar III.4 Perbandingan Sinyal ECG Sebelum dan Sesudah Normalisasi



Gambar III.5 Detail Perbandingan Sinyal ECG Sebelum dan Sesudah Normalisasi

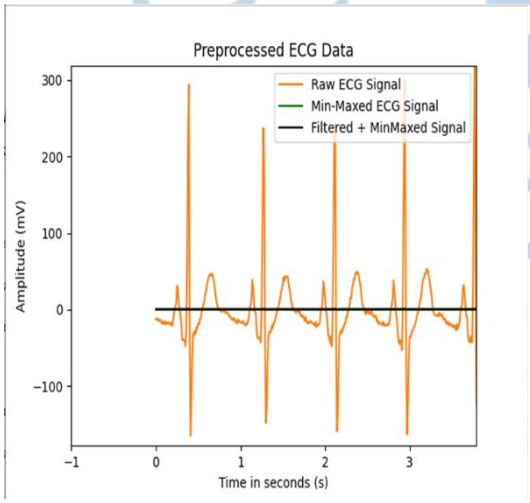
III.1.2.2 *Signal Filtering*

Tahap *signal filtering* menerima masukan berupa sinyal ECG ternormalisasi dimana nilai amplituda sinyal sudah diskalakan dengan rentang nilai 0 hingga 1. Pada tahap ini, dilakukan *filtering* terhadap sinyal untuk mengurangi *noise* yang hadir pada sinyal ECG hasil rekaman dari alat *Electrocardiograph*. Sinyal ECG dari Apnea-ECG Database yang di*sampling* dengan frekuensi *sampling* ($f_s = 100\ Hz$) sudah terbebas dari *powerline interference noise* sehingga tidak dibutuhkan *Low Pass Filter* untuk mengatasi *noise* tersebut. Namun, terdapat *noise* lainnya juga yang masih mempengaruhi sinyal ECG yang disebut dengan *baseline wander noise* yang disebabkan oleh pergerakan elektroda alat *Electrocardiograph* akibat pergerakan pasien selama rekaman yang berada pada frekuensi 0.5 Hz (merujuk pada pembahasan di Bab II.2). Oleh karena itu digunakan *IIR Notch Filter* orde 2 dengan frekuensi *cutoff* ($f_c = 0.5\ Hz$) untuk mengurangi *baseline wander noise*. Kode program tahap *signal filtering* adalah sebagai berikut.

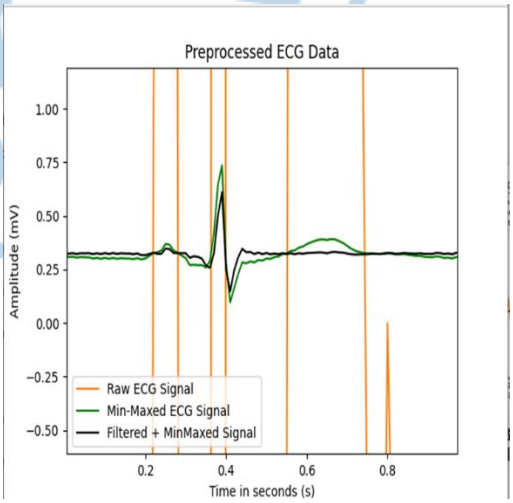
```
from scipy.io import loadmat
import numpy as np
import pandas as pd
from scipy import signal
from scipy.signal import butter, filtfilt, iirnotch, savgol_filter


def filter_final():
   'Read record values'
```

```
    record = loadmat("cleaned data/ECG2/a01_s1.mat") #read from
a01_s1 matlab file
    val = record['val']
    val = np.array(val)

    'Scaler/MinMax Normalize only'
    scaler = MinMaxScaler()
    norm_val = scaler.fit_transform(val)
    time = [0]
    i=0
    x=0
    while i<len(val)-1:
        x = x + 0.01
        x  = round(x,4)
        time.append(x)
        i = i+1
    cutoff=0.5
    sample_rate=100
    norm_val_tp = np.transpose(norm_val)
    data=norm_val_tp
    b, a = iirnotch(cutoff, Q = 0.005, fs = sample_rate)
    filtered_data = filtfilt(b, a, data)
    return filtered_data, time
```

Gambar III.6 dan Gambar III.7 menunjukkan perbandingan sinyal ECG sebelum normalisasi, sinyal ECG ternormalisasi, dan sinyal ECG hasil filter.



Gambar III.6 Perbandingan Sinyal ECG Sebelum Normalisasi, Sinyal ECG Ternormalisasi, dan Sinyal ECG Hasil *Filter*



Gambar III.7 Detail Perbandingan Sinyal ECG Sebelum Normalisasi, Sinyal ECG Ternormalisasi, dan Sinyal ECG Hasil *Filter*

Sinyal ECG sebelum dan sesudah normalisasi memiliki bentuk sinyal yang serupa, namun sinyal ECG hasil *filter* dengan *IIR Notch Filter*, menunjukkan sedikit perbedaan pada interval S-T dari sinyal ECG karena karakteristik *Notch Filter* yang menghilangkan komponen sinyal pada frekuensi *cutoff* ($f_c = 0.5\ Hz$). Keluaran tahap *signal filtering* berupa sinyal ECG ternormalisasi yang telah di*filter*, yang kemudian akan menjadi masukan tahap *Empirical Mode Decomposition* (EMD).

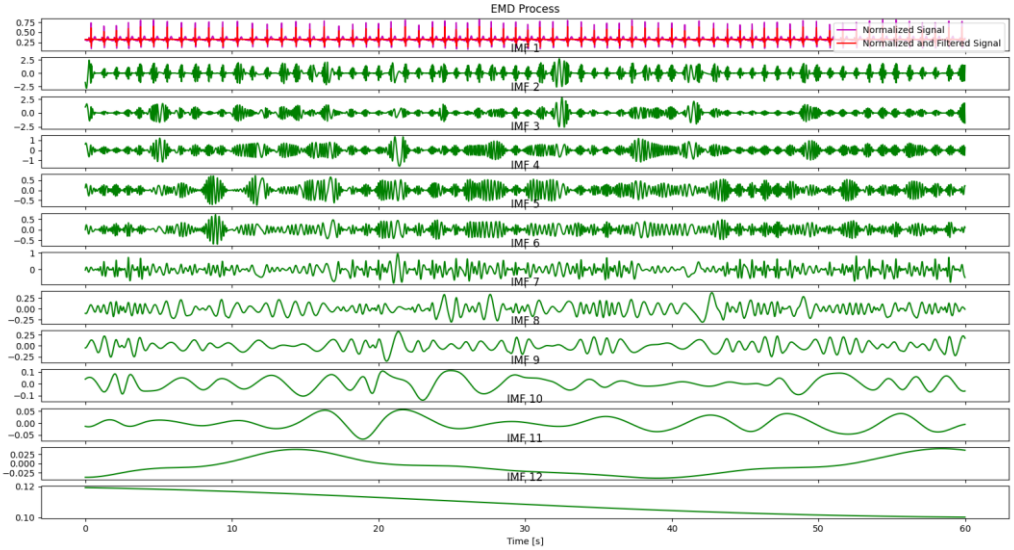
III.1.2.3 *Empirical Mode Decomposition* (EMD)

Tahap *Empirical Mode Decomposition* (EMD) menerima masukan berupa sinyal ECG hasil tahap normalisasi dan tahap *signal filtering*. Pada tahap ini, sinyal ECG ter*filter* akan didekomposisi menjadi komponen osilasi sederhana, yang kemudian disebut sebagai *Intrinsic Mode Function* (IMF). Penggunaan teknik EMD dalam mendekomposisi sinyal menjadi komponen osilasi sederhana merupakan upaya yang dilakukan untuk mendekatkan sinyal ECG ke bentuk aslinya sehingga komponen osilasi yang tidak dibutuhkan seperti *noise* dapat dihilangkan. Keluaran dari tahap EMD akan menghasilkan beberapa IMF sesuai dengan kompleksitas sinyal ECG yang didekomposisi. Seiring bertambahnya IMF, maka IMF yang dihasilkan merupakan komponen osilasi lebih sederhana dibanding IMF sebelumnya, dengan demikian, IMF terakhir merupakan komponen osilasi paling sederhana dari suatu sinyal. Berikut adalah kode program untuk melakukan proses EMD.

```
from PyEMD import EMD, Visualisation
import numpy as np
import pandas as pd

def emd_signal(filtered_data_tp,time):
    'Define time array from time in filter_final funct.'
    time_arr = np.array(time)
    t = time_arr
    'S_tp = Signal Input from Filtered+Normalized Signal'
    S_tp = filtered_data_tp.flatten()

    'EMD Process'
    emd = EMD() #run EMD process
    emd.emd(S_tp)
```

```
    imfs, res = emd.get_imfs_and_residue()
    data = pd.DataFrame(data = imfs) #data = data IMF hasil EMD


    'ambil imf ke-1,2,3,4 dari row ke 0,1,2,3'
    data_i_1 = data.iloc[[0]]
    data_i_2 = data.iloc[[1]]
    data_i_3 = data.iloc[[2]]
    data_i_4 = data.iloc[[3]]

    'convert dataframes to np in order to use np.reshape()'
    data_i_1_conv = data_i_1.to_numpy()
    data_i_2_conv = data_i_2.to_numpy()
    data_i_3_conv = data_i_3.to_numpy()
    data_i_4_conv = data_i_4.to_numpy()
    'from 1 x 360.000 reshape to 60x6000, 60 rows and 6000 cols'
    data_i_1_conv = data_i_1_conv.reshape(60,6000)
    data_i_2_conv = data_i_1_conv.reshape(60,6000)
    data_i_3_conv = data_i_1_conv.reshape(60,6000)
    data_i_4_conv = data_i_1_conv.reshape(60,6000)


    'conv back to pandas dataframe in order to conv them to csv
files...'
    data_i_1_conv = pd.DataFrame(data_i_1_conv)
    data_i_2_conv = pd.DataFrame(data_i_2_conv)
    data_i_3_conv = pd.DataFrame(data_i_3_conv)
    data_i_4_conv = pd.DataFrame(data_i_4_conv)

    return data_i_1_conv, data_i_2_conv, data_i_3_conv,
data_i_4_conv
```

Gambar III.8 menunjukkan perbandingan sinyal ECG hasil *filter* dengan hasil EMD yang berupa IMF. Sinyal ECG ternormalisasi digambarkan dengan garis berwarna ungu, sinyal ECG hasil *filter* digambarkan dengan garis berwarna merah, dan IMF dari sinyal ECG hasil EMD digambarkan dengan garis berwarna hijau.



Gambar III.8 Perbandingan Sinyal ECG Ternormalisasi,
Sinyal ECG Hasil *Filter*, dan IMF Hasil EMD

Merujuk pada Gambar III.8 (disajikan dengan lebih jelas pada Lampiran B) ditunjukkan bahwa hasil proses EMD dari sinyal ECG ternomalisai yang sudah di *filter* memberikan keluaran sebanyak 12 IMF. Menurut penelitian yang telah dilakukan sebelumnya, IMF yang digunakan merupakan IMF yang masih menggambarkan karakteristik sinyal asli masukannya, dalam kasus ini IMF 1 – 2[25], sehingga digunakan IMF 1 – 2 pada *input layer* dari arsitektur 1DCNN yang digunakan.

Merujuk kepada kode program proses EMD, terdapat proses *resize* terhadap keluaran IMF hasil EMD. Masukan pada proses EMD merupakan sinyal ECG ternormalisasi dan sudah di*filter* berupa *array* berukuran 1x360000 sehingga keluaran dari hasil EMD dengan 12 IMF merupakan *array* berukuran 12x360000. Tiap baris pada *array* tersebut merupakan tipe IMFnya, dari IMF 1 – IMF 12. Kemudian, untuk menyesuaikan data *training* dengan *ground truth* berupa anotasi ‘A’ dan ‘N’ yang diberikan untuk setiap rekaman sinyal ECG berdurasi satu menit, maka dilakukan tahap *resize* sehingga sinyal ECG dicuplik menjadi segmen satu menit. Keluaran dari tahap EMD dalam bentuk IMF berupa *array* berukuran 60x6000, dengan tiap baris dari *array* merupakan nilai IMF dari rekaman sinyal ECG berdurasi satu menit.

III.1.3 *Data Training and Testing*

Pada tahap *data training and testing* akan dilakukan pelatihan dan pengujian pada model arsitektur 1DCNN yang akan digunakan dalam mendeteksi peristiwa Apen dengan sinyal ECG. Pada tahap data *training and testing*, akan dilakukan penyesuaian terlebih dahulu pada *input layer* dari arsitektur 1DCNN yang digunakan, yakni tahap *resize and annotation mapping*, dan tahap *train-test-split*. Kemudian setelah data terbagi menjadi data latihan, data uji, dan data validasi, maka akan dilakukan pelatihan terhadap arsitektur 1DCNN yang dirancang oleh Hung-Yu Chang[15].

III.1.3.1 *Resize and Annotation Mapping*

Hasil tahap *signal preprocessing* sebelumnya berupa IMF dengan *array* berukuran 60x6000 untuk tiap satu data rekaman sinyal ECG berdurasi satu jam. Terdapat 260 jumlah keseluruhan data rekaman sinyal ECG berdurasi satu jam hasil tahap *signal processing* dengan ukuran *array* 60x6000. Kemudian, 260 data tersebut digabung ke dalam suatu *array* berukuran 15600x6000 yang mencakup keseluruhan data.

Serupa dengan data rekaman sinyal ECG hasil tahap *signal preprocessing*, data anotasi rekaman juga harus digabung menjadi suatu *array* berukuran 15600x1 yang disesuaikan dengan data rekaman sinyal ECG hasil tahap *signal preprocessing*. Setiap baris menandakan anotasi untuk satu menit rekaman sinyal ECG hasil tahap *preprocessing*. Kemudian, data anotasi merupakan kumpulan data label ‘A’ dan ‘N’ akan diubah nilainya sesuai dengan pembagian kelas yang ada, kelas 0 untuk peristiwa Normal, dan kelas 1 untuk peristiwa Apnea. Berikut adalah kode program untuk menggabungkan data rekaman dan anotasi, serta mengubah label anotasi.

```
import pandas as pd
import numpy as np

new_annot_dir = 'E:/Felix/1822041/Tugas
Akhir/annot/annot_binary_final.csv'
annotate = pd.read_csv(new_annot_dir)
print('Annotation shape: ', annotate.shape)
```

```
df_imf_1_dir = 'E:/Felix/1822041/Tugas
Akhir/new/imf_1/record_imf1_a19_s8.csv'

'concat with'
df2_imf_1_dir= 'E:/Felix/1822041/Tugas
Akhir/new/imf_1/record_imf1_c10_s7.csv'


'IMF1'
df_imf_1 = pd.read_csv(df_imf_1_dir)
print('Success reading df_imf1_a01_b05 from : ', df_imf_1_dir,' with
shape: ', df_imf_1.shape)

df2_imf_1= pd.read_csv(df2_imf_1_dir)
print('Success reading df_imf1_c10 from : ', df2_imf_1_dir,' with
shape: ', df2_imf_1.shape)

df_concat_imf1 = pd.DataFrame([])
df_concat_imf1 =
df_concat_imf1.append(df_imf_1).append(df2_imf_1)#concat files
save_df_concat_imf1 = 'E:/Felix/1822041/Tugas
Akhir/new/imf_1/record_imf1_final.csv'

print('Concat success... with shape: ', df_concat_imf1.shape)
print('Saving concated files to ', save_df_concat_imf1)
df_concat_imf1.to_csv(save_df_concat_imf1,index=False)
print('Success saving df_concat_imf1 files to: ',
save_df_concat_imf1, ' with shape: ', df_concat_imf1.shape)

'Annot Mapping'
annot_map = annotate.to_numpy()
annot_map = np.where(annot_map == 'N', 0,annot_map)
annot_map = np.where(annot_map == 'A', 1,annot_map)

annot_map = np.array(annot_map)
annot_map = pd.DataFrame(annot_map)
annot_map.to_csv('D:/Kuliyah/.Tugas Akhir/cleaned
data/annot/test/annotNew_c10_s7.csv',index=False)
```

Pada Gambar III.9 ditunjukkan hasil dari tahap *resize and annotation mapping*.

```
Annotation shape:  (15600, 1)
Number of As in annot:  6065
Number of Ns in annot:  9535
Begin train funct
data input shape before train-test-split:  (15600, 6000, 1)
annot input shape before train-test-split:  (15600, 1)
```

Gambar III.9 Hasil Tahap *Resize* and *Annotation Mapping*

Merujuk pada Gambar III.9 terlihat bahwa data pada Apnea-ECG Database bersifat *imbalance*, atau tidak seimbang dengan perbandingan mendekati 6:9. Didapati bahwa sampel kelas 1 (peristiwa Apnea) lebih sedikit dibanding sampel kelas 0 (peristiwa Normal). Dalam Tugas Akhir ini, dilakukan perbandingan terhadap penggunaan *imbalanced dataset* dengan *balanced dataset* dan dilihat pengaruhnya terhadap hasil *metrics* yang digunakan dalam melakukan deteksi terhadap peristiwa *Sleep Apnea*.

Demi mencapai *balanced dataset* ditempuh dengan cara membuang sebagian sampel kelas 0 (peristiwa Normal) sehingga jumlah sampel kelas 0 (peristiwa Normal) dan kelas 1 (peristiwa Apnea) seimbang. Teknik membuang sebagian sampel kelas 0 dilakukan dengan cara menggabungkan data rekaman sinyal ECG hasil tahap *preprocessing* dengan data anotasi yang sudah diubah menjadi nilai 0 dan 1 pada kolom terakhir setelah data sinyal, dan kemudian sampel yang bernilai 0 pada kolom terkahir tersebut dibuang secara acak sehingga jumlah sampel bernilai 0 sama dengan jumlah sampel bernilai 1, yakni sebanyak 6065 sampel. Berikut adalah kode program yang digunakan untuk mencapai *balanced dataset*.

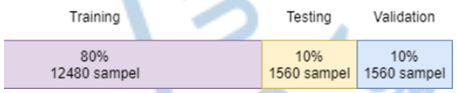
```
import pandas as pd
import numpy as np


new_annot_dir = 'E:/Felix/1822041/Tugas
Akhir/annot/annot_binary_final.csv'
annotate = pd.read_csv(new_annot_dir)
print('Annotation shape: ', annotate.shape)

'Data Final dgn array (15600,600)'
save_df_concat_imf1 = 'E:/Felix/1822041/Tugas
Akhir/new/imf_1/record_imf1_final.csv'

'Menggabungkan Annot dengan Data Final'
df_final_concat_imf_1 = pd.read_csv(save_df_concat_imf1)
'Melettakan data Annot pada kolom terakhir'
df_final_concat_annot_imf_1 = pd.concat([df_final_concat_imf_1,
annotate],axis=1)
print('ANNOT Concat  success... with shape: ',
df_final_concat_annot_imf_1.shape)
save_df_final_concat_annot_imf_1 = 'E:/Felix/1822041/Tugas
Akhir/new/imf_1/recordAnnot_imf1_final.csv'
```

```
df_final_concat_annot_imf_1.to_csv(save_df_final_concat_annot_imf_1,
index=False)
print('Success saving df_final_concat_imf_1 files to: ',
save_df_final_concat_annot_imf_1, ' with shape: ',
df_final_concat_annot_imf_1.shape)
```

Keluaran dari *balanced dataset* berupa *array* data rekaman berukuran 12130x6000 dan *array* data anotasi berukuran 12130x1. Keluaran tersebut kemudian akan dibagi menjadi data latih, data uji, dan data validasi pada tahap *train-test-split*.

III.1.3.2 *Train-Test-Split*

Pada tahap *train-test-split* akan dilakukan pembagian data dengan proporsi tertentu menjadi data latih, data uji, dan data validasi. Perbandingan pembagian data yang digunakan dalam Tugas Akhir ini adalah 80:10:10, untuk data latih, data uji, dan data validasi. Dari *imbalanced dataset*, maka didapat proporsi pembagian data seperti pada Gambar III.10, yakni terdapat 12480 sampel data latih, 1560 sampel data uji, dan 1560 sampel data validasi. Dari *balanced dataset*, maka didapat perbandingan pembagian data seperti pada Gambar III.11, yakni terdapat 9704 sampel data latih, 1230 sampel data uji, dan 1230 sampel data validasi.



Gambar III.10 Perbandingan Pembagian Data pada *Imbalanced Dataset*

Gambar III. 11 Perbandingan Pembagian Data pada *Balanced Dataset*

Kode program yang digunakan untuk melakukan pembagian data menjadi data latih, data uji, dan data validasi pada tahap *train-test-split* adalah sebeagai berikut.

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

df_2 = pd.read_csv('E:/Felix/1822041/Tugas
Akhir/new/imf_1/recordAnnot_imf1_final.csv')
df_2 = df_2.to_numpy()
```

```
annot = pd.read_csv('E:/Felix/1822041/Tugas
Akhir/annot/annot_binary_final.csv')
'Scale first then train-test-split'
print('Begin Scaler')
scaler = StandardScaler()
df_2_scaled = scaler.fit_transform(df_2)
print('df scaled with StandardScaler()')


'Expand dims before train-test-split'
df_2_scaled = np.expand_dims(df_2_scaled, axis=-1)
def train(df_2_scaled, annot_map):
    print('Begin train funct')
    global X_train, y_train
    global X_test, y_test
    global X_val, y_val
    global train_x_scaled,test_x_scaled

    X_train, X_test, y_train, y_test = train_test_split(df_2_scaled,
annot_map, test_size=0.2,stratify=np.array(annot))
    X_train, X_val, y_train, y_val = train_test_split(X_train,
y_train, test_size=0.25,stratify=np.array(y_train))
    'Tipe data float32'
    X_train = np.asarray(X_train).astype(np.float32)
    X_test = np.asarray(X_test).astype(np.float32)
    X_val = np.asarray(X_val).astype(np.float32)
    y_train = np.asarray(y_train).astype(np.float32)
    y_val = np.asarray(y_val).astype(np.float32)
    y_test = np.asarray(y_test).astype(np.float32)
    print('X_train shape: ', X_train.shape,'y_train shape: ',
y_train.shape)
    print('X_test shape: ', X_test.shape,'y_test shape: ',
y_test.shape)
    print('X_val shape: ', X_val.shape,'y_val shape: ', y_val.shape)
    return X_train, y_train, X_test, y_test, X_val, y_val
```

Sebelum data rekaman sinyal ECG hasil tahap *preprocessing* dilakukan pembagian data, data sinyal tersebut dilakukan standarisasi dengan *StandardScaler()* sehingga data sinyal memiliki nilai yang mendekati nol, baik nilai positif dan negatif. Setelah itu, pada data rekaman sinyal ECG juga dilakukan penambahan dimensi, sehingga dari sebelumnya merupakan *array* berukuran 15600x6000 diubah menjadi *array* berukuran 15600x6000x1. Hal ini dilakukan sebagai syarat pada *input layer* dari arsitektur 1DCNN yang hanya memiliki 1

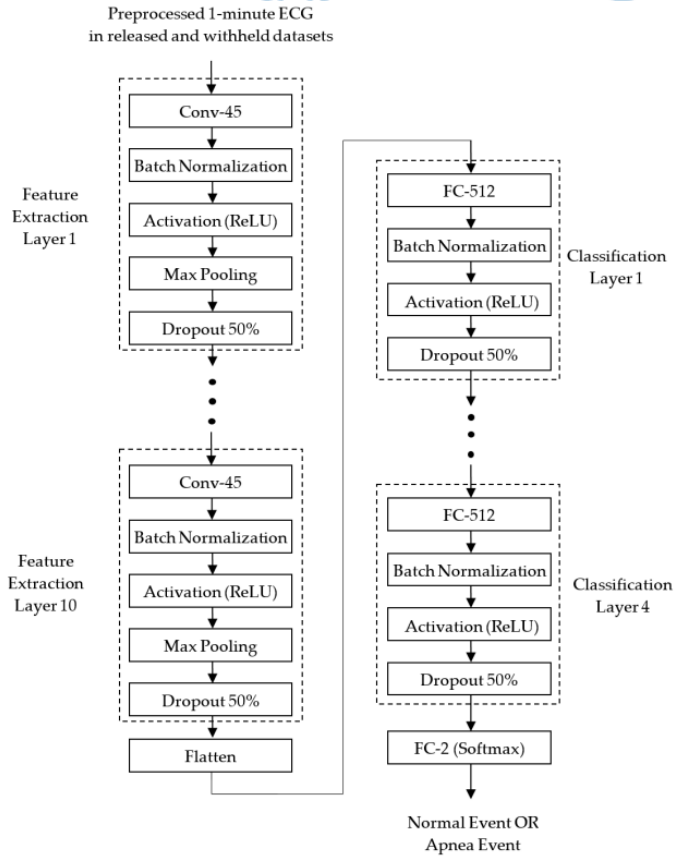
channel. Keluaran dari tahap *train-test-split* merupakan *array* yang menunjukkan pembagian data untuk data latih, data uji, dan data validasi. Hasilnya dapat dilihat pada Gambar III.12.

```
X_train shape:  (12480, 6000, 1) y_train shape:  (12480, 1)
X_test shape:  (1560, 6000, 1) y_test shape:  (1560, 1)
X_val shape:  (1560, 6000, 1) y_val shape:  (1560, 1)
```

Gambar III.12 Hasil Keluaran Tahap *Train-Test-Split*

III.2 Perancangan *Arsitektur 1DCNN*

Pada tahap ini dilakukan pelatihan ke arsitektur 1DCNN yang ditentukan sebelumnya yaitu dengan arsitektur 1DCNN yang dirancang oleh Hung-Yu Chang dalam mendeteksi peristiwa *Sleep Apnea*[15]. Arsitektur Hung-Yu Chang terdiri atas 1 *input layer*, 10 *feature extraction layer*, 4 *classification (dense) layer*, dan 1 *output layer*. Diagram blok dari arsitektur 1DCNN yang dirancang oleh Hung-Yu Chang ditunjukkan pada Gambar III.13.



Gambar III.13 Diagram Blok Arsitektur 1DCNN Hung-Yu Chang[15]

Merujuk pada Gambar III.13, *input layer* pada arsitektur 1DCNN yang akan digunakan adalah sinyal ECG hasil tahap *preprocessing* dengan durasi satu menit, dengan demikian sesuai dengan keluaran tahap *train-test-split* yakni *array* berukuran 15600x6000x1 untuk *imbalanced dataset* dan 12310x6000x1 untuk *balanced dataset*. Pada bagian *feature extraction layer*, terdapat *convolutional layer, batch normalization layer, activation layer, max pooling layer,* dan *dropout layer*. Pada *convolutional layer* digunakan konvolusi 1D dengan parameter sebagai berikut: ukuran *filter* = 45, ukuran *kernel* = 32, jenis *padding = 'same', kernel initializer* = '*he_normal*'. *Activation layer* digunakan *activaiton function*= Rectified Linear Unit (ReLu). *Max pooling layer* digunakan parameter *pool size* = 2, dan *strides* = 2. Terakhir, *Droupout layer* digunakan dengan *dropout rate* = 0,5. Pada bagian *classification* (*dense layer*), digunakan parameter sebagai berikut: *dense layer* dengan 512 neuron, dan *kernel initializer* = '*he_normal*', *batch normalization layer, activation layer* dengan *activation function* = Rectified Linear Unit (ReLu), dan *dropout layer* dengan *dropout rate* = 0,5. Pada bagian *output layer* digunakan *dense layer* dengan dua neuron dengan fungsi aktivasi *Softmax*. Akan tetapi, untuk klasifikasi biner seperti yang dilakukan pada Tugas Akhir ini, pada bagian *output layer* digunakan dense layer dengan satu neuron dengan fungsi aktivasi *Sigmoid*. Tabel III.1 menunjukkan arsitektur 1DCNN yang digunakan dalam Tugas Akhir ini.

Tabel III.1 Arsitektur 1DCNN yang Digunakan pada Tugas Akhir

Layer	**Parameter**	**Ukuran *Output***
Feature Extraction Layer 1		(None, 6000, 1)
Conv. Layer	*filter* = 45, *kernel size* =32, *padding* = 'same', *kernel initializer* ='he_normal'	
Batch Normalization		(None, 6000, 45)
Activation Layer	ReLu	(None, 6000, 45)
Max Pooling Layer	*Pool size = 2, strides = 2*	(None, 3000, 45)
Dropout Layer	*Dropout rate = 0.5*	(None, 3000, 45)

…		
Feature Extraction Layer 10		
Conv. Layer	*filter* = 45, *kernel size* =32, *padding* = ‘same’, *kernel initializer* =‘he_normal’	(None, 11, 45)
Batch Normalization		(None, 11, 45)
Activation Layer	ReLu	(None, 11, 45)
Max Pooling Layer	*Pool size = 2, strides = 2*	(None, 5, 45)
Dropout Layer	*Dropout rate = 0.5*	(None, 5, 45)
Flatten		(None, 225)
Classification Layer 1		
Fully Connected Layer	*Units* = 512, *kernel initializer* = ‘he_normal’	(None, 512)
Batch Normalization		(None, 512)
Activation Layer	ReLu	
Dropout Layer	*Dropout rate* = 0.5	(None, 512)
…		
Classification Layer 4		
Fully Connected Layer	*Units* = 512, *kernel initializer* = ‘he_normal’	(None, 512)
Batch Normalization		(None, 512)
Activation Layer	ReLu	
Dropout Layer	*Dropout rate* = 0.5	(None, 512)
Fully Connected Layer	Sigmoid	(None, 1)

Setelah melewati tahap pelatihan, keluaran dari *output layer* pada arsitektur 1DCNN merupakan nilai probabilitas sampel termasuk ke dalam kelas 1 (peristiwa Apnea). Apabila nilai probabilitas melebihi 0.5, maka sampel tersebut termasuk ke dalam kelas 1 (peristiwa Apnea) dan sebaliknya. Tiap arsitektur akan diuji dengan data uji yang sudah dibagi sebelumnya pada tahap *train-test-split*. Performa arsitektur akan dinilai dari hasil prediksi terhadap *metrics* yang digunakan, antara

lain: akurasi, sensitivitas (*recall*), spesifisitas, presisi, dan skor F-1. Algoritma pembelajaran ini akan dilakukan untuk seluruh data latih sebanyak 200 kali (200 *epoch*), dengan parameter *patience* 30 *epoch*, sehingga jika dalam 30 *epoch* terakhir model tidak memberikan perubahan yang signifikan maka tahap *data training* akan berhenti. Berikut adalah potongan kode program untuk tahap *data training and testing*.

```
import tensorflow as tf
from tensorflow.keras.layers import Flatten, Dense, Conv1D,
MaxPool1D, Dropout, Input
from tensorflow.keras import Model
from tensorflow.keras.utils import plot_model
from sklearn.metrics import confusion_matrix, accuracy_score,
precision_score, recall_score, f1_score
import imblearn
import seaborn as sns

model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['
accuracy'])
model.summary()
'Checkpoint Callback'
'Jika lebih dari patience dan tidak ada perubahan signifikan, data
train stop'
checkpoint_cb =
tf.keras.callbacks.ModelCheckpoint("DL/tes_model.h5",save_best_only=
True)
checkpoint_cb_csv
=  tf.keras.callbacks.CSVLogger("DL/tes_model.csv")
early_stopping_cb =
tf.keras.callbacks.EarlyStopping(patience=30,restore_best_weights=Tr
ue)

#fit the model - Train Data
history =
model.fit(x=X_train,y=y_train,epochs=200,validation_data=(X_val,y_va
l),callbacks=[checkpoint_cb,early_stopping_cb,checkpoint_cb_csv])
model.save("DL/tes_model.h5")

'use this if u wanna load trained data...'
# saved_model = tf.keras.models.load_model('DL/tes_model.h5')

'Predict your data using test data'
my_predict = model.predict(X_test)
print('mypredict_ :  ' , my_predict)
```

```
my_predict_2 = my_predict.copy()
my_predict_2[my_predict_2>=0.5]=1
my_predict_2[my_predict_2<0.5]=0
np.savetxt('my_predict_2.csv', my_predict_2, delimiter=',')

'Plot metrics'
accuracy = accuracy_score(y_test, my_predict_2)
precision = precision_score(y_test, my_predict_2)
recall = recall_score(y_test, my_predict_2)
f1 =  f1_score(y_test, my_predict_2)
sensitivity = imblearn.metrics.sensitivity_score(y_test,
my_predict_2)
specificity = imblearn.metrics.specificity_score(y_test,
my_predict_2)
print('Accuracy: %.3f ' % (accuracy*100))
print('Precision: %.3f  ' % (precision*100))
print('Recall: %.3f  ' % (recall*100))
print('F1 Score: %.3f  ' % (f1*100))
print('Sensitivity: %.3f ' % (sensitivity*100))
print('Specificity: %.3f  ' % (specificity*100))
```

III.2.1 Variasi Perancangan *Arsitektur 1DCNN*

Pada Tugas Akhir akan dilakukan dengan delapan variasi arsitektur 1DCNN. Adapun variasi parameter secara detail disajikan melalui Tabel III.2.

Tabel III.2 Variasi Parameter dari Arsitektur 1DCNN Hung-Yu Chang[15]

Tipe Arsitektur	**Jumlah *Feature Extraction Layer***	**Jumlah *Filter* pada *Convolutional Layer***	**Jumlah *Classification Layer***
Arsitektur 1 (Hung-Yu Chang)	10	45	4
Arsitektur 2	10	45	6
Arsitektur 3	12	45	4
Arsitektur 4	12	45	6
Arsitektur 5	10	60	4
Arsitektur 6	10	60	6

Arsitektur 7	12	60	4
Arsitektur 8	12	60	6

Merujuk pada Tabel III.2, parameter yang tidak divariasikan tidak dicantumkan pada tabel tersebut, sehingga untuk parameter yang tidak divariasikan dapat dilihat secara lengkap pada Tabel III.1. Arsitektur 1 yang dimaksud pada Tabel III.2 merupakan arsitektur 1DCNN yang seluruh parameternya diuraikan pada Tabel III.1.

BAB IV
DATA PENGAMATAN DAN ANALISIS

Pada bab ini dijelaskan data pengamatan dan analisis dari hasil Tugas Akhir. Analisis dari data pengamatan dilakukan berdasarkan kurva perkembangan hasil pelatihan dan validasi, serta nilai *metrics* yang dihasilkan dari hasil prediksi yang dilakukan model terhadap data uji dengan menggunakan delapan variasi arsitektur 1DCNN yang telah dirancang sebelumnya. Kedelapan variasi arsitektur 1DCNN yang dimaksud diuraikan pada Tabel III.2. Secara umum, nilai *metrics* yang dihasilkan memiliki rentang nilai 0 hingga 1, artinya, semakin nilai *metrics* mendekati 1, maka model memiliki performa yang semakin baik dalam mendeteksi *Sleep Apnea*. Adapun juga akan dianalisa pengaruh dataset tidak seimbang (*imbalanced dataset*) dan dataset seimbang (*balanced dataset*) terhadap nilai *metrics* dan kurva perkembangan hasil pelatihan dan validasi. Dalam mencapai tujuan dari Tugas Akhir untuk mendeteksi *Sleep Apnea*, maka dari kelima *metrics* yang digunakan, akan dibandingkan nilai skor F-1 dan nilai akurasi dari tiap variasi arsitektur 1DCNN yang dirancang dan *input* yang digunakan.

IV.1 *Imbalanced Dataset*

Terdapat 15600 sampel yang terdapat pada *imbalanced dataset*, dengan 9535 sampel Normal dan 6065 sampel Apnea. Hasil prediksi model dengan delapan variasi arsitektur 1DCNN dengan dua variasi *input* berupa IMF 1 dan IMF 2 disajikan pada Tabel IV.1.

Tabel IV.1 Hasil *Metrics* dari *Imbalanced Dataset*

Tipe Arsitektur	*Metrics*				
	Akurasi (%)	Presisi (%)	Sensitivitas/*Recall* (%)	Skor F1 (%)	Spesifisitas (%)
Arsitektur 1					
Input: IMF1	**91.06**	**89.05**	**87.80**	**88.42**	**93.13**

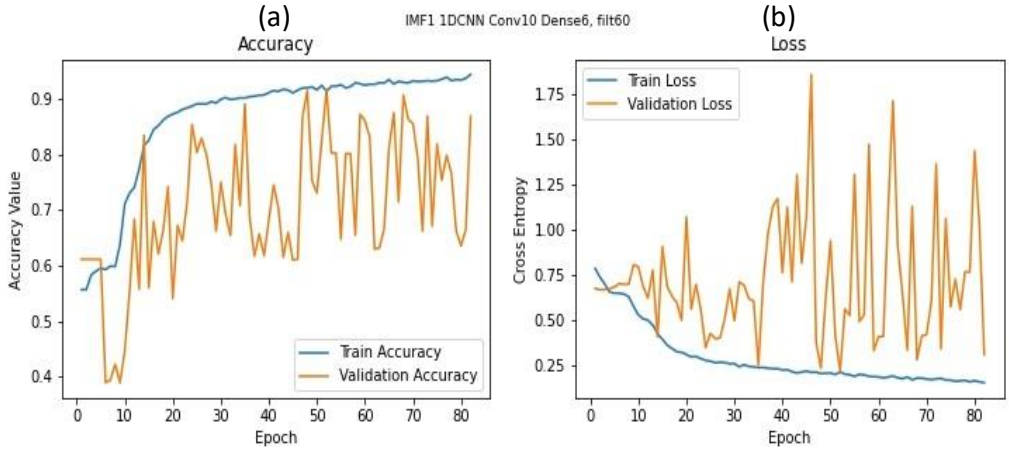
Input: IMF2	**90.58**	**88.78**	**86.73**	**87.74**	**93.03**
Rata-rata	**90.82**	**88.92**	**87.27**	**88.08**	**93.08**
Arsitektur 2					
Input: IMF1	**91.25**	**90.10**	**87.06**	**88.55**	**93.92**
Input: IMF2	**90.51**	**87.67**	**87.96**	**87.82**	**92.13**
Rata-rata	**90.88**	**88.89**	**87.51**	**88.19**	**93.03**
Arsitektur 3					
Input: IMF1	**90.26**	**88.48**	**86.15**	**87.30**	**92.87**
Input: IMF2	**91.54**	**88.67**	**89.69**	**89.18**	**92.71**
Rata-rata	**90.90**	**88.58**	**87.92**	**88.24**	**92.79**
Arsitektur 4					
Input: IMF1	**90.32**	**93.26**	**80.96**	**86.67**	**96.28**
Input: IMF2	**90.06**	**90.35**	**83.35**	**86.71**	**94.34**
Rata-rata	**90.19**	**91.81**	**82.16**	**86.69**	**95.31**
Arsitektur 5					
Input: IMF1	**91.41**	**89.67**	**88.05**	**88.85**	**93.55**
Input: IMF2	**90.19**	**93.82**	**80.05**	**86.39**	**96.64**
Rata-rata	**90.80**	**91.75**	**84.05**	**87.62**	**95.10**
Arsitektur 6					
Input: IMF1	**92.15**	**91.16**	**88.38**	**89.74**	**94.55**
Input: IMF2	**88.34**	**89.91**	**78.57**	**83.84**	**94.39**
Rata-rata	**90.25**	**90.54**	**83.48**	**86.79**	**94.47**
Arsitektur 7					
Input: IMF1	**89.52**	**91.48**	**80.54**	**85.66**	**95.23**
Input: IMF2	**91.06**	**89.05**	**87.80**	**88.42**	**93.13**
Rata-rata	**90.29**	**90.27**	**84.17**	**87.04**	**94.18**
Arsitektur 8					
Input: IMF1	**91.03**	**92.84**	**83.35**	**87.84**	**95.91**
Input: IMF2	**87.44**	**84.18**	**83.35**	**83.76**	**90.04**
Rata-rata	**89.24**	**88.51**	**83.35**	**85.80**	**92.98**
Rata-rata *Imbalanced Dataset*	**90.42**	**89.90**	**84.99**	**87.31**	**93.87**

Merujuk pada Tabel IV.1, dari delapan variasi arsitektur 1DCNN dan dua variasi *input* berupa IMF 1 dan IMF 2, selalu didapatkan nilai sensitivitas yang lebih kecil dibandingkan nilai spesifisitas. Hal ini disebabkan oleh karena *dataset* yang

digunakan bersifat tidak seimbang (*imbalanced*). Sampel Normal yang lebih banyak dibanding sampel Apnea menyebabkan model lebih banyak memprediksi sampel Normal dengan benar dibandingkan sampel Apnea, sehingga nilai spesifisitas selalu lebih besar dibanding sensitivitas untuk seluruh kombinasi arsitektur dan *input*. Hasil *metrics* terbaik dengan *imbalanced dataset* dengan *input* IMF 1 dihasilkan oleh Arsitektur 6, hal tersebut ditunjukkan dengan nilai akurasi mencapai 92.15% dan skor F-1 mencapai 89.74. Adapun hasil *metrics* terbaik dengan *imbalanced dataset* dengan *input* IMF 2 dihasilkan oleh Arsitektur 3, dengan nilai akurasi 91.54% dan skor F-1 sebesar 89.18%. Dengan *imbalanced dataset*, diperoleh nilai rata-rata akurasi dari seluruh kombinasi variasi arsitektur 1DCNN dan variasi *input* sebesar 90.42% dan nilai skor F-1 sebesar 87.31%.

IV.1.1 Kombinasi Arsitektur 6 dengan *Input* IMF 1 dari *Imbalanced Dataset*

Arsitektur 6 dirancang dengan 10 *feature extraction layer*, 60 *filter* pada *convolutional layer*, dan 6 *classification layer*. Dengan menggunakan *imbalanced dataset*, Arsitektur 6 memberikan hasil rata-rata *metrics* terbesar dengan *input* berupa IMF 1, yakni mencapai 91.20%. Berikut ini adalah kurva perkembangan hasil pelatihan dan validasi dari Arsitektur 6 dengan *input* IMF 1 dari *imbalanced dataset*.

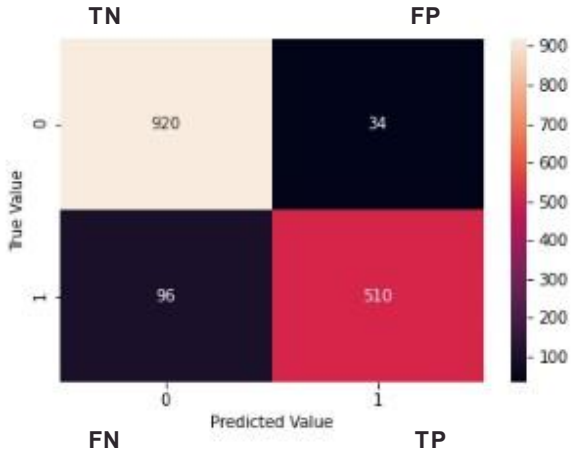


Gambar IV.1 Kurva Perkembangan Pelatihan dan Validasi
Arsitektur 6 dengan *Input* IMF 1 dari *Imbalanced Dataset*: (a) akurasi; (b) *loss*

Merujuk pada Gambar IV.1, gambar (a) menunjukkan nilai akurasi, dan (b) menunjukkan nilai *loss*. Pada kombinasi Arsitektur 6 dan *input* IMF 1, pelatihan dilakukan sebanyak 86 iterasi (*epoch*). Seiring bertambahnya iterasi, didapati nilai kurva nilai akurasi pelatihan bertambah besar, sedangkan kurva nilai *loss* pelatihan bertambah kecil. Hal ini menunjukkan upaya model dalam mempelajari data pelatihan untuk melakukan prediksi.

Namun, dari hasil yang didapat pada Gambar IV.1, ditunjukkan bahwa kurva nilai akurasi validasi dan nilai *loss* validasi memiliki nilai yang berfluktuasi. Semakin banyak iterasi yang dilakukan, fluktuasi nilai akurasi validasi dan nilai *loss* validasi semakin besar, sehingga pada *epoch* tertentu nilai akurasi validasi tinggi tetapi pada *epoch* lainnya nilai akurasi validasi rendah. Begitupun untuk nilai *loss* validasi, pada *epoch* tertentu memiliki nilai yang rendah hingga mendekati nilai *loss* pelatihan, tetapi pada *epoch* berikutnya nilainya menjadi tinggi dan selanjutnya terus berfluktuasi hingga *epoch* terakhir. Hal tersebut dapat terjadi akibat pembobotan yang awalnya diberikan untuk *epoch* yang satu memberikan nilai validasi yang baik, akan tetapi apabila diberikan nilai pembobotan yang mendekati nilai pembobotan awal kepada *epoch* selanjutnya, justru memberikan nilai validasi yang buruk, artinya nilai akurasi validasi kecil dan nilai *loss* validasi besar.

Adapun hasil prediksi model terhadap data uji disajikan melalui *confusion matrix* pada Gambar IV.2 sebagai berikut.



Gambar IV.2 *Confusion Matrix* Arsitektur 6 dengan *Input* IMF 1 dari *Imbalanced Dataset*

Keterangan:

TP (*True Positive*) : Hasil prediksi positif (Apnea) dan pada kenyataannya data positif (Apnea), sehingga prediksi benar.

TN (*True Negative*) : Hasil prediksi negatif (Normal) dan pada kenyataannya data negatif (Normal), sehingga prediksi benar.

FP (*False Positive*) : Hasil prediksi positif (Apnea), tetapi pada kenyataannya data negatif (Normal), sehingga prediksi salah.

FN (*False Negative*) : Hasil prediksi negatif (Normal), tetapi pada kenyataannya data positif (Apnea), sehingga prediksi salah.

Label 0 : Normal

Label 1 : Apnea

Dari *confusion matrix* yang disajikan pada Gambar IV.2, ditunjukkan hasil prediksi model dalam empat kelas, TP, TN, FP, dan FN. Dengan keempat jumlah sampel dalam kelas tersebut akan ditentukan lima *metrics* yang digunakan dalam mengukur performa model, yakni: akurasi, presisi, sensitivitas, spesifisitas, dan skor F-1. Adapun kelima nilai *metrics* dari kombinasi Arsitektur 6 dan *input* IMF 1, disajikan dalam Tabel IV. 2.

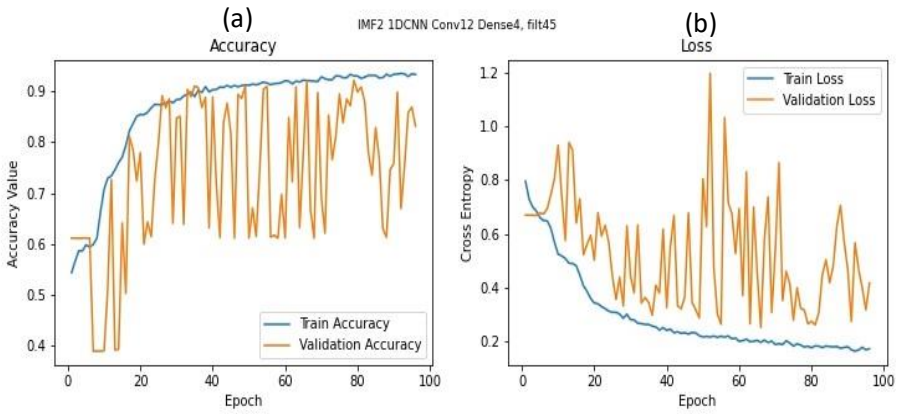
Tabel IV.2 Nilai *Metrics* dari Arsitektur 6 dengan *Input* IMF 1 dari *Imbalanced Dataset*

Tipe Arsitektur	***Metrics***				
	Akurasi (%)	**Presisi (%)**	**Sensitivitas/*Recall* (%)**	**Skor F1 (%)**	**Spesifisitas (%)**
Arsitektur 6					
Input: IMF1	**92.15**	**91.16**	**88.38**	**89.74**	**94.55**

IV.1.2 Kombinasi Arsitektur 3 dengan *Input* IMF 2 dari *Imbalanced Dataset*

Arsitektur 3 dirancang dengan 12 *feature extraction layer*, 45 *filter* pada *convolutional layer*, dan empat *classification layer*. Dengan menggunakan *imbalanced dataset*, Arsitektur 3 memberikan hasil rata-rata *metrics* terbesar dengan *input* berupa IMF 2, yakni mencapai 90.31%. Namun, setelah dikaji dengan seluruh variasi tipe arsitektur, Arsitektur 3 memberikan nilai *metrics* terbaik dengan

input IMF 2, dengan nilai akurasi mencapai 91.54%, dan nilai skor F-1 mencapai 89.18%. sehingga dari kedelapan variasi arsitektur, Arsitektur 3 memberikan performa yang paling baik apabila digunakan *input* berupa IMF 2 dari *imbalanced dataset*. Berikut ini adalah kurva perkembangan hasil pelatihan dan validasi dari Arsitektur 3 dengan *input* IMF 2 dari *imbalanced dataset*.



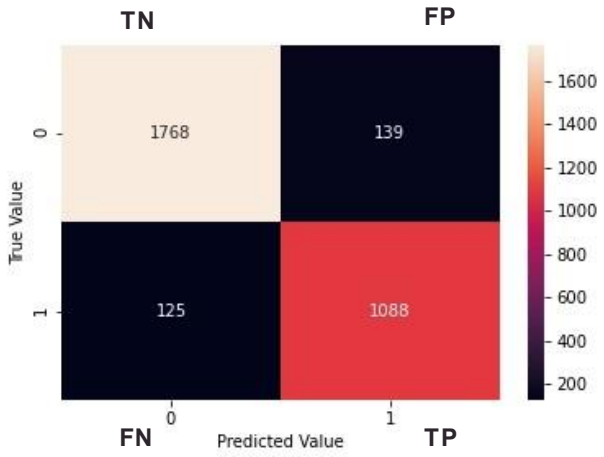
Gambar IV.3 Kurva Perkembangan Pelatihan dan Validasi
Arsitektur 3 dengan *Input* IMF 2 dari *Imbalanced Dataset*: (a) akurasi; (b) *loss*

Merujuk pada Gambar IV.3, gambar (a) menunjukkan nilai akurasi, dan (b) menunjukkan nilai *loss*. Pada kombinasi Arsitektur 3 dan *input* IMF 2, pelatihan dilakukan sebanyak 108 iterasi (*epoch*). Seiring bertambahnya *epoch*, didapati nilai kurva nilai akurasi pelatihan bertambah besar, sedangkan kurva nilai *loss* pelatihan bertambah kecil. Hal ini menunjukkan upaya model dalam mempelajari data pelatihan untuk melakukan prediksi.

Namun, dari hasil yang didapat pada Gambar IV.3, ditunjukkan bahwa kurva nilai akurasi validasi dan nilai *loss* validasi memiliki nilai yang berfluktuasi. Semakin banyak iterasi yang dilakukan, fluktuasi nilai akurasi validasi dan nilai *loss* validasi semakin besar, sehingga pada *epoch* tertentu nilai akurasi validasi tinggi tetapi pada *epoch* lainnya nilai akurasi validasi rendah. Begitupun untuk nilai *loss* validasi, pada *epoch* tertentu memiliki nilai yang rendah hingga mendekati nilai *loss* pelatihan, tetapi pada *epoch* berikutnya nilainya menjadi tinggi dan selanjutnya terus berfluktuasi hingga *epoch* terakhir. Pada kombinasi Arsitektur 3 dengan *input* IMF 2 terjadi fluktuasi nilai *loss* validasi terbesar pada rentang *epoch* 40 hingga *epoch* 60, dimana nilai *loss* validasi berubah dari rentang nilai *cross entropy* 0.3 hingga ke 1.2 kemudian turun kembali ke rentang nilai 0.3. Hal tersebut dapat

terjadi akibat pembobotan yang awalnya diberikan untuk *epoch* yang satu memberikan nilai validasi yang baik, akan tetapi apabila diberikan nilai pembobotan yang mendekati nilai pembobotan awal kepada *epoch* selanjutnya, justru memberikan nilai validasi yang buruk, artinya nilai akurasi validasi kecil dan nilai *loss* validasi besar.

Adapun hasil prediksi model terhadap data uji disajikan melalui *confusion matrix* pada Gambar IV.4 sebagai berikut.



Gambar IV.4 *Confusion Matrix* Arsitektur 3 dengan *Input* IMF 2 dari *Imbalanced Dataset*

Keterangan:

TP (*True Positive*) : Hasil prediksi positif (Apnea) dan pada kenyataannya data positif (Apnea), sehingga prediksi benar.

TN (*True Negative*) : Hasil prediksi negatif (Normal) dan pada kenyataannya data negatif (Normal), sehingga prediksi benar.

FP (*False Positive*) : Hasil prediksi positif (Apnea), tetapi pada kenyataannya data negatif (Normal), sehingga prediksi salah.

FN (*False Negative*) : Hasil prediksi negatif (Normal), tetapi pada kenyataannya data positif (Apnea), sehingga prediksi salah.

Label 0 : Normal

Label 1 : Apnea

Dari *confusion matrix* yang disajikan pada Gambar IV.4, ditunjukkan hasil prediksi model dalam empat kelas, TP, TN, FP, dan FN. Dengan keempat jumlah sampel dalam kelas tersebut akan ditentukan lima *metrics* yang digunakan dalam

mengukur performa model, yakni: akurasi, presisi, sensitivitas, spesifisitas, dan skor F-1.

Adapun kelima nilai *metrics* dari kombinasi Arsitektur 3 dan *input* IMF 2 dan IMF 2 dari *imbalanced dataset* disajikan pada Tabel IV.3.

Tabel IV.3 Nilai Metrics dari Arsitektur 3 dengan *Input* 2 dari *Imbalanced Dataset*

Tipe Arsitektur	*Metrics*				
	Akurasi (%)	Presisi (%)	Sensitivitas/*Recall* (%)	Skor F1 (%)	Spesifisitas (%)
Arsitektur 3					
Input: IMF2	**91.54**	**88.67**	**89.69**	**89.18**	**92.71**

IV.2 *Balanced Dataset*

Terdapat 12130 sampel yang terdapat pada *balanced dataset*, dengan 6065 sampel Normal dan 6065 sampel Apnea. Hasil prediksi model dengan delapan variasi arsitektur 1DCNN dengan dua variasi *input* berupa IMF 1 dan IMF 2 disajikan pada Tabel IV.4.

Tabel IV.4 Hasil *Metrics* dari *Balanced Dataset*

Tipe Arsitektur	*Metrics*				
	Akurasi (%)	Presisi (%)	Sensitivitas/*Recall* (%)	Skor F1 (%)	Spesifisitas (%)
Arsitektur 1					
Input: IMF1	**91.59**	**92.14**	**90.92**	**91.53**	**92.57**
Input: IMF2	**92.58**	**93.14**	**91.91**	**92.52**	**93.24**
Rata-rata	**92.09**	**92.64**	**91.42**	**92.03**	**92.91**
Arsitektur 2					
Input: IMF1	**92.91**	**93.49**	**92.24**	**92.86**	**93.57**
Input: IMF2	**92.58**	**97.08**	**87.79**	**92.2**	**97.36**
Rata-rata	**92.75**	**95.29**	**90.02**	**92.53**	**95.47**
Arsitektur 3					
Input: IMF1	**91.59**	**92.86**	**90.1**	**91.46**	**93.08**

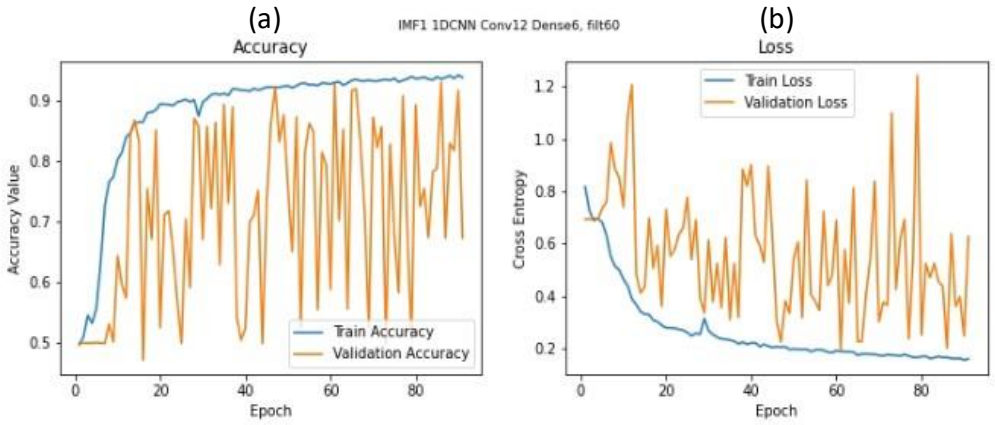
Input: IMF2	**91.92**	**89.81**	**94.56**	**92.12**	**89.29**
Rata-rata	**91.76**	**91.34**	**92.33**	**91.79**	**91.19**
Arsitektur 4					
Input: IMF1	**87.47**	**82.15**	**95.71**	**88.41**	**79.24**
Input: IMF2	**92**	**92.63**	**91.25**	**91.94**	**92.75**
Rata-rata	**89.74**	**87.39**	**93.48**	**90.18**	**86.00**
Arsitektur 5					
Input: IMF1	**90.68**	**93.02**	**87.95**	**90.42**	**93.41**
Input: IMF2	**92.5**	**94.63**	**90.1**	**92.31**	**94.89**
Rata-rata	**91.59**	**93.83**	**89.03**	**91.37**	**94.15**
Arsitektur 6					
Input: IMF1	**92.25**	**94.14**	**90.1**	**92.07**	**94.4**
Input: IMF2	**91.18**	**90.31**	**92.24**	**91.26**	**90.11**
Rata-rata	**91.72**	**92.23**	**91.17**	**91.67**	**92.26**
Arsitektur 7					
Input: IMF1	**89.86**	**85.57**	**95.87**	**90.43**	**83.86**
Input: IMF2	**91.06**	**94.62**	**87.13**	**90.72**	**95.06**
Rata-rata	**90.46**	**90.10**	**91.50**	**90.58**	**89.46**
Arsitektur 8					
Input: IMF1	**93.24**	**94.11**	**92.24**	**93.17**	**94.23**
Input: IMF2	**90.6**	**88.2**	**93.73**	**90.88**	**87.48**
Rata-rata	**91.92**	**91.16**	**92.99**	**92.03**	**90.86**
Rata-rata *Balanced Dataset*	**91.36**	**91.62**	**90.00**	**90.72**	**92.31**

Merujuk pada Tabel IV.4, dari delapan variasi arsitektur 1DCNN dan dua variasi *input* berupa IMF 1 dan IMF 2, terdapat variasi perbandingan nilai sensitivitas dengan nilai spesifisitas, pada kombinasi arsitektur dan *input* tertentu didapati nilai sensitivitas yang lebih tinggi. Selain itu, dengan menggunakan *balanced dataset* perbedaan nilai sensitivitas dan spesifisitas dari seluruh kombinasi arsitektur dan *input* yang ada, memiliki perbedaan yang tidak terlalu besar, sedangkan dengan *imbalanced dataset* didapati perbedaan mendekati 20%. Hal tersebut disebabkan oleh karena sampel Normal dan Apnea yang seimbang. Hasil *metrics* terbaik dengan *input* IMF 1 dari *balanced dataset*, didapat dengan Arsitektur 8, sedangkan *input* IMF 2 dari *balanced dataset*, didapat dengan

Arsitektur 2. Hal tersebut ditunjukkan dengan nilai akurasi dan nilai skor F-1 untuk variasi *input* berupa IMF 1 dengan Arsitektur 8 didapat sebesar 92.91 dan 92.86%. Adapun hasil terbaik dari *balanced* dataset dengan *input* IMF 2 memperoleh nilai akurasi sebesar 93.24% dan nilai skor F-1 sebesar 93.17%. Dengan *balanced dataset* diperoleh nilai rata-rata *metrics* dari seluruh kombinasi variasi arsitektur 1DCNN dan variasi *input* sebesar 91.53%.

IV.2.1 Kombinasi Arsitektur 8 dengan *Input* IMF 1 dari *Balanced Dataset*

Arsitektur 8 dirancang dengan 12 *feature extraction layer*, 60 *filter* pada *convolutional layer*, dan 6 *classification layer*. Dengan menggunakan *balanced dataset*, Arsitektur 8 dengan *input* IMF 1 memberikan hasil nilai akurasi terbesar yakni 93.24%, dan nilai skor F-1 sebesar 93.17%. Berikut adalah kurva perkembangan hasil pelatihan dan validasi dari Arsitektur 8 dengan *input* IMF 1 dari *imbalanced dataset*.

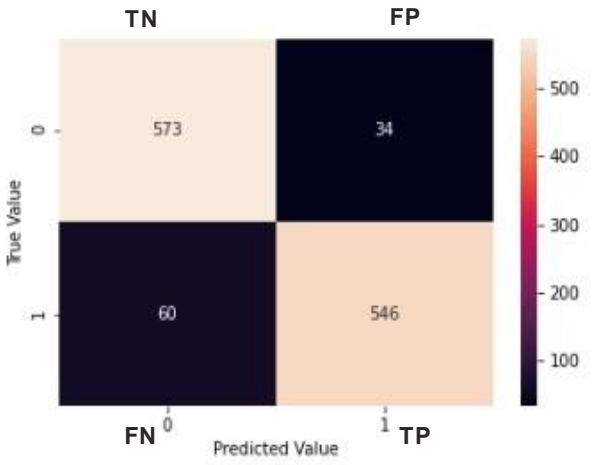


Gambar IV.5 Kurva Perkembangan Pelatihan dan Validasi
Arsitektur 8 dengan *input* IMF 1 dari *balanced dataset*: (a) akurasi; (b) *loss*

Merujuk pada Gambar IV.5, gambar (a) menunjukkan nilai akurasi, dan (b) menunjukkan nilai *loss*. Pada kombinasi Arsitektur 8 dan *input* IMF 1, pelatihan dilakukan sebanyak 93 iterasi (*epoch*). Seiring bertambahnya iterasi, didapati nilai kurva nilai akurasi pelatihan bertambah besar, sedangkan kurva nilai *loss* pelatihan bertambah kecil. Hal ini menunjukkan upaya model dalam mempelajari data pelatihan untuk melakukan prediksi.

Namun, dari hasil yang didapat pada Gambar IV.5, ditunjukkan bahwa kurva nilai akurasi validasi dan nilai *loss* validasi memiliki nilai yang berfluktuasi. Semakin banyak iterasi yang dilakukan, fluktuasi nilai akurasi validasi dan nilai *loss* validasi semakin besar, sehingga pada *epoch* tertentu nilai akurasi validasi tinggi tetapi pada *epoch* lainnya nilai akurasi validasi rendah. Begitupun untuk nilai *loss* validasi, pada *epoch* tertentu memiliki nilai yang rendah hingga mendekati nilai *loss* pelatihan, tetapi pada *epoch* berikutnya nilainya menjadi tinggi dan selanjutnya terus berfluktuasi hingga *epoch* terakhir. Hal tersebut dapat terjadi akibat pembobotan yang awalnya diberikan untuk *epoch* yang satu memberikan nilai validasi yang baik, akan tetapi apabila diberikan nilai pembobotan yang mendekati nilai pembobotan awal kepada *epoch* selanjutnya, justru memberikan nilai validasi yang buruk, artinya nilai akurasi validasi kecil dan nilai *loss* validasi besar.

Adapun hasil prediksi model terhadap data uji disajikan melalui *confusion matrix* pada Gambar IV.6 sebagai berikut.



Gambar IV.6 *Confusion Matrix* Arsitektur 8 dengan *Input* IMF 1 dari *Balanced Dataset*

Keterangan:

TP (*True Positive*) : Hasil prediksi positif (Apnea) dan pada kenyataannya data positif (Apnea), sehingga prediksi benar.

TN (*True Negative*) : Hasil prediksi negatif (Normal) dan pada kenyataannya data negatif (Normal), sehingga prediksi benar.

FP (*False Positive*) : Hasil prediksi positif (Apnea), tetapi pada kenyataannya data negatif (Normal), sehingga prediksi salah.

FN (*False Negative*) : Hasil prediksi negatif (Normal), tetapi pada kenyataannya data positif (Apnea), sehingga prediksi salah.

Label 0 : Normal

Label 1 : Apnea

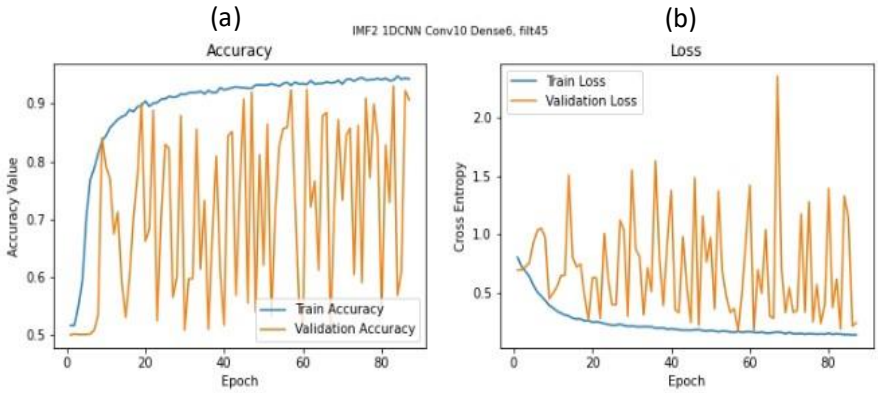
Dari *confusion matrix* yang disajikan pada Gambar IV.6, ditunjukkan hasil prediksi model dalam empat kelas, TP, TN, FP, dan FN. Dengan keempat jumlah sampel dalam kelas tersebut akan ditentukan lima *metrics* yang digunakan dalam mengukur performa model, yakni: akurasi, presisi, sensitivitas, spesifisitas, dan skor F-1. Adapun kelima nilai *metrics* dari kombinasi Arsitektur 8 dan *input* IMF 1, disajikan dalam Tabel IV.5.

Tabel IV.5 Nilai Metrics dari Arsitektur 8 dengan *Input* IMF 1 dari *Balanced Dataset*

Tipe Arsitektur	***Metrics***				
	Akurasi (%)	**Presisi (%)**	**Sensitivitas/*Recall* (%)**	**Skor F1 (%)**	**Spesifisitas (%)**
Arsitektur 8					
Input: IMF1	**93.24**	**94.11**	**92.24**	**93.17**	**94.23**

IV.2.2 Kombinasi Arsitektur 2 dengan *Input* IMF 2 dari *Balanced Dataset*

Arsitektur 2 dirancang dengan 10 *feature extraction layer*, 45 *filter* pada *convolutional layer*, dan 6 *classification layer*. Dengan menggunakan *balanced dataset*, Arsitektur 2 memberikan hasil performa terbaik dengna *input* IMF 2, yakni mencapai nilai akurasi sebesar 92.58% dan nilai skor F-1 sebesar 92.20%. Namun, setelah dikaji dengan seluruh variasi tipe arsitektur, Arsitektur 2 memberikan performa yang paling baik apabila memproses *input* berupa IMF 2. Berikut ini adalah kurva perkembangan hasil pelatihan dan validasi dari Arsitektur 2 dengan *input* IMF 2 dari *imbalanced dataset*.



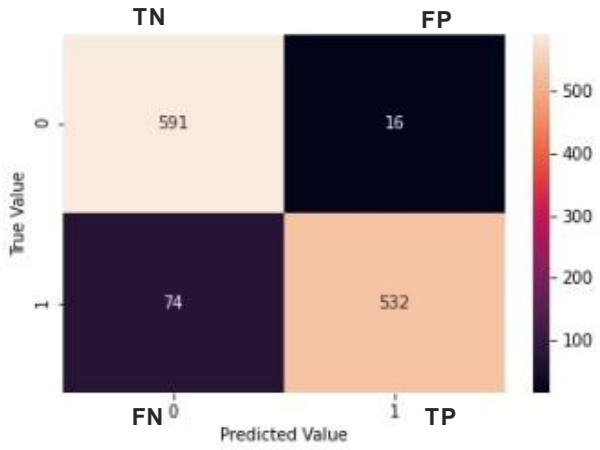
Gambar IV.7 Kurva Perkembangan Pelatihan dan Validasi
Arsitektur 2 dengan *Input* IMF 2 dari *Balanced Dataset*: (a) akurasi; (b) *loss*

Merujuk pada Gambar IV.7, gambar (a) menunjukkan nilai akurasi, dan (b) menunjukkan nilai *loss*. Pada kombinasi Arsitektur 2 dan *input* IMF 2, pelatihan dilakukan sebanyak 92 iterasi (*epoch*). Seiring bertambahnya *epoch*, didapati nilai kurva nilai akurasi pelatihan bertambah besar, sedangkan kurva nilai *loss* pelatihan bertambah kecil. Hal ini menunjukkan upaya model dalam mempelajari data pelatihan untuk melakukan prediksi.

Ditunjukkan bahwa kurva nilai akurasi validasi dan nilai *loss* validasi memiliki nilai yang berfluktuasi. Semakin banyak iterasi yang dilakukan, fluktuasi nilai akurasi validasi dan nilai *loss* validasi semakin besar, sehingga pada *epoch* tertentu nilai akurasi validasi tinggi tetapi pada *epoch* lainnya nilai akurasi validasi rendah. Begitupun untuk nilai *loss* validasi, pada *epoch* tertentu memiliki nilai yang rendah hingga mendekati nilai *loss* pelatihan, tetapi pada *epoch* berikutnya nilainya menjadi tinggi dan selanjutnya terus berfluktuasi hingga *epoch* terakhir.

Pada kombinasi Arsitektur 2 dengan *input* IMF 2 terjadi fluktuasi nilai *loss* validasi terbesar pada rentang *epoch* 60 hingga *epoch* 80, dimana nilai *loss* validasi berubah dari rentang nilai *cross entropy* 0.3 hingga ke 2.3 kemudian turun kembali ke rentang nilai 0.3. Hal tersebut dapat terjadi akibat pembobotan yang awalnya diberikan untuk *epoch* yang satu memberikan nilai validasi yang baik, akan tetapi apabila diberikan nilai pembobotan yang mendekati nilai pembobotan awal kepada *epoch* selanjutnya, justru memberikan nilai validasi yang buruk, artinya nilai akurasi validasi kecil dan nilai *loss* validasi besar.

Adapun hasil prediksi model terhadap data uji disajikan melalui *confusion matrix* pada Gambar IV.8 sebagai berikut.



Gambar IV.8 *Confusion Matrix* Arsitektur 2 dengan *Input* IMF 2 dari *Balanced Dataset*

Keterangan:

TP (*True Positive*) : Hasil prediksi positif (Apnea) dan pada kenyataannya data positif (Apnea), sehingga prediksi benar.

TN (*True Negative*) : Hasil prediksi negatif (Normal) dan pada kenyataannya data negatif (Normal), sehingga prediksi benar.

FP (*False Positive*) : Hasil prediksi positif (Apnea), tetapi pada kenyataannya data negatif (Normal), sehingga prediksi salah.

FN (*False Negative*) : Hasil prediksi negatif (Normal), tetapi pada kenyataannya data positif (Apnea), sehingga prediksi salah.

Label 0 : Normal

Label 1 : Apnea

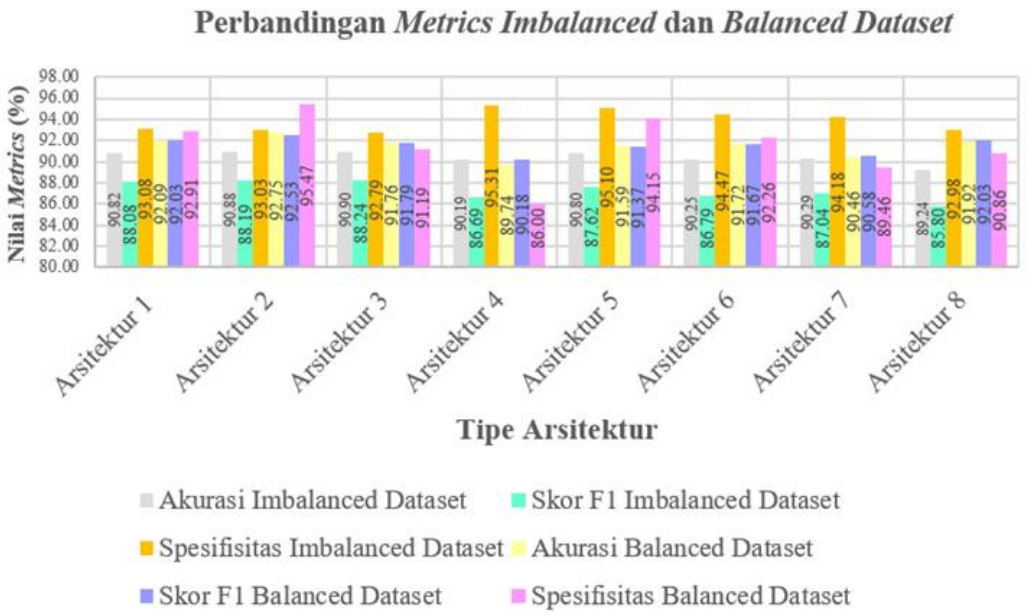
Dari *confusion matrix* yang disajikan pada Gambar IV.8, ditunjukkan hasil prediksi model dalam empat kelas, TP, TN, FP, dan FN. Dengan keempat jumlah sampel dalam kelas tersebut akan ditentukan lima *metrics* yang digunakan dalam mengukur performa model, yakni: akurasi, presisi, sensitivitas, spesifisitas, dan skor F-1. Adapun kelima nilai *metrics* dari kombinasi Arsitektur 2 dan *input* IMF 2 dari *balanced dataset* disajikan pada Tabel IV.6.

Tabel IV.6 Nilai *Metrics* dari Arsitektur 2 dengan *Input* IMF 2 dari *Balanced Dataset*

Tipe Arsitektur	*Metrics*				
	Akurasi (%)	Presisi (%)	Sensitivitas/*Recall* (%)	Skor F1 (%)	Spesifisitas (%)
Arsitektur 5					
Input: IMF2	**92.5**	**94.63**	**90.1**	**92.31**	**94.89**

IV.3 Perbandingan Performa Arsitektur 1DCNN dengan *Imbalanced Dataset* dan *Balanced Dataset*

Dalam Tugas Akhir ini akan dinilai perbandingan performa arsitektur 1DCNN dengan dua jenis *dataset* yang digunakan, yakni *imbalanced dataset* dan *balanced dataset*. Adapun perbandingan performa tersebut disajikan pada Gambar IV.9.



Gambar IV.9 Perbandingan Performa Arsitektur 1DCNN dengan *Imbalanced Dataset dan Balanced Dataset*

Merujuk pada Gambar IV.9 terlihat bahwa dengan *imbalanced dataset* diperoleh spesifisitas yang tinggi dibanding dengan *balanced dataset*, hal ini menunjukkan bahwa dengan *imbalanced dataset* terdapat jauh lebih banyak sampel Normal dibanding Apnea, sehingga memberikan nilai spesifisitas yang lebih tinggi

yakni dengan rentang nilai 92-95%. Jika melihat dari *metrics* akurasi, *balanced dataset* memberikan nilai yang lebih tinggi dibanding *imbalanced dataset* hampir untuk seluruh tipe arsitektur yang digunakan. Nilai akurasi *balanced dataset* berkisar pada rentang nilai 89-92%, sedangkan nilai akurasi *imbalanced dataset* berkisar pada rentang nilai 89-90%. Apabila dilihat dari *metrics* skor F-1, *balanced dataset* juga memberikan nilai yang lebih tinggi dengan rentang nilai 90-92% sedangkan *imbalanced dataset* memiliki rentang nilai 86-88%. Sehingga, dengan menggunakan *balanced dataset*, nilai *metrics* yang dihasilkan lebih baik dibanding dengan menggunakan *imbalanced dataset*, kecuali untuk nilai *metrics* spesifisitas. Karena tujuan dari Tugas Akhir adalah medeteksi *Sleep Apnea*, maka diinginkan model yang mampu melakukan prediksi Apnea lebih baik dibanding prediksi Normal, sehingga diinginkan model yang memiliki nilai spesifisitas yang lebih kecil dan nilai skor F-1 yang lebih besar.

BAB V
SIMPULAN DAN SARAN

V.1 Simpulan

Berikut adalah simpulan yang dapat diperoleh dari Tugas Akhir, antara lain sebagai berikut.

1. Deteksi *Sleep Apnea* berdasarkan sinyal Electrocardiogram (ECG) telah berhasil diimplementasikan dengan menggunakan metode One Dimensional Convolutional Neural Network (1DCNN).
2. Deteksi *Sleep Apnea* dilakukan dengan menggunakan dua jenis *dataset*; *balanced* dan *imbalanced dataset* dengan delapan jenis variasi arsitektur 1DCNN yang dirancang.
3. Dengan *input* berupa IMF 1 dari *imbalanced dataset*, Arsitektur 6 memberikan performa terbaik dibanding kombinasi variasi arsitektur dan *input* lainnya, dengan nilai akurasi sebesar 92.15% dan nilai skor F-1 sebesar 89.74%. Apabila digunakan *input* berupa IMF 2 dari *imbalanced dataset*, Arsitektur 3 memberikan performa terbaik, dengan nilai akurasi mencapai 91.54% dan nilai skor F-1 mencapai 89.18%.
4. Dengan *input* berupa IMF 1 dari *balanced dataset*, Arsitektur 8 memberikan performa terbaik dibanding kombinasi variasi arsitektur dan *input* lainnya, dengan nilai akurasi sebesar 93.24% dan nilai skor F-1 sebesar 93.17%. Apabila digunakan *input* berupa IMF 2 dari *balanced dataset*, Arsitektur 3 memberikan performa terbaik, dengan nilai akurasi mencapai 92.58% dan nilai skor F-1 mencapai 92.12%.
5. Jika dilihat dari jenis *dataset* yang digunakan, maka *balanced dataset* memberikan performa yang lebih baik dibandingkan dengan *imbalanced dataset*. Dengan setiap kombinasi arsitektur dan *input* yang menggunakan *balanced dataset* memberikan nilai rata-rata akurasi sebesar 91.36%, dan nilai rata-rata skor F-1 sebesar 90.72%, sedangkan *imbalanced dataset* hanya mampu memberikan nilai rata-rata akurasi sebesar 90.42%, dan nilai rata-rata skor F-1 sebesar 87.31%.

6. Jika dilihat dari kurva data pelatihan dan validasi, maka akan terlihat bahwa model yang berhasil dibentuk dari seluruh variasi arsitektur 1DCNN dan *input* serta variasi *dataset* yang digunakan, terjadi fluktuasi pada nilai akurasi validasi dan nilai *loss* validasi. Hal tersebut mengakibatkan kurva data pelatihan dan validasi tidak dapat menilai kemampuan model untuk memprediksi data uji.

V.2 Saran

Adapun saran yang diperoleh dari Tugas Akhir ini, antara lain sebagai berikut.

1. Merancang arsitektur 1DCNN dengan menerapkan arsitektur 1DCNN lainnya selain dari arsitektur yang dirancang oleh Hung-Yu Chang, dkk.[16] yang lebih cocok mengolah *input* berupa sinyal ECG.
2. Menggunakan metode *k-fold cross validation* dalam tahap validasi hasil tahap pelatihan sehingga mampu memberikan kurva pelatihan dan validasi yang lebih baik. Dengan kurva pelatihan dan validasi yang lebih baik, dapat dinilai kemampuan model dalam memprediksi data uji.
3. Hasil deteksi Apnea/Normal dapat dikembangkan untuk menentukan berbagai jenis penyakit Sleep Apnea.
4. Menggunakan metode preprocessing lainnya seperti *Ensemble Empirical Mode Decomposition* (EEMD) dan *Complete Ensemble Empirical Mode Decomposition with Adaptive Noise* (CEEMDAN)

DAFTAR REFERENSI

[1] X. M. Liu, Y. Lian, S. C. Chen, and S. S. Tang, "The design of multi-parameter and portable monitor for sleep apnea status," *Lect. Notes Electr. Eng.*, vol. 140 LNEE, pp. 583–588, 2012, doi: 10.1007/978-3-642-27296-7_89.

[2] M. Al-Mardini, F. Aloul, A. Sagahyroon, and L. Al-Husseini, "Classifying obstructive sleep apnea using smartphones," *J. Biomed. Inform.*, vol. 52, pp. 251–259, 2014, doi: 10.1016/j.jbi.2014.07.004.

[3] S. Brown, "Sleep apnea monitoring," *Ugeskr. Laeger*, vol. 154, no. 26, p. 1853, 2018.

[4] A. M. Osman, S. G. Carter, J. C. Carberry, and D. J. Eckert, "Obstructive sleep apnea: current perspectives," *Nat. Sci. Sleep*, vol. 10, pp. 21–34, 2018, doi: 10.2147/NSS.S124657.

[5] R. Mehra *et al.*, "Association of nocturnal arrhythmias with sleep-disordered breathing: The sleep heart health study," *Am. J. Respir. Crit. Care Med.*, vol. 173, no. 8, pp. 910–916, 2006, doi: 10.1164/rccm.200509-1442OC.

[6] A. Sheta *et al.*, "Diagnosis of obstructive sleep apnea from ECG signals using machine learning and deep learning classifiers," *Appl. Sci.*, vol. 11, no. 14, 2021, doi: 10.3390/app11146622.

[7] H. Korkalainen, *DEEP LEARNING FOR NEXT-GENERATION SLEEP DIAGNOSTICS : SOPHISTICATED COMPUTATIONAL METHODS FOR MORE EFFICIENT AND ACCURATE ASSESSMENT OF SLEEP AND OBSTRUCTIVE SLEEP APNEA*. Kuopio: University of Eastern Finland Faculty of Science and Forestry, Department of Applied Physics, 2020.

[8] R. Kher, "Signal Processing Techniques for Removing Noise from ECG Signals," *Jber*, vol. 3, pp. 1–9, 2019, doi: 10.17303/jber.2019.3.101.

[9] A. Chacko and S. Ari, "Denoising of ECG signals using Empirical Mode Decomposition based technique," *IEEE-International Conf. Adv. Eng. Sci. Manag. ICAESM-2012*, vol. 2, no. iii, pp. 6–9, 2012.

[10] H. Ge, G. Chen, H. Yu, H. Chen, and F. An, "Theoretical analysis of

empirical mode decomposition," *Symmetry (Basel).*, vol. 10, no. 11, 2018, doi: 10.3390/sym10110623.

[11] J. Corthout, S. Van Huffel, M. O. Mendez, A. M. Bianchi, T. Penzel, and S. Cerutti, "Automatic screening of Obstructive Sleep Apnea from the ECG based on Empirical Mode Decomposition and Wavelet Analysis Automatic screening of Obstructive Sleep Apnea from the ECG based on Empirical Mode Decomposition and Wavelet Analysis," no. February 2008, 2014, doi: 10.1109/IEMBS.2008.4649987.

[12] J. Wu, "Introduction to Convolutional Neural Networks," *Introd. to Convolutional Neural Networks*, pp. 1–31, 2017, [Online]. Available: https://web.archive.org/web/20180928011532/https://cs.nju.edu.cn/wujx/teaching/15_CNN.pdf.

[13] Y. Bai, L. Zhang, D. Wan, Y. Xie, and H. Deng, "Detection of sleep apnea syndrome by CNN based on ECG," *J. Phys. Conf. Ser.*, vol. 1757, no. 1, 2021, doi: 10.1088/1742-6596/1757/1/012043.

[14] D. Mukherjee, K. Dhar, F. Schwenker, and R. Sarkar, "Ensemble of deep learning models for sleep apnea detection: An experimental study," *Sensors*, vol. 21, no. 16, pp. 1–17, 2021, doi: 10.3390/s21165425.

[15] H. Y. Chang, C. Y. Yeh, C. Te Lee, and C. C. Lin, "A sleep apnea detection system based on a one-dimensional deep convolution neural network model using single-lead electrocardiogram," *Sensors (Switzerland)*, vol. 20, no. 15, pp. 1–15, 2020, doi: 10.3390/s20154157.

[16] T. Penzel, G. B. Moody, R. G. Mark, A. L. Goldberger, and J. H. Peter, "Apnea-ECG database," *Comput. Cardiol.*, pp. 255–258, 2000.

[17] C. Lin, Y. Wang, F. Setiawan, N. Thi, H. Trang, and C. Lin, "Sleep Apnea Classification Algorithm Development Using a Machine-Learning Framework and Bag-of-Features Derived from Electrocardiogram Spectrograms," 2022.

[18] A. Yildiz, M. Akin, and M. Poyraz, "An expert system for automated recognition of patients with obstructive sleep apnea using electrocardiogram recordings," *Expert Syst. Appl.*, vol. 38, no. 10, pp. 12880–12890, 2011, doi: 10.1016/j.eswa.2011.04.080.

[19] J. C. Nunes and É. Deléchelle, “Empirical mode decomposition: Applications on signal and image processing,” *Adv. Adapt. Data Anal.*, vol. 1, no. 1, pp. 125–175, 2009, doi: 10.1142/S1793536909000059.

[20] S. F. Q. Conrad Iber, Sonia Ancoli, Andrew L. Chesson JR., “The AASM Manual for the Scoring of Sleep and Associated Events.” 2007.

[21] M. H. Mozaffari and L.-L. Tay, “A Review of 1D Convolutional Neural Networks toward Unknown Substance Identification in Portable Raman Spectrometer,” 2020, [Online]. Available: http://arxiv.org/abs/2006.10575.

[22] M. J. J. Douglass, *Book Review: Hands-on Machine Learning with Scikit-Learn, Keras, and Tensorflow, 2nd edition by Aurélien Géron*, vol. 43, no. 3. 2020.

[23] S. Raschka, “An Overview of General Performance Metrics of Binary Classifier System,” *Nature*, vol. 388. 2014.

[24] H. M and S. M.N, “A Review on Evaluation Metrics for Data Classification Evaluations,” *Int. J. Data Min. Knowl. Manag. Process*, vol. 5, no. 2, pp. 01–11, 2015, doi: 10.5121/ijdkp.2015.5201.

[25] E. Izci, M. A. Özdemir, R. Sadighzadeh, and A. Akan, “Arrhythmia Detection on ECG Signals by Using Empirical Mode Decomposition,” *2018 Med. Technol. Natl. Congr. TIPTEKNO 2018*, no. December, 2018, doi: 10.1109/TIPTEKNO.2018.8597094.

LAMPIRAN A – PENGGALAN KODE PROGRAM

A.1 *Signal Preprocessing*

```
# -*- coding: utf-8 -*-
"""
Created on Mon Nov 29 21:21:09 2021

@author: fnx
"""
import matplotlib.pyplot as plt
import scipy
import wfdb
import numpy as np
import pandas as pd
import os
import sklearn
'Filter'
from scipy import signal
from scipy.io import loadmat
from scipy.signal import butter, filtfilt, iirnotch, savgol_filter

'iterasi'
import os
import pathlib

'Scaler'
from sklearn.preprocessing import MinMaxScaler

'EMD'
from PyEMD import EMD, Visualisation
import numpy as np
os.chdir('E:/Felix/1822041/Tugas Akhir/')

'Timer'
from timeit import default_timer as timer
from datetime import timedelta

'Iterasi'
letter =
['a01','a02','a03','a04','a05','a06','a07','a08','a09','a10','a11','
a12','a13','a14','a15','a16','a17','a18','a19','a20','b01','b02','b0
```

```
3','b04','b05','c01','c02','c03','c04','c06','c07','c08','c09','c10'
]
num =
[8,8,8,8,7,8,8,8,8,8,7,8,8,8,8,8,6,8,8,8,8,8,7,7,7,8,8,7,8,7,7,8,7,7
]


data_i_1_conv_temp =  pd.DataFrame([])
data_i_2_conv_temp =  pd.DataFrame([])
data_i_3_conv_temp =  pd.DataFrame([])
data_i_4_conv_temp =  pd.DataFrame([])
start = timer()

'all imfs'
def main():

    i = 0
    j = 1

    while i < len(letter):
        while j <= num[i]:
            record_num = letter[i] + '_s' + str(j)
            record_dir = "cleaned data/ECG2/" + record_num + '.mat'

            record = loadmat(record_dir)

            filtered_data, time,norm_val = filter_final(record)
            filtered_data_tp = np.transpose(filtered_data)
            data_i_1, data_i_2, data_i_3, data_i_4, data_i_1_conv,
data_i_2_conv, data_i_3_conv, data_i_4_conv =
emd_signal(filtered_data_tp,time)
            save_file_imf_1(data_i_1, data_i_1_conv,
record_num,time)
            save_file_imf_2(data_i_2,data_i_2_conv,record_num,time)
            save_file_imf_3(data_i_3,data_i_3_conv,record_num,time)
            save_file_imf_4(data_i_4,data_i_4_conv,record_num,time)

            j+=1
        j=1
        i+=1

def filter_final (record):
    global val, filtered_data, time, norm_val
    'File input'

    val = record['val']
    val = np.array(val)
```

```
    val = np.transpose(val)
    '1 jam'

    val = val


    'Scaler/MinMax Normalize only'
    scaler = MinMaxScaler()
    norm_val = scaler.fit_transform(val)
    time = [0]
    i=0
    x=0
    while i<len(val)-1:
        x = x + 0.01
        x  = round(x,4)
        time.append(x)
        i = i+1
    cutoff=0.05
    sample_rate=100
    norm_val_tp = np.transpose(norm_val)
    data=norm_val_tp
    'Filtering with Notch Filter'
    b, a = iirnotch(cutoff, Q = 0.005, fs = sample_rate)
    filtered_data = filtfilt(b, a, data)
    filtered_data_tp = np.transpose(filtered_data)
    return filtered_data, time, norm_val




'EMD Signal Function'
def emd_signal(filtered_data_tp,time):
    global data
    global data_i_1
    global data_i_1_conv, data_i_2_conv, data_i_3_conv,
data_i_4_conv
    global data_i_2
    global data_i_3
    global data_i_4
    global imfs, data_i_1_conv_reshape , data_i_2_conv_reshape,
data_i_3_conv_reshape, data_i_4_conv_reshape
    # global data_i_1
    time_arr = np.array(time)
    t = time_arr
    S_tp = filtered_data_tp.flatten()
    emd = EMD()
    emd.emd(S_tp)
```

```
    imfs, res = emd.get_imfs_and_residue()
    data = pd.DataFrame(data = imfs)

    'ambil imf ke-1,2,3,4 dari row ke 0,1,2,3'
    data_i_1 = data.iloc[[0]]
    data_i_2 = data.iloc[[1]]
    data_i_3 = data.iloc[[2]]
    data_i_4 = data.iloc[[3]]

    'convert dataframes to np in order to use np.reshape()'
    data_i_1_conv = data_i_1.to_numpy()
    data_i_2_conv = data_i_2.to_numpy()
    data_i_3_conv = data_i_3.to_numpy()
    data_i_4_conv = data_i_4.to_numpy()
    'from 1 x 360.000 reshape to 60x6000, 60 rows + 6000 cols'
    data_i_1_conv = data_i_1_conv.reshape(60,6000)
    print('IMF 1 data has been reshaped to :', data_i_1_conv.shape)
    data_i_2_conv = data_i_2_conv.reshape(60,6000)
    print('IMF 2 data has been reshaped to :', data_i_2_conv.shape)
    data_i_3_conv = data_i_3_conv.reshape(60,6000)
    print('IMF 3 data has been reshaped to :', data_i_3_conv.shape)
    data_i_4_conv = data_i_4_conv.reshape(60,6000)
    print('IMF 4 data has been reshaped to :', data_i_4_conv.shape)

    data_i_1_conv_reshape = data_i_1_conv.transpose()
    data_i_2_conv_reshape = data_i_2_conv.transpose()
    data_i_3_conv_reshape = data_i_3_conv.transpose()
    data_i_4_conv_reshape = data_i_4_conv.transpose()

    'conv back to pandas dataframe in order to conv them to csv
files...'
    data_i_1_conv = pd.DataFrame(data_i_1_conv)
    print('IMF 1 data has been converted to DataFrame with shape :
', data_i_1_conv.shape)
    data_i_2_conv = pd.DataFrame(data_i_2_conv)
    print('IMF 2 data has been converted to DataFrame with shape :
', data_i_2_conv.shape)
    data_i_3_conv = pd.DataFrame(data_i_3_conv)
    print('IMF 3 data has been converted to DataFrame with shape :
', data_i_3_conv.shape)
    data_i_4_conv = pd.DataFrame(data_i_4_conv)
    print('IMF 4 data has been converted to DataFrame with shape :
', data_i_4_conv.shape)

    return data_i_1, data_i_2, data_i_3, data_i_4, data_i_1_conv,
data_i_2_conv, data_i_3_conv, data_i_4_conv
```

```
'save imf data'
def save_file_imf_1(data_i_1, data_i_1_conv, record_num,time):
    global data_i_1_conv_temp
    data_i_1_conv_temp = data_i_1_conv_temp.append(data_i_1_conv,
ignore_index=True)
    data_i_1_conv_temp.to_csv('E:/Felix/1822041/Tugas
Akhir/new/imf_1/record_imf1_' + record_num + '.csv',index=False)
    print('Success! CSV Format of IMF1 Record from ',record_num,'
with shape : ',data_i_1_conv_temp.shape)


def save_file_imf_2(data_i_2,data_i_2_conv,record_num,time):
    global data_i_2_conv_temp
    data_i_2_conv_temp = data_i_2_conv_temp.append(data_i_2_conv,
ignore_index=True)
    data_i_2_conv_temp.to_csv('E:/Felix/1822041/Tugas
Akhir/new/imf_2/record_imf2_' + record_num +  '.csv',index=False)
    print('Success! CSV Format of IMF2 Record from ',record_num,'
with shape : ',data_i_2_conv_temp.shape)

def save_file_imf_3(data_i_3,data_i_3_conv, record_num,time):
    global data_i_3_conv_temp
    data_i_3_conv_temp = data_i_3_conv_temp.append(data_i_3_conv,
ignore_index=True)
    data_i_3_conv_temp.to_csv('E:/Felix/1822041/Tugas
Akhir/new/imf_3/record_imf3_'  + record_num + '.csv',index=False)
    print('Success! CSV Format of IMF3 Record from ',record_num,'
with shape : ',data_i_3_conv_temp.shape)

def save_file_imf_4(data_i_4,data_i_4_conv, record_num,time):
    global data_i_4_conv_temp
    data_i_4_conv_temp = data_i_4_conv_temp.append(data_i_4_conv,
ignore_index=True)
    data_i_4_conv_temp.to_csv('E:/Felix/1822041/Tugas
Akhir/new/imf_4/record_imf4_'  + record_num + '.csv',index=False)
    print('Success! CSV Format of IMF4 Record from ',record_num,'
with shape : ',data_i_4_conv_temp.shape)

'run program'
if __name__ == "__main__":
    main()
    end = timer()
    print('Time elapsed: ', timedelta(seconds=end-start))
```

A.2 *Training Model* dengan *Imbalanced Dataset*

```
# -*- coding: utf-8 -*-
"""
Created on Tue Dec  7 16:43:37 2021

@author: fnx
"""

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from timeit import default_timer as timer
from datetime import timedelta
import seaborn as sns
'Standard Scaler'
from sklearn.preprocessing import StandardScaler

'Deep Learning'
import tensorflow as tf
from tensorflow.keras.layers import Flatten, Dense, Conv1D,
MaxPool1D, Dropout, Input
from tensorflow.keras import Model
from tensorflow.keras.utils import plot_model
import os
os.chdir('E:/Felix/1822041/Tugas Akhir/')

'Confusion matrix'
from sklearn.metrics import confusion_matrix, accuracy_score,
precision_score, recall_score, f1_score
from sklearn import metrics

'Specificity+Sensitivity'
import imblearn

def main():
    global df_final,df_final_scaled
    global annot, annot_map
    global imf_no
    global test_no, conv_no, dense_no,
filt_no,k_size,dropout_rate,k_init
    global filt_size

    imf_no = '2'
```

```
    test_no = '22'
    conv_no = '10'
    dense_no = '4'
    filt_no ='45'

    filt_size = 45
    k_size =32
    dropout_rate = 0.5
    global dense_neuron
    dense_neuron = 512
    k_init = 'he_normal'
    print('Begin main funct')
    # recordAnnot_imf1_final
    # df_final_dir = 'E:/Felix/1822041/Tugas
Akhir/new/imf_'+imf_no+'/recordAnnot_imf'+imf_no+'_final.csv'
    df_final_dir = 'E:/Felix/1822041/Tugas
Akhir/data_imfs_rev/imf_'+imf_no+'/record_imf'+imf_no+'_final.csv'
    print('Now reading... : '+df_final_dir)
    df_final = pd.read_csv(df_final_dir)
    print('Success reading record file from: ', df_final_dir)
    'np array to expand dim'
    df_final = df_final.to_numpy()
    print('Success converting ', df_final_dir, ' to np format')

    'Scale first then train-test-split'
    print('Begin Scaler')
    scaler = StandardScaler()
    df_final_scaled = scaler.fit_transform(df_final)
    print('df scaled with StandardScaler()')


    'Expand dims before train-test-split'
    df_final_scaled = np.expand_dims(df_final, axis=-1)

    'Read annotation files'
    annot_dir = 'E:/Felix/1822041/Tugas Akhir/annot/annot.csv'
    annot = pd.read_csv(annot_dir)
    print('Annotation shape: ', annot.shape)


    count_a = np.count_nonzero(annot == 'A')
    print('Number of As in annot: ', count_a)
    count_n = np.count_nonzero(annot == 'N')
    print('Number of Ns in annot: ', count_n)


    'Change Labels from A/N to 1/0, where A = 1, N = 0'
```

```
    annot_map = annot.to_numpy()
    annot_map = np.where(annot_map == 'N', 0,annot_map)
    annot_map = np.where(annot_map == 'A', 1,annot_map)

    annot_map = np.array(annot_map)

    train(df_final_scaled, annot_map)
    model = dl_algo(X_train, y_train, X_test, y_test, X_val, y_val)
    model.summary()

    'Plot Model'


    'Plot'
    train_data(X_train, y_train, X_test, y_test, X_val, y_val,
model)


def train(df_2_scaled, annot_map):
    print('Begin train funct')
    global X_train, y_train
    global X_test, y_test
    global X_val, y_val
    global train_x_scaled,test_x_scaled

    'Train-Test-Split data into Train, Test, Valid datasets with
random state 10'
    print('data input shape before train-test-split:
',df_2_scaled.shape)
    print('annot input shape before train-test-split:
',annot_map.shape)
    X_train, X_test, y_train, y_test = train_test_split(df_2_scaled,
annot_map,
test_size=0.4,stratify=np.array(annot_map),random_state=10)
    X_test, X_val, y_test, y_val = train_test_split(X_test, y_test,
test_size=0.5,stratify=np.array(y_test),random_state=10)

    'array values in each portion of data is set to type float32'
    X_train = np.asarray(X_train).astype(np.float32)
    X_test = np.asarray(X_test).astype(np.float32)
    X_val = np.asarray(X_val).astype(np.float32)
    y_train = np.asarray(y_train).astype(np.float32)
    y_val = np.asarray(y_val).astype(np.float32)
    y_test = np.asarray(y_test).astype(np.float32)
    print('X_train shape: ', X_train.shape,'y_train shape: ',
y_train.shape)
```

```
    print('X_test shape: ', X_test.shape,'y_test shape: ',
y_test.shape)
    print('X_val shape: ', X_val.shape,'y_val shape: ', y_val.shape)

    return X_train, y_train, X_test, y_test, X_val, y_val




def dl_algo(X_train, y_train, X_test, y_test, X_val, y_val):


    input_ = Input(shape = X_train.shape[1:])

    'Training Block'
    'Feature Extraction Layer starts here'
    # x = Conv1D(filters=45, kernel_size=32, padding='same',
kernel_initializer='he_normal',activation ='relu',
name='block1_conv1') (input_)
    print('Begin computing block no: 1')
    x = Conv1D(filters=filt_size, kernel_size=k_size,
padding='same', kernel_initializer=k_init, name='block1_conv')
(input_)
    x = tf.keras.layers.BatchNormalization(name='block1_BatchNorm')
(x)
    x = tf.keras.layers.Activation(tf.keras.activations.relu) (x)
    x = MaxPool1D(pool_size=2, strides=2, name = 'block1_MaxPool')
(x)
    # x = Dropout(dropout_rate,name = 'block1_DropOut') (x)
    print('Finish computing block no: 1')

    print('Begin computing block no: 2')
    x = Conv1D(filters=filt_size, kernel_size=k_size,
padding='same', kernel_initializer=k_init, name='block2_conv') (x)
    x = tf.keras.layers.BatchNormalization(name='block2_BatchNorm')
(x)
    x = tf.keras.layers.Activation(tf.keras.activations.relu) (x)
    x = MaxPool1D(pool_size=2, strides=2, name = 'block2_MaxPool')
(x)
    # x = Dropout(dropout_rate,name = 'block2_DropOut') (x)
    print('Finish computing block no: 2')


    print('Begin computing block no: 3')
    x = Conv1D(filters=filt_size, kernel_size=k_size,
padding='same', kernel_initializer=k_init, name='block3_conv') (x)
```

```
    x = tf.keras.layers.BatchNormalization(name='block3_BatchNorm')
(x)
    x = tf.keras.layers.Activation(tf.keras.activations.relu) (x)
    x = MaxPool1D(pool_size=2, strides=2, name = 'block3_MaxPool')
(x)
    # x = Dropout(dropout_rate,name = 'block3_DropOut') (x)
    print('Finish computing block no: 3')


    print('Begin computing block no: 4')
    x = Conv1D(filters=filt_size, kernel_size=k_size,
padding='same', kernel_initializer=k_init, name='block4_conv') (x)
    x = tf.keras.layers.BatchNormalization(name='block4_BatchNorm')
(x)
    x = tf.keras.layers.Activation(tf.keras.activations.relu) (x)
    x = MaxPool1D(pool_size=2, strides=2, name = 'block4_MaxPool')
(x)
    # x = Dropout(dropout_rate,name = 'block4_DropOut') (x)
    print('Finish computing block no: 4')



    print('Begin computing block no: 5')
    x = Conv1D(filters=filt_size, kernel_size=k_size,
padding='same', kernel_initializer=k_init, name='block5_conv') (x)
    x = tf.keras.layers.BatchNormalization(name='block5_BatchNorm')
(x)
    x = tf.keras.layers.Activation(tf.keras.activations.relu) (x)
    x = MaxPool1D(pool_size=2, strides=2, name = 'block5_MaxPool')
(x)
    # x = Dropout(dropout_rate,name = 'block5_DropOut') (x)
    print('Finish computing block no: 5')


    print('Begin computing block no: 6')
    x = Conv1D(filters=filt_size, kernel_size=k_size,
padding='same', kernel_initializer=k_init, name='block6_conv') (x)
    x = tf.keras.layers.BatchNormalization(name='block6_BatchNorm')
(x)
    x = tf.keras.layers.Activation(tf.keras.activations.relu) (x)
    x = MaxPool1D(pool_size=2, strides=2, name = 'block6_MaxPool')
(x)
    # x = Dropout(dropout_rate,name = 'block6_DropOut') (x)
    print('Finish computing block no: 6')


    print('Begin computing block no: 7')
```

```
    x = Conv1D(filters=filt_size, kernel_size=k_size,
padding='same', kernel_initializer=k_init, name='block7_conv') (x)
    x = tf.keras.layers.BatchNormalization(name='block7_BatchNorm')
(x)
    x = tf.keras.layers.Activation(tf.keras.activations.relu) (x)
    x = MaxPool1D(pool_size=2, strides=2, name = 'block7_MaxPool')
(x)
    # x = Dropout(dropout_rate,name = 'block7_DropOut') (x)
    print('Finish computing block no: 7')



    print('Begin computing block no: 8')
    x = Conv1D(filters=filt_size, kernel_size=k_size,
padding='same', kernel_initializer=k_init, name='block8_conv') (x)
    x = tf.keras.layers.BatchNormalization(name='block8_BatchNorm')
(x)
    x = tf.keras.layers.Activation(tf.keras.activations.relu) (x)
    x = MaxPool1D(pool_size=2, strides=2, name = 'block8_MaxPool')
(x)
    # x = Dropout(dropout_rate,name = 'block8_DropOut') (x)
    print('Finish computing block no: 8')



    print('Begin computing block no: 9')
    x = Conv1D(filters=filt_size, kernel_size=k_size,
padding='same', kernel_initializer=k_init, name='block9_conv') (x)
    x = tf.keras.layers.BatchNormalization(name='block9_BatchNorm')
(x)
    x = tf.keras.layers.Activation(tf.keras.activations.relu) (x)
    x = MaxPool1D(pool_size=2, strides=2, name = 'block9_MaxPool')
(x)
    # x = Dropout(dropout_rate,name = 'block9_DropOut') (x)
    print('Finish computing block no: 9')


    print('Begin computing block no: 10')
    x = Conv1D(filters=filt_size, kernel_size=k_size,
padding='same', kernel_initializer=k_init, name='block10_conv') (x)
    x = tf.keras.layers.BatchNormalization(name='block10_BatchNorm')
(x)
    x = tf.keras.layers.Activation(tf.keras.activations.relu) (x)
    x = MaxPool1D(pool_size=2, strides=2, name = 'block10_MaxPool')
(x)
    # x = Dropout(dropout_rate,name = 'block10_DropOut') (x)
    print('Finish computing block no: 10')
```

```
    # print('Begin computing block no: 11')
    # x = Conv1D(filters=filt_size, kernel_size=k_size,
padding='same', kernel_initializer=k_init, name='block11_conv') (x)
    # x =
tf.keras.layers.BatchNormalization(name='block11_BatchNorm') (x)
    # x = tf.keras.layers.Activation(tf.keras.activations.relu) (x)
    # x = MaxPool1D(pool_size=2, strides=2, name =
'block11_MaxPool') (x)
    # x = Dropout(dropout_rate,name = 'block11_DropOut') (x)
    # print('Finish computing block no: 11')

    # print('Begin computing block no: 12')
    # x = Conv1D(filters=filt_size, kernel_size=k_size,
padding='same', kernel_initializer=k_init, name='block12_conv') (x)
    # x =
tf.keras.layers.BatchNormalization(name='block12_BatchNorm') (x)
    # x = tf.keras.layers.Activation(tf.keras.activations.relu) (x)
    # x = MaxPool1D(pool_size=2, strides=2, name =
'block12_MaxPool') (x)
    # x = Dropout(dropout_rate,name = 'block12_DropOut') (x)
    # print('Finish computing block no: 12')

    'Feature Extraction Layer ends here'

    'Flatten before FC'
    x = Flatten(name='flatten') (x)
    print('Finish flattening')

    'Classification Layer starts here'

    print('Begin computing Classification Layer-1')
    x = Dense(dense_neuron, kernel_initializer=k_init,
name='fc1_dense' )(x)
    x = tf.keras.layers.BatchNormalization(name='fc1_BatchNorm') (x)
    x = tf.keras.layers.Activation(tf.keras.activations.relu) (x)
    x = Dropout(dropout_rate,name = 'fc1_DropOut') (x)
    print('Finish computing Classification Layer-1')

    print('Begin computing Classification Layer-2')
    x = Dense(dense_neuron, kernel_initializer=k_init,
name='fc2_dense')(x)
    x = tf.keras.layers.BatchNormalization(name='fc2_BatchNorm') (x)
    x = tf.keras.layers.Activation(tf.keras.activations.relu) (x)
    x = Dropout(dropout_rate,name = 'fc2_DropOut') (x)
    print('Finish computing Classification Layer-2')
```

```
    print('Begin computing Classification Layer-3')
    x = Dense(dense_neuron, kernel_initializer=k_init,
name='fc3_dense')(x)
    x = tf.keras.layers.BatchNormalization(name='fc3_BatchNorm') (x)
    x = tf.keras.layers.Activation(tf.keras.activations.relu) (x)
    x = Dropout(dropout_rate,name = 'fc3_DropOut') (x)
    print('Finish computing Classification Layer-3')

    print('Begin computing Classification Layer-4')
    x = Dense(dense_neuron, kernel_initializer=k_init,
name='fc4_dense')(x)
    x = tf.keras.layers.BatchNormalization(name='fc4_BatchNorm') (x)
    x = tf.keras.layers.Activation(tf.keras.activations.relu) (x)
    x = Dropout(dropout_rate,name = 'fc4_DropOut') (x)
    print('Finish computing Classification Layer-4')

    # print('Begin computing Classification Layer-5')
    # x = Dense(512, kernel_initializer='he_normal',
name='fc5_dense')(x)
    # x = tf.keras.layers.BatchNormalization(name='fc5_BatchNorm')
(x)
    # x = tf.keras.layers.Activation(tf.keras.activations.relu) (x)
    # x = Dropout(0.5,name = 'fc5_DropOut') (x)
    # print('Finish computing Classification Layer-5')

    # print('Begin computing Classification Layer-6')
    # x = Dense(512, kernel_initializer='he_normal',
name='fc6_dense')(x)
    # x = tf.keras.layers.BatchNormalization(name='fc6_BatchNorm')
(x)
    # x = tf.keras.layers.Activation(tf.keras.activations.relu) (x)
    # x = Dropout(0.5,name = 'fc6_DropOut') (x)
    # print('Finish computing Classification Layer-6')
    'Classification Layer ends here'


    'Output Layer'
    'softmax'
    # x = Dense(2,activation='softmax', name='out')(x)
    'sigmoid'
    x = Dense(1,activation='sigmoid', name='out')(x)
    model = Model(input_, x)

    return model

def train_data(X_train, y_train, X_test, y_test, X_val, y_val,
model):
```

```
    global epoch_list,y_train_acc, my_predict, my_predict_2

    'compile model with crossentropy loss function and adam
optimizer'
    model.compile(loss='crossentropy',optimizer='adam',metrics=['acc
uracy'])

    model.summary()

    'callback directory'
    cb_dir =
'DL/filt'+filt_no+'/imf'+imf_no+'/'+test_no+'_cb_imf'+imf_no+'_conv'
+conv_no+'_d'+dense_no+'.h5'
    cb_dir_csv =
'DL/filt'+filt_no+'/imf'+imf_no+'/'+test_no+'_cb_imf'+imf_no+'_conv'
+conv_no+'_d'+dense_no+'.csv'

    'callbakcs'
    checkpoint_cb =
tf.keras.callbacks.ModelCheckpoint(cb_dir,save_best_only=True)
    checkpoint_cb_csv =  tf.keras.callbacks.CSVLogger(cb_dir_csv)
    early_stopping_cb =
tf.keras.callbacks.EarlyStopping(patience=20,restore_best_weights=Tr
ue)


    model_dir
='DL/filt'+filt_no+'/imf'+imf_no+'/'+test_no+'_model_imf'+imf_no+'_c
onv'+conv_no+'_d'+dense_no+'.h5'
    model_arch_dir =
'DL/filt'+filt_no+'/imf'+imf_no+'/'+test_no+'_arch_imf'+imf_no+'_con
v'+conv_no+'_d'+dense_no+'.png'

    metrics_dir =
'DL/filt'+filt_no+'/imf'+imf_no+'/'+test_no+'_metrics_imf'+imf_no+'_
conv'+conv_no+'_d'+dense_no+'.csv'

    #fit the model - Train Data
    history =
model.fit(x=X_train,y=y_train,epochs=200,validation_data=(X_val,y_va
l),callbacks=[checkpoint_cb,early_stopping_cb,checkpoint_cb_csv])
    model.save(model_dir)
    model.summary()
    plot_model(model, to_file=model_arch_dir, show_shapes=True)
    my_predict = model.predict(X_test) #use new train model

    'use this if u wanna load trained data...'
```

```
    # saved_model = tf.keras.models.load_model(model_dir)
    # my_predict = saved_model.predict(X_test) #use saved_model
train file


    print('mypredict_ :  ' , my_predict)
    # ' use if SOFTMAX'
    # global output_my_predict_2
    # my_predict_2 = pd.DataFrame(my_predict,
columns=['Normal','Apnea'])
    # output_my_predict_2 = pd.DataFrame([])
    # output_my_predict_2 =
np.where(my_predict_2['Normal']>=my_predict_2['Apnea'],0,1)


    'Data test scores with SIGMOID function'
    my_predict_2 = my_predict.copy()
    my_predict_2[my_predict_2>=0.5]=1
    my_predict_2[my_predict_2<0.5]=0


    accuracy = accuracy_score(y_test, my_predict_2)
    precision = precision_score(y_test, my_predict_2)
    recall = recall_score(y_test, my_predict_2)
    f1 =  f1_score(y_test, my_predict_2)

    accuracy = (accuracy*100)
    precision= (precision*100)
    recall= (recall*100)
    f1 = (f1*100)
    sensitivity = imblearn.metrics.sensitivity_score(y_test,
my_predict_2)
    sensitivity = (sensitivity*100)
    specificity = imblearn.metrics.specificity_score(y_test,
my_predict_2)
    specificity = (specificity*100)
    print('Accuracy: %.3f' % accuracy + '%')
    print('Precision:  %.3f' % precision+ '%' )
    print('Recall:  %.3f' % recall+'%'  )
    print('F1 Score:  %.3f' %f1 +'%' )
    print('Sensitivity: %.3f ' % sensitivity + '%')
    print('Specificity: %.3f  ' % specificity + '%')

    metrics = []
    metrics = [accuracy,precision,recall,f1,sensitivity,specificity]
    metrics = pd.DataFrame([metrics])
```

```
    metrics.columns=['Accuracy','Precision','Recall','F1_Score','Sen
sitivity','Specificity']


    #save metrics report to csv
    print('Saving metrics to: '+metrics_dir)
    metrics.to_csv(metrics_dir,index=False)

    # report = metrics.classification_report(my_predict_2,
output_dict=True)
    # df_report = pd.DataFrame(report).transpose()
    # save_df_report =
df_report.to_csv("DL/filt60/imf"+imf_no+"/metrics_train_result_imf"+
imf_no+"_e100_test_"+test_no+'cov'+conv_no+'_de_'+ dense_no+".csv")

    'plot confusion matrix'
    confusion_matrix_sk = confusion_matrix(y_test, my_predict_2)
    sns.heatmap(confusion_matrix_sk, annot=True, fmt="d");

    plt.xlabel("Predicted Value");
    plt.ylabel("True Value");
    plt.savefig('DL/filt'+filt_no+'/imf'+imf_no+'/'+test_no+'confmat
rix_imf'+imf_no+'_conv'+conv_no+'_d'+dense_no+'.png')

    return



if __name__ == "__main__":
    start = timer()
    main()
    end = timer()
    print('Time elapsed: ', timedelta(seconds=end-start))
```

A.3 *Training Model* dengan *Balanced Dataset*

```
# -*- coding: utf-8 -*-
"""
Created on Sat Jan  1 02:02:13 2022

@author: fnx
"""

import pandas as pd
import numpy as np
```

```
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from timeit import default_timer as timer
from datetime import timedelta
import seaborn as sns
'Standard Scaler'
from sklearn.preprocessing import StandardScaler

'Deep Learning'
import tensorflow as tf
from tensorflow.keras.layers import Flatten, Dense, Conv1D,
MaxPool1D, Dropout, Input
from tensorflow.keras import Model
from tensorflow.keras.utils import plot_model
import os
os.chdir('E:/Felix/1822041/Tugas Akhir/')

'Confusion matrix'
from sklearn.metrics import confusion_matrix, accuracy_score,
precision_score, recall_score, f1_score
from sklearn import metrics

'Specificity+Sensitivity'
import imblearn

def main():
    global df_final,df_final_scaled
    global annot, annot_map
    global imf_no
    global test_no, conv_no, dense_no, filt_no
    imf_no = '2'
    test_no = '99'
    conv_no = '10'
    dense_no = '6'
    filt_no ='60'
    print('Begin main funct')

    df_final_dir = 'E:/Felix/1822041/Tugas
Akhir/new/imf_'+imf_no+'/0imf_'+imf_no+'_sorted.csv'
    print('Now reading... : '+df_final_dir)
    df_final = pd.read_csv(df_final_dir)
    print('Success reading record file from: ', df_final_dir)
    'np array to expand dim'
    df_final = df_final.to_numpy()
    print('Success converting ', df_final_dir, ' to np format')
```

```
    'Scale first then train-test-split'
    print('Begin Scaler')
    scaler = StandardScaler()
    df_final_scaled = scaler.fit_transform(df_final)
    print('df scaled with StandardScaler()')


    'Expand dims before train-test-split'
    df_final_scaled = np.expand_dims(df_final, axis=-1)

    'Read annotation files'
    annot_dir = 'E:/Felix/1822041/Tugas
Akhir/new/imf_'+imf_no+'/annot_sorted_imf'+imf_no+'.csv'
    annot = pd.read_csv(annot_dir)
    print('Annotation shape: ', annot.shape)

    # annot = pd.read_csv('D:/Kuliyah/.Tugas Akhir/cleaned
data/annot/test/c10_s7.csv') #check imbalance data
    'Check if data is imbalance or not'
    count_a = np.count_nonzero(annot == 1)
    print('Number of As in annot: ', count_a)
    count_n = np.count_nonzero(annot == 0)
    print('Number of Ns in annot: ', count_n)



    'Read balanced labels dataset'
    annot_map = annot.to_numpy()
    # annot_map = np.where(annot_map == 'N', 0,annot_map)
    # annot_map = np.where(annot_map == 'A', 1,annot_map)

    annot_map = np.array(annot_map)

    train(df_final_scaled, annot_map)
    model = dl_algo(X_train, y_train, X_test, y_test, X_val, y_val)
    model.summary()


    'Plot'
    train_data(X_train, y_train, X_test, y_test, X_val, y_val,
model)


def train(df_2_scaled, annot_map):
    print('Begin train funct')
    global X_train, y_train
    global X_test, y_test
```

```
    global X_val, y_val
    global train_x_scaled,test_x_scaled

    print('data input shape before train-test-split:
',df_2_scaled.shape)
    print('annot input shape before train-test-split:
',annot_map.shape)
    X_train, X_test, y_train, y_test = train_test_split(df_2_scaled,
annot_map,
test_size=0.2,stratify=np.array(annot_map),random_state=10)
    X_test, X_val, y_test, y_val = train_test_split(X_test, y_test,
test_size=0.5,stratify=np.array(y_test),random_state=10)
    X_train = np.asarray(X_train).astype(np.float32)
    X_test = np.asarray(X_test).astype(np.float32)
    X_val = np.asarray(X_val).astype(np.float32)
    y_train = np.asarray(y_train).astype(np.float32)
    y_val = np.asarray(y_val).astype(np.float32)
    y_test = np.asarray(y_test).astype(np.float32)
    print('X_train shape: ', X_train.shape,'y_train shape: ',
y_train.shape)
    print('X_test shape: ', X_test.shape,'y_test shape: ',
y_test.shape)
    print('X_val shape: ', X_val.shape,'y_val shape: ', y_val.shape)


    return X_train, y_train, X_test, y_test, X_val, y_val




def dl_algo(X_train, y_train, X_test, y_test, X_val, y_val):


    input_ = Input(shape = X_train.shape[1:])
    filt_size = 60
    # x = Conv1D(filters=45, kernel_size=32, padding='same',
kernel_initializer='he_normal',activation ='relu',
name='block1_conv1') (input_)
    print('Begin computing block no: 1')
    x = Conv1D(filters=filt_size, kernel_size=32, padding='same',
kernel_initializer='he_normal', name='block1_conv') (input_)
    x = tf.keras.layers.BatchNormalization(name='block1_BatchNorm')
(x)
    x = tf.keras.layers.Activation(tf.keras.activations.relu) (x)
    x = MaxPool1D(pool_size=2, strides=2, name = 'block1_MaxPool')
(x)
```

```
    x = Dropout(0.5,name = 'block1_DropOut') (x)
    print('Finish computing block no: 1')

    print('Begin computing block no: 2')
    x = Conv1D(filters=filt_size, kernel_size=32, padding='same',
kernel_initializer='he_normal', name='block2_conv') (x)
    x = tf.keras.layers.BatchNormalization(name='block2_BatchNorm')
(x)
    x = tf.keras.layers.Activation(tf.keras.activations.relu) (x)
    x = MaxPool1D(pool_size=2, strides=2, name = 'block2_MaxPool')
(x)
    x = Dropout(0.5,name = 'block2_DropOut') (x)
    print('Finish computing block no: 2')


    print('Begin computing block no: 3')
    x = Conv1D(filters=filt_size, kernel_size=32, padding='same',
kernel_initializer='he_normal', name='block3_conv') (x)
    x = tf.keras.layers.BatchNormalization(name='block3_BatchNorm')
(x)
    x = tf.keras.layers.Activation(tf.keras.activations.relu) (x)
    x = MaxPool1D(pool_size=2, strides=2, name = 'block3_MaxPool')
(x)
    x = Dropout(0.5,name = 'block3_DropOut') (x)
    print('Finish computing block no: 3')


    print('Begin computing block no: 4')
    x = Conv1D(filters=filt_size, kernel_size=32, padding='same',
kernel_initializer='he_normal', name='block4_conv') (x)
    x = tf.keras.layers.BatchNormalization(name='block4_BatchNorm')
(x)
    x = tf.keras.layers.Activation(tf.keras.activations.relu) (x)
    x = MaxPool1D(pool_size=2, strides=2, name = 'block4_MaxPool')
(x)
    x = Dropout(0.5,name = 'block4_DropOut') (x)
    print('Finish computing block no: 4')



    print('Begin computing block no: 5')
    x = Conv1D(filters=filt_size, kernel_size=32, padding='same',
kernel_initializer='he_normal', name='block5_conv') (x)
    x = tf.keras.layers.BatchNormalization(name='block5_BatchNorm')
(x)
    x = tf.keras.layers.Activation(tf.keras.activations.relu) (x)
```

```
    x = MaxPool1D(pool_size=2, strides=2, name = 'block5_MaxPool')
(x)
    x = Dropout(0.5,name = 'block5_DropOut') (x)
    print('Finish computing block no: 5')


    print('Begin computing block no: 6')
    x = Conv1D(filters=filt_size, kernel_size=32, padding='same',
kernel_initializer='he_normal', name='block6_conv') (x)
    x = tf.keras.layers.BatchNormalization(name='block6_BatchNorm')
(x)
    x = tf.keras.layers.Activation(tf.keras.activations.relu) (x)
    x = MaxPool1D(pool_size=2, strides=2, name = 'block6_MaxPool')
(x)
    x = Dropout(0.5,name = 'block6_DropOut') (x)
    print('Finish computing block no: 6')


    print('Begin computing block no: 7')
    x = Conv1D(filters=filt_size, kernel_size=32, padding='same',
kernel_initializer='he_normal', name='block7_conv') (x)
    x = tf.keras.layers.BatchNormalization(name='block7_BatchNorm')
(x)
    x = tf.keras.layers.Activation(tf.keras.activations.relu) (x)
    x = MaxPool1D(pool_size=2, strides=2, name = 'block7_MaxPool')
(x)
    x = Dropout(0.5,name = 'block7_DropOut') (x)
    print('Finish computing block no: 7')



    print('Begin computing block no: 8')
    x = Conv1D(filters=filt_size, kernel_size=32, padding='same',
kernel_initializer='he_normal', name='block8_conv') (x)
    x = tf.keras.layers.BatchNormalization(name='block8_BatchNorm')
(x)
    x = tf.keras.layers.Activation(tf.keras.activations.relu) (x)
    x = MaxPool1D(pool_size=2, strides=2, name = 'block8_MaxPool')
(x)
    x = Dropout(0.5,name = 'block8_DropOut') (x)
    print('Finish computing block no: 8')



    print('Begin computing block no: 9')
    x = Conv1D(filters=filt_size, kernel_size=32, padding='same',
kernel_initializer='he_normal', name='block9_conv') (x)
```

```
    x = tf.keras.layers.BatchNormalization(name='block9_BatchNorm')
(x)
    x = tf.keras.layers.Activation(tf.keras.activations.relu) (x)
    x = MaxPool1D(pool_size=2, strides=2, name = 'block9_MaxPool')
(x)
    x = Dropout(0.5,name = 'block9_DropOut') (x)
    print('Finish computing block no: 9')


    print('Begin computing block no: 10')
    x = Conv1D(filters=filt_size, kernel_size=32, padding='same',
kernel_initializer='he_normal', name='block10_conv') (x)
    x = tf.keras.layers.BatchNormalization(name='block10_BatchNorm')
(x)
    x = tf.keras.layers.Activation(tf.keras.activations.relu) (x)
    x = MaxPool1D(pool_size=2, strides=2, name = 'block10_MaxPool')
(x)
    x = Dropout(0.5,name = 'block10_DropOut') (x)
    print('Finish computing block no: 10')

    # print('Begin computing block no: 11')
    # x = Conv1D(filters=filt_size, kernel_size=32, padding='same',
kernel_initializer='he_normal', name='block11_conv') (x)
    # x =
tf.keras.layers.BatchNormalization(name='block11_BatchNorm') (x)
    # x = tf.keras.layers.Activation(tf.keras.activations.relu) (x)
    # x = MaxPool1D(pool_size=2, strides=2, name =
'block11_MaxPool') (x)
    # x = Dropout(0.5,name = 'block11_DropOut') (x)
    # print('Finish computing block no: 11')

    # print('Begin computing block no: 12')
    # x = Conv1D(filters=filt_size, kernel_size=32, padding='same',
kernel_initializer='he_normal', name='block12_conv') (x)
    # x =
tf.keras.layers.BatchNormalization(name='block12_BatchNorm') (x)
    # x = tf.keras.layers.Activation(tf.keras.activations.relu) (x)
    # x = MaxPool1D(pool_size=2, strides=2, name =
'block12_MaxPool') (x)
    # x = Dropout(0.5,name = 'block12_DropOut') (x)
    # print('Finish computing block no: 12')


    'Flatten before FC'
    x = Flatten(name='flatten') (x)
    print('Finish flattening')
    'Classification Layer'
```

```
    print('Begin computing Classification Layer-1')
    x = Dense(512, kernel_initializer='he_normal',
name='fc1_dense')(x)
    x = tf.keras.layers.BatchNormalization(name='fc1_BatchNorm') (x)
    x = tf.keras.layers.Activation(tf.keras.activations.relu) (x)
    x = Dropout(0.5,name = 'fc1_DropOut') (x)
    print('Finish computing Classification Layer-1')

    print('Begin computing Classification Layer-2')
    x = Dense(512, kernel_initializer='he_normal',
name='fc2_dense')(x)
    x = tf.keras.layers.BatchNormalization(name='fc2_BatchNorm') (x)
    x = tf.keras.layers.Activation(tf.keras.activations.relu) (x)
    x = Dropout(0.5,name = 'fc2_DropOut') (x)
    print('Finish computing Classification Layer-2')

    print('Begin computing Classification Layer-3')
    x = Dense(512, kernel_initializer='he_normal',
name='fc3_dense')(x)
    x = tf.keras.layers.BatchNormalization(name='fc3_BatchNorm') (x)
    x = tf.keras.layers.Activation(tf.keras.activations.relu) (x)
    x = Dropout(0.5,name = 'fc3_DropOut') (x)
    print('Finish computing Classification Layer-3')

    print('Begin computing Classification Layer-4')
    x = Dense(512, kernel_initializer='he_normal',
name='fc4_dense')(x)
    x = tf.keras.layers.BatchNormalization(name='fc4_BatchNorm') (x)
    x = tf.keras.layers.Activation(tf.keras.activations.relu) (x)
    x = Dropout(0.5,name = 'fc4_DropOut') (x)
    print('Finish computing Classification Layer-4')

    print('Begin computing Classification Layer-5')
    x = Dense(512, kernel_initializer='he_normal',
name='fc5_dense')(x)
    x = tf.keras.layers.BatchNormalization(name='fc5_BatchNorm') (x)
    x = tf.keras.layers.Activation(tf.keras.activations.relu) (x)
    x = Dropout(0.5,name = 'fc5_DropOut') (x)
    print('Finish computing Classification Layer-5')

    print('Begin computing Classification Layer-6')
    x = Dense(512, kernel_initializer='he_normal',
name='fc6_dense')(x)
    x = tf.keras.layers.BatchNormalization(name='fc6_BatchNorm') (x)
    x = tf.keras.layers.Activation(tf.keras.activations.relu) (x)
    x = Dropout(0.5,name = 'fc6_DropOut') (x)
```

```
    print('Finish computing Classification Layer-6')
    'Softmax Output'
    # x = Dense(2,activation='softmax', name='out')(x)
    x = Dense(1,activation='sigmoid', name='out')(x)
    model = Model(input_, x)

    return model

def train_data(X_train, y_train, X_test, y_test, X_val, y_val,
model):
    global epoch_list,y_train_acc, my_predict, my_predict_2
    'convert to tensor'

    model.compile(loss='binary_crossentropy',optimizer='adam',metric
s=['accuracy'])

    model.summary()


    cb_dir =
'DL/balanced/filt'+filt_no+'/imf'+imf_no+'/'+test_no+'_cb_imf'+imf_n
o+'_conv'+conv_no+'_d'+dense_no+'.h5'
    cb_dir_csv =
'DL/balanced/filt'+filt_no+'/imf'+imf_no+'/'+test_no+'_cb_imf'+imf_n
o+'_conv'+conv_no+'_d'+dense_no+'.csv'

    checkpoint_cb =
tf.keras.callbacks.ModelCheckpoint(cb_dir,save_best_only=True)
    checkpoint_cb_csv =  tf.keras.callbacks.CSVLogger(cb_dir_csv)
    early_stopping_cb =
tf.keras.callbacks.EarlyStopping(patience=30,restore_best_weights=Tr
ue)


    model_dir
='DL/balanced/filt'+filt_no+'/imf'+imf_no+'/'+test_no+'_model_imf'+i
mf_no+'_conv'+conv_no+'_d'+dense_no+'.h5'
    model_arch_dir =
'DL/balanced/filt'+filt_no+'/imf'+imf_no+'/'+test_no+'_arch_imf'+imf
_no+'_conv'+conv_no+'_d'+dense_no+'.png'

    metrics_dir =
'DL/balanced/filt'+filt_no+'/imf'+imf_no+'/'+test_no+'_metrics_imf'+
imf_no+'_conv'+conv_no+'_d'+dense_no+'.csv'

    #fit the model - Train Data
```

```
    history =
model.fit(x=X_train,y=y_train,epochs=200,validation_data=(X_val,y_va
l),callbacks=[checkpoint_cb,early_stopping_cb,checkpoint_cb_csv])
    model.save(model_dir)
    model.summary()
    plot_model(model, to_file=model_arch_dir, show_shapes=True)
    my_predict = model.predict(X_test) #use new train model

    'use this if u wanna load trained data...'
    # saved_model = tf.keras.models.load_model(model_dir)
    # my_predict = saved_model.predict(X_test) #use saved_model
train file



    print('mypredict_ :  ' , my_predict)

    my_predict_2 = my_predict.copy()
    my_predict_2[my_predict_2>=0.5]=1
    my_predict_2[my_predict_2<0.5]=0

    # my_predict_2

    accuracy = accuracy_score(y_test, my_predict_2)
    precision = precision_score(y_test, my_predict_2)
    recall = recall_score(y_test, my_predict_2)
    f1 =  f1_score(y_test, my_predict_2)

    accuracy = (accuracy*100)
    precision= (precision*100)
    recall= (recall*100)
    f1 = (f1*100)
    print('Accuracy: %.3f' % accuracy + '%')
    print('Precision:  %.3f' % precision+ '%' )
    print('Recall:  %.3f' % recall+'%'  )
    print('F1 Score:  %.3f' %f1 +'%' )

    sensitivity = imblearn.metrics.sensitivity_score(y_test,
my_predict_2)
    sensitivity = (sensitivity*100)
    specificity = imblearn.metrics.specificity_score(y_test,
my_predict_2)
    specificity = (specificity*100)
    print('Sensitivity: %.3f ' % sensitivity + '%')
    print('Specificity: %.3f  ' % specificity + '%')

    metrics = []
    metrics = [accuracy,precision,recall,f1,sensitivity,specificity]
```

```
    metrics = pd.DataFrame([metrics])
    metrics.columns=['Accuracy','Precision','Recall','F1_Score','Sen
sitivity','Specificity']


    #save metrics report to csv
    print('Saving metrics to: '+metrics_dir)
    metrics.to_csv(metrics_dir,index=False)



    confusion_matrix_sk = confusion_matrix(y_test, my_predict_2)
    sns.heatmap(confusion_matrix_sk, annot=True, fmt="d");

    plt.xlabel("Predicted Value");
    plt.ylabel("True Value");
    plt.savefig('DL/balanced/filt'+filt_no+'/imf'+imf_no+'/'+test_no
+'confmatrix_imf'+imf_no+'_conv'+conv_no+'_d'+dense_no+'.png')


    return


if __name__ == "__main__":
    start = timer()
    main()
    end = timer()
    print('Time elapsed: ', timedelta(seconds=end-start))
```

A.4 Memetakan Kurva Validasi

```
# -*- coding: utf-8 -*-
"""
Created on Mon Dec 13 08:44:48 2021

@author: alani
"""

import pandas as pd
import matplotlib.pyplot as plt
'99: kernel size 16'

'architecture parameters'
imf_no = '2'
test_no = '29'
conv_no = '10'
dense_no = '4'
```

```
filt_no ='45'

cb_dir =
'DL/filt'+filt_no+'/imf'+imf_no+'/'+test_no+'_cb_imf'+imf_no+'_conv'
+conv_no+'_d'+dense_no+'.h5'
cb_dir_csv =
'DL/filt'+filt_no+'/imf'+imf_no+'/'+test_no+'_cb_imf'+imf_no+'_conv'
+conv_no+'_d'+dense_no+'.csv'
val_fig_dir =
'DL/filt'+filt_no+'/imf'+imf_no+'/'+test_no+'_valmat_imf'+imf_no+'_c
onv'+conv_no+'_d'+dense_no+'.png'


'Metrics of Training-Validation w/ ratio 80 10 10'
df_epoch = pd.read_csv(cb_dir_csv)
acc = df_epoch.loc[:,'accuracy'].tolist()
val_acc = df_epoch.loc[:,'val_accuracy'].tolist()
loss = df_epoch.loc[:,'loss'].tolist()
val_loss = df_epoch.loc[:,'val_loss'].tolist()



epoch_list = list(range(1,36)) #EPOCH=150
y_train_acc = acc
y_val_acc = val_acc
y_train_loss = loss
y_val_loss = val_loss

f,(ax1,ax2) = plt.subplots(1,2,figsize=(12,4))
t = f.suptitle('IMF'+imf_no+' 1DCNN Conv'+conv_no+'
Dense'+dense_no+', filt'+filt_no+'',fontsize=9)

ax1.plot(epoch_list,y_train_acc,label='Train Accuracy')
ax1.plot(epoch_list,y_val_acc,label='Validation Accuracy')
#ax1.set_xticks(np.arange(0,12,1))
#ax1.set_ylim(0.75,0.85)
ax1.set_ylabel('Accuracy Value')
ax1.set_xlabel('Epoch')
ax1.set_title('Accuracy')
l1=ax1.legend(loc="best")

ax2.plot(epoch_list,y_train_loss,label='Train Loss')
ax2.plot(epoch_list,y_val_loss,label='Validation Loss')
#ax2.set_xticks(np.arange(0,12,1))
#ax2.set_ylim(0,1)
ax2.set_ylabel('Cross Entropy')
ax2.set_xlabel('Epoch')
```

```
ax2.set_title('Loss')
l2=ax2.legend(loc="best")
plt.savefig(val_fig_dir)
print('Validation saved to: ', val_fig_dir)
```

A.5 *Resize and Annotation Mapping*

```
import pandas as pd
import numpy as np

new_annot_dir = 'E:/Felix/1822041/Tugas
Akhir/annot/annot_binary_final.csv'
annotate = pd.read_csv(new_annot_dir)
print('Annotation shape: ', annotate.shape)
df_imf_1_dir = 'E:/Felix/1822041/Tugas
Akhir/new/imf_1/record_imf1_a19_s8.csv'

'concat with'
df2_imf_1_dir= 'E:/Felix/1822041/Tugas
Akhir/new/imf_1/record_imf1_c10_s7.csv'


'IMF1'
df_imf_1 = pd.read_csv(df_imf_1_dir)
print('Success reading df_imf1_a01_b05 from : ', df_imf_1_dir,' with
shape: ', df_imf_1.shape)

df2_imf_1= pd.read_csv(df2_imf_1_dir)
print('Success reading df_imf1_c10 from : ', df2_imf_1_dir,' with
shape: ', df2_imf_1.shape)

df_concat_imf1 = pd.DataFrame([])
df_concat_imf1 =
df_concat_imf1.append(df_imf_1).append(df2_imf_1)#concat files
save_df_concat_imf1 = 'E:/Felix/1822041/Tugas
Akhir/new/imf_1/record_imf1_final.csv'

print('Concat success... with shape: ', df_concat_imf1.shape)
print('Saving concated files to ', save_df_concat_imf1)
df_concat_imf1.to_csv(save_df_concat_imf1,index=False)
print('Success saving df_concat_imf1 files to: ',
save_df_concat_imf1, ' with shape: ', df_concat_imf1.shape)

'Annot Mapping'
annot_map = annotate.to_numpy()
annot_map = np.where(annot_map == 'N', 0,annot_map)
```

```
annot_map = np.where(annot_map == 'A', 1,annot_map)

annot_map = np.array(annot_map)
annot_map = pd.DataFrame(annot_map)
annot_map.to_csv('D:/Kuliyah/.Tugas Akhir/cleaned
data/annot/test/annotNew_c10_s7.csv',index=False
```

LAMPIRAN B – PERBANDINGAN SINYAL ECG TERNORMALISASI, SINYAL ECG HASIL *FILTER*, DAN IMF HASIL EMD

Berikut adalah cuplikan gambar perbandingan sinyal ECG ternormalisasi, sinyal ECG hasil *filter*, dan IMF Hasil EMD.