TUGAS AKHIR – KI141502

IMPLEMENTASI DETEKSI SERANGAN EPILEPSI DARI DATA REKAMAN EEG MENGGUNAKAN WEIGHTED PERMUTATION ENTROPY DAN SUPPORT VECTOR VECTOR MACHINE

LOPHITA Y NAPITUPULU
5113100053

Dosen Pembimbing
Dr. Eng. Nanik Suciati, S.Kom., M.Kom.
Dini Adni Navastara, S.Kom., M.Sc.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2017

TUGAS AKHIR – KI141502

IMPLEMENTASI DETEKSI SERANGAN EPILEPSI DARI DATA REKAMAN EEG MENGGUNAKAN WEIGHTED PERMUTATION ENTROPY DAN SUPPORT VECTOR MACHINE

LOPHITA Y NAPITUPULU
5113100053

Dosen Pembimbing I
Dr. Eng. Nanik Suciati, S.Kom., M.Kom.

Dosen Pembimbing II
Dini Adni Navastara, S.Kom., M.Sc.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya, 2017

[*Halaman ini sengaja dikosongkan*]

FINAL PROJECT - KI141502

IMPLEMENTATION OF EPILEPTIC SEIZURES DETECTION FROM EEG RECORDING DATA USING WEIGHTED PERMUTATION ENTROPY AND SUPPORT VECTOR MACHINE

LOPHITA Y NAPITUPULU
5113100053

Supervisor I
Dr. Eng. Nanik Suciati, S.Kom., M.Kom.

Supervisor II
Dini Adni Navastara, S.Kom., M.Sc.

DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY
Sepuluh Nopember Institute of Technology
Surabaya, 2017

[*Halaman ini sengaja dikosongkan*]

LEMBAR PENGESAHAN

IMPLEMENTASI DETEKSI SERANGAN EPILEPSI DARI DATA REKAMAN EEG MENGGUNAKAN WEIGHTED PERMUTATION ENTROPY DAN SUPPORT VECTOR MACHINE

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Rumpun Mata Kuliah Komputasi Cerdas dan Visi
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh:
LOPHITA Y NAPITUPULU
NRP: 5113 100 053

Disetujui oleh Dosen Pembimbing Tugas Akhir:

Dr. Eng. Nanik Suciati, S.Kom., M.Kom.
NIP. 197208091995121001 ..... (Pembimbing 1)

Dini Adni Navastara, S.Kom., M.Sc.
NIP. 198510172015042001 ..... (Pembimbing 2)

SURABAYA
JUNI, 2017

[*Halaman ini sengaja dikosongkan*]

IMPLEMENTASI DETEKSI SERANGAN EPILEPSI DARI DATA REKAMAN EEG MENGGUNAKAN WEIGHTED PERMUTATION ENTROPY DAN SUPPORT VECTOR MACHINE

Nama Mahasiswa : Lophita Y Napitupulu
NRP : 5113 100 053
Jurusan : Teknik Informatika, FTIf ITS
Dosen Pembimbing 1 : Dr. Eng. Nanik Suciati, S.Kom., M.Kom.
Dosen Pembimbing 2 : Dini Adni Navastara, S.Kom., M.Sc.

Abstrak

Epilepsi merupakan gangguan neurologis jangka panjang yang ditandai dengan serangan-serangan epileptik. Serangan epileptik dapat terjadi dalam waktu singkat hingga guncangan kuat dalam waktu yang lama. Epilepsi adalah penyakit yang cenderung terjadi secara berulang dan tidak dapat disembuhkan, namun serangan-serangan epileptik yang terjadi dapat dikontrol melalui pengobatan.

Pada tugas akhir ini, data rekaman electroencephalogram (EEG) dibagi menjadi beberapa window menggunakan segmentasi atau dekomposisi. Proses selanjutnya adalah mengekstraksi setiap window dengan menggunakan Weighted Permutation Entropy yang menghasilkan satu fitur setiap window. Uji coba fitur menggunakan k-fold cross-validation dengan membagi data menjadi data training dan data testing. Selanjutnya data diklasifikasi menggunakan Support Vector Machine. Data rekaman EEG yang digunakan untuk pengujian ini berasal dari "Klinik für Epileptologie, Universität Bonn" yang diperoleh secara online yang berjumlah 500 data. Data ini terdiri dari serangan epilepsi (set S) dan bukan serangan epilepsi (set Z, N, O, F) yang masing-masing set terdiri dari 100 data. Set Z direkam dari lima orang sehat dengan mata tertutup dan set O direkam dari lima orang sehat mata terbuka. Set F direkam dari penderita

epilepsi yang tidak mengalami serangan di hippocampal formation, set N direkam dari penderita epilepsi yang tidak mengalami sernagan di epileptogenic zone, dan set S direkam dari penderita epilepsi ketika terjadi serangan di epileptogenic zone.

Uji coba dilakukan pada data set S digabung dengan setiap set lain. Sehingga data yang digunakan sebanyak 200 data rekaman EEG untuk setiap uji coba. Berdasarkan uji coba, metode tersebut menghasilkan akurasi rata-rata sebesar 91,88%.

Kata kunci: Epilepsi, EEG, Weighted Permutation Entropy, Support Vector Machine.

IMPLEMENTATION OF EPILEPTIC SEIZURE DETECTION FROM EEG RECORDING DATA USING WEIGHTED PERMUTATION ENTROPY AND SUPPORT VECTOR MACHINE

Student Name	: Lophita Y Napitupulu
Registration Number	: 5113 100 053
Department	: Informatics Engineering, FTIf ITS
First Supervisor	: Dr. Eng. Nanik Suciati, S.Kom., M.Kom.
Second Supervisor	: Dini Adni Navastara, S.Kom., M.Sc.

Abstract

Epilepsy is a long-term neurological disorder characterized by epileptic seizures. Epileptic seizures can occur in a short period of time until a strong shock for a long time. Epilepsy is a disease that tends to occur repeatedly and cannot be healed, but epileptic seizures that occur can be controlled through treatment.

In this undergraduate thesis, the electroencephalogram (EEG) record data will be divided into several windows using segmentation or decomposition. The next process is to extract each window by using a Weighted Permutation Entropy that produces a feature of each window. The feature will be tested using k-fold cross-validation by dividing data into training and testing data. Furthermore data is classified using Support Vector Machine. The EEG record data used for testing in this experiment was taken from the online data collected by 500 online '' Clinical für Epileptologie, Universität Bonn '. This data consists of epileptic seizure (set S) and seizure free (set Z, N, O, F) each of which consists of 100 data. Set Z was recorded from five healthy people when eyes are closed and set O recorded from five healthy people when eyes are opened. Set F was recorded from five epilepsy patients during seizure free in hippocampal formation, N sets was recorded from five epilepsy patients during seizure free in epileptogenic zone, and set S was

recorded from five epilepsy patients during seizure in epileptogenic zone.

The trial test use set S combined with every other set. So the data used are 200 EEG record data for each test. Based on the trials, the proposed method above gave an average accuracy of 91.88%.

Keywords: Epilepsy, EEG, Weighted Permutation Entropy, Support Vector Machine.

KATA PENGANTAR

Segala puji syukur kehadirat Tuhan Yang Maha Esa yang telah melimpahkan rahmat dan anugerah-Nya sehingga penulis dapat menyelesaikan tugas akhir yang berjudul "**Implementasi Deteksi Serangan Epilepsi dari Data Rekaman EEG Menggunakan Weighted Permutation Entropy dan Support Vector Machine**".

Buku tugas akhir ini disusun dengan harapan dapat memberikan manfaat dalam bidang kesehatan. Selain itu, penulis berharap dapat memberikan kontribusi positif bagi kampus Teknik Informatika ITS.

Selama pengerjaan tugas akhir ini, penulis mendapatkan pengalaman dan pengetahuan berharga. Penulis dapat memperdalam dan meningkatkan pengetahuan yang telah didapatkan selama kuliah di Teknik Informatika ITS. Dalam pengerjaan tugas akhir ini, penulis mendapat bantuan dan dukungan dari berbagai pihak. Oleh karena itu, penulis ingin mengucapkan terima kasih kepada:

1. Ibu Dr. Eng. Nanik Suciati, S.Kom., M.Kom. dan Ibu Dini Adni Navastara, S.Kom., M.Sc. selaku dosen pembimbing yang telah membimbing dan membantu penulis dalam menyelesaikan tugas akhir ini dengan tepat waktu.
2. Orang tua dan saudara/i yang telah memberikan kepercayaan, dukungan, doa, motivasi, perhatian, dan keuangan selama penulis kuliah hingga menyelesaikan tugas akhir ini.
3. Teman baik penulis yaitu, Devira, Ayu, Asri, Putri, Eriko, Franky, Hari, Gian, Budi dan Saddam yang telah menemani penulis melewati masa-masa sulit dan teman berbagi cerita.
4. Teman-teman Teknik Informatika angkatan 2013 yang bersama-sama selama empat tahun melalui masa perkuliahan di Teknik Informatika.

5. Teman-teman TA RMK KCV yang telah melalui suka duka bersama dan memberikan dukungan serta bantuan selama mengerjakan tugas akhir ini.
6. Teman-teman seperantauan yaitu, Renova, Dorlinca, Marissa, Yuni, Clarissa, Habema, Suhunan, Niko, Ramosan, dan Tomson yang memberikan dukungan, canda tawa, dan saling berbagi pengalaman selama berkuliah di ITS.
7. Teman-teman se-kost penulis, Belli dan Brigita, yang telah memberikan canda tawa dan kenyamanan selama kost.
8. Pihak-pihak lain yang tidak bisa penulis sebutkan satu-persatu.

Penulis menyadari masih ada kekurangan dalam penyusunan tugas akhir ini. Penulis mohon maaf atas kesalahan dan kekurangan dalam penyusunan tugas akhir ini. Kritik dan saran yang membangun dapat disampaikan sebagai bahan perbaikan ke depan.

Surabaya, Juni 2017

Penulis

DAFTAR ISI

[*Halaman ini sengaja dikosongkan*]

DAFTAR GAMBAR

DAFTAR TABEL

[*Halaman ini sengaja dikosongkan*]

DAFTAR KODE SUMBER

[*Halaman ini sengaja dikosongkan*]

BAB I
PENDAHULUAN

Pada bab ini dibahas hal-hal yang mendasari tugas akhir. Bahasan meliputi latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, metodologi, dan sistematika laporan tugas akhir.

1.1 Latar Belakang

Epilepsi adalah gangguan neurologis jangka panjang yang ditandai dengan adanya serangan-serangan epileptik [1]. Serangan epileptik bisa bermacam-macam mulai dari serangan singkat dan hampir tidak terdeteksi hingga guncangan kuat dalam waktu yang lama. Epilepsi cenderung terjadi secara berulang dan tidak ada penyebab yang mendasari secara langsung. Dalam kebanyakan kasus, penyebab epilepsi ini tidak dapat diketahui, walaupun beberapa orang menderita epilepsi karena cedera dari aktivitas sel saraf kortikal yang berlebihan dan tidak normal di dalam otak. Epilepsi tidak bisa disembuhkan, namun serangan-serangan dapat dikontrol melalui pengobatan pada sekitar 70% kasus [2].

Elektrosefalografi (EEG) adalah rekaman aktivitas listrik sepanjang kulit kepala yang dihasilkan oleh penembakan neuron dalam otak. EEG mengacu pada rekaman spontan listrik aktivitas otak selama periode tertentu. Dalam dunia neurologi, diagnosis menggunakan EEG digunakan untuk mendeteksi penyakit epilepsi. Pemeriksaan EEG dapat membantu memberikan gambaran aktivitas otak yang menunjukkan peningkatan risiko terjadinya serangan epilepsi.

Pemantauan serangan epilepsi atau kejang secara tradisional dilakukan oleh neurofisiolog dengan meninjau dan menganalisis data rekaman EEG. Namun, menemukan serangan epilepsi dalam rekaman EEG akan memakan waktu dan melelahkan. Pada penelitian sebelumnya [3], melibatkan 4 ahli untuk menemukan serangan epilepsi dalam 8 jam rekaman EEG. Setiap ahli memberikan hasil pemantauan yang sedikit berbeda-beda. Kejang

yang dialami penderita dapat menyebabkan hilangnya kesadaran, gangguan pada perasa dan fungsi kognitif. Penderita epilepsi juga cenderung mengalami masalah psikologis dan tingkat kematian dini tiga kali lebih tinggi [4]. Untuk mengontrol serangan-serangan yang terjadi, maka perlu dilakukan pengobatan yang cepat sehingga penderita tidak mengalami guncangan yang merugikan dirinya sendiri.

Seiring berkembangya teknologi yang dapat mempercepat pekerjaan manusia, penggunaan teknologi di bidang kesehatan pun sudah dilakukan. Penerapan teknologi untuk mendeteksi berbagai penyakit termasuk penyakit epilepsi telah dilakukan. Mengingat perlunya mendeteksi serangan epilepsi dan pengobatan secara cepat, beberapa penelitian telah dilakukan. Penelitian yang dilakukan menggunakan metode berbeda-beda, diantaranya menggunakan algoritma *permutation entropy* sebagai ekstraksi fitur dan *Support Vector Machine* sebagai metode klasifikasi yang menghasilkan akurasi rata-rata sebesar 86,10% [5] dan menggunakan algoritma *time frequency analysis* sebagai ekstraksi fitur dan *Artificial Neural Network* sebagai metode klasifikasi yang menghasilkan akurasi rata-rata sebesar 89,00% [6].

Pada tugas akhir ini dilakukan pengembangan teknologi untuk mendeteksi serangan epilepsi menggunakan *Weighted Permutation Entropy* (WPE) dan *Support Vector Machine* (SVM). WPE adalah adalah modifikasi *Permutation Entropy* yang mengukur kompleksitas dan ketidakteraturan *time series* dengan menggabungkan pola permutasi dan amplitudo setiap sampel. Proses pengembangan sistem dilakukan dengan beberapa tahap yaitu, *segmentation/pre-processing* data, ekstraksi fitur menggunakan WPE, dan klasifikasi menggunakan SVM. Data masukan awal yang digunakan adalah data rekaman EEG otak manusia yang direkam dari manusia normal, penderita yang tidak mengalami serangan epilepsi dan penderita yang sedang mengalami serangan epilepsi. Tugas akhir ini diharapkan dapat memberikan hasil deteksi serangan epilepsi yang baik.

1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam tugas akhir ini dapat dipaparkan sebagai berikut:

1. Bagaimana melakukan ekstraksi fitur data rekaman EEG dengan meggunakan metode *Weighted Permutation Entropy* (WPE)?
2. Bagaimana melakukan klasifikasi dengan menggunakan metode *Support Vector Machine* (SVM) untuk mendeteksi serangan epilepsi?
3. Bagaimana melakukan uji performa perangkat lunak yang akan dibangun?

1.3 Batasan Masalah

Permasalahan yang dibahas dalam tugas akhir ini memiliki beberapa batasan antara lain:

1. Aplikasi yang dibangun menggunakan Bahasa Python 2.7
2. Dataset yang digunakan adalah dataset rekaman EEG manusia normal dan penderita epilepsi yang diambil dari *Klinik fur Epileptologi, Universitat Bonn* sebanyak 500.
3. Ada dua hasil klasifikasi yaitu, serangan epilepsi atau bukan serangan epilepsi.

1.4 Tujuan Tugas Akhir

Tujuan tugas akhir ini adalah untuk mengimplementasikan sistem deteksi serangan epilepsi dari data rekaman EEG menggunakan *Weighted Permutation Entropy* dan *Support Vector Machine.*

1.5 Manfaat Tugas Akhir

Manfaat yang diperoleh dari tugas akhir ini adalah untuk memudahkan ahli kedokteran dalam proses menemukan serangan epilepsi yang terjadi pada penderita. Semakin cepat diketahui maka semakin cepat penanganan serangan dengan melakukan pengobatan sehingga serangan yang terjadi tidak mengakibatkan guncangan dan gangguan pada tubuh penderita.

1.6 Metodologi

Tahapan-tahapan yang dilakukan dalam pengerjaan tugas akhir ini adalah sebagai berikut:

1. Studi Literatur

   Pada studi literatur ini akan dipelajari sejumlah referensi yang diperlukan dalam pengerjaan deteksi serangan epilepsi pada rekaman EEG yaitu metode *Discrete Wavelet Transform*, EEG *segmentation*, *Weighted Permutation Entropy*, *Support Vector Machine*, dan literatur lain yang menunjang dan berkaitan dengan pembuatan tugas akhir ini.

2. Analisis dan Desain Perangkat Lunak

   Pada tahap ini disusun rancang bangun dari perangkat lunak yang dibangun. Data rekaman EEG sebagai data masukan. Perangkat lunak memproses data rekaman EEG mentah dengan dua cara yaitu, pertama dengan melakukan *segmentation* pada data rekaman EEG mentah. Kedua, melakukan *pre-processing* menggunakan *Discrete Wavelet Transform* pada data rekaman EEG mentah. Kemudian *output* dari proses *segmentation* dan *Discrete Wavelet Transform* diekstraksi menggunakan WPE untuk mendapatkan nilai setiap fitur. Output dari proses ekstraksi fitur diklasifikasi menggunakan *Support Vector Machine*.

3. Implementasi Perangkat Lunak

   Sistem deteksi serangan epilepsi dibuat dengan Bahasa pemrograman python 2.7. *Library* yang digunakan untuk mendukung pengerjaan adalah *matplotlib, numpy, pandas, csv*, dan *sklearn.* Kakas bantu pendukung lain diantaranya *Pycharm Community Edition 2016.2.3* dan *Microsoft Excel* sebagai dokumentasi dan pengolah angka.

4. Uji Coba dan Evaluasi

   Dalam tahap ini, dilakukan pengujian parameter-parameter yang dibutuhkan pada proses ekstraksi fitur dan klasifikasi. Pada proses klasifikasi data hasil ekstraksi fitur dilakukan

proses *training* kemudian dilakukan proses *testing* dengan menghitung nilai akurasi.

1.7 Sistematika Laporan

Buku tugas akhir ini bertujuan untuk mendapatkan gambaran dari pengerjaan tugas akhir ini. Selain itu, diharapkan dapat berguna untuk pembaca yang tertarik untuk melakukan pengembangan lebih lanjut. Secara garis besar, buku tugas akhir terdiri atas beberapa bagian seperti berikut:

Bab I Pendahuluan

Bab yang berisi mengenai latar belakang, tujuan, dan manfaat dari pembuatan tugas akhir. Selain itu permasalahan, batasan masalah, metodologi yang digunakan, dan sistematika penulisan juga merupakan bagian dari bab ini.

Bab II Dasar Teori

Bab ini berisi penjelasan secara detail mengenai dasar-dasar penunjang dan teori-teori yang digunakan untuk mendukung pembuatan tugas akhir ini.

Bab III Analisis dan Perancangan

Bab ini berisi tentang analisis dan perancangan desain sistem deteksi serangan epilepsi.

Bab IV Implementasi

Bab ini membahas implementasi dari desain yang telah dibuat pada bab sebelumnya. Penjelasan berupa kode yang digunakan untuk proses implementasi.

Bab V Uji Coba dan Evaluasi

Bab ini membahas tahap-tahap uji coba. Kemudian hasil uji coba dievaluasi untuk kinerja dari aplikasi yang dibangun.

Bab VI Kesimpulan dan Saran

Bab ini merupakan bab terakhir yang menyampaikan kesimpulan dari hasil uji coba yang dilakukan dan saran untuk pengembangan aplikasi ke depannya.

BAB II
DASAR TEORI

Pada bab ini diuraikan mengenai dasar-dasar teori yang digunakan dalam pengerjaan tugas akhir dengan tujuan untuk memberikan gambaran secara umum terhadap penelitian yang dikerjakan. Bab ini berisi penjelasan mengenai epilepsi, *electroencephalography* (EEG), *segmentation, wavelet transform, Weighted Permutation Entropy* (WPE) untuk ekstraksi fitur, dan *Support Vector Machine* (SVM) untuk klasifikasi.

2.1 Epilepsi

Epilepsi adalah kondisi neurologis jangka panjang dimana aktivitas sel saraf di otak menjadi terganggu yang ditandai dengan adanya serangan-serangan epileptik [1][2]. Serangan epileptik yang terjadi ada yang berlangsung lama dan ada yang berlangsung singkat sampai tak terdeteksi. Epilepsi ditandai dengan kejang berulang, kejang yang terjadi bervariasi dapat berupa sentakan perhatian atau penyumbatan otot yang parah dan berkepanjangan.

Kejang pada penderita epilepsi disebabkan oleh pelepasan listrik yang berlebihan dalam sekelompok sel otak secara tiba-tiba dan biasanya singkat. Karakteristik kejang bervariasi dan bergantung pada letak gangguan otak dimulai dan seberapa jauh penyebarannya. Kejang dapat menyebabkan hilangnya kesadaran, gangguan gerak, gangguan pada perasa seperti penglihatan, pendengaran dan pengecap, mood dan fungsi kognitif lainnya. Penderita epilepsi cenderung memiliki masalah psikilogis seperti kecemasan dan depresi serta tingkat kematian dini tiga kali lebih tinggi [4].

Sekitar 50 juta orang di seluruh dunia mengalami penyakit epilepsi, yang mana penderitanya orang dari segala umur. Ada dua jenis epilepsi, yaitu epilepsi idiopatik yang penyebabnya tidak dapat dikenali dan epilepsi sekunder yang penyebabnya dapat dikenali. Penyebab dari epilepsi sekunder adalah kerusakan pada

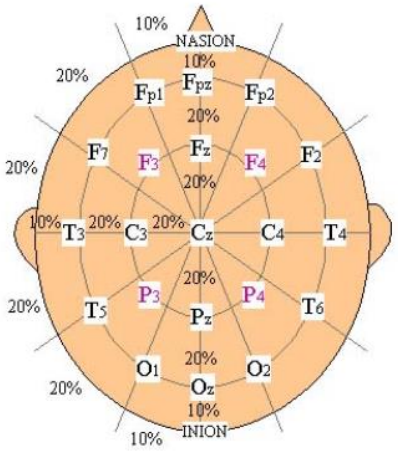
otak akibat cedera, cedera kepala parah, stroke, infeksi otak dan tumor otak.

Dua pertiga pasien epilepsi dapat dikontrol melalui pengobatan dan sisanya resistan terhadap obat dan dirujuk untuk operasi reseksi dimana *Seizure Onset Zone* (SOZ) dihapus. Pasien yang akan melakukan operasi menjalani banyak penyelidikan pra-operasi. Penyelidikan ini dilakukan dengan memantau data rekaman aktivitasi otak jangka panjang yang direkam oleh *Electroencephalographic* untuk dianalisa secara offline. Ahli neurologi mendeteksi aktivitas epilepsi untuk memperoleh daerah kejang dimulai atau fokus kejang sehingga dapat ditentukan apakah pembedahan layak dilakukan [1].

2.2 *Electroencephalography*

Electroencephalograhy (EEG) adalah teknik pencitraan medis yang mengukur aktivitas elektrik kulit kepala yang dihasilkan oleh struktur otak dan mewakili perbedaan potensial dari berbagai lokasi otak manusia. EEG direkam menggunakan elektroda logam dan media konduktif [7]. Sinyal biomedik seperti EEG adalah sinyal non-stasioner yang karakteristiknya berubah-ubah. Sinyal EEG dapat digambarkan oleh fitur *time-frequency* seperti distribusi frekuensi dan analisis *wavelet* [1][8].

Pada EEG konvensial, perekaman diperoleh dengan menempatkan elektroda pada kulit kepala dengan gel atau pasta konduktif. Beberapa sistem menggunakan elektroda yang dilekatkan pada kawat. Beberapa sistem menggunakan topi atau jaring dimana elektroda disematkan. Lokasi elektroda dan label ditentukan oleh sistem internasional 10-20 untuk kebanyakan aplikasi klinis dan penelitian. **Gambar 2.1** menunjukkan label penempatan elektroda pada sistem internasional 10-20. Aktivitas listrik otak manusia bisa dibagi menurut rentang frekuensi. Aktivitas delta (δ) berada pada kisaran 0-4 Hz, theta (θ) berada pada kisaran 4-8 Hz, alpha (α) berada pada kisaran 8-13 Hz, beta (β) berada pada kisaran 13-30 Hz, dan gamma (γ) berada pada kisaran 30-60 Hz [1].



Gambar 2.1 label penempatan elektroda sistem internasional 10-20

2.3 *Segmentation*

Segmentation atau segmentasi adalah proses membagi sinyal menjadi beberapa *window* dengan karateristik sinyal seperti amplitudo dan frekuensi yang sama. Segmentasi biasanya digunakan sebagai tahap *pre-processing* untuk pengolahan sinyal non-stasioner. Hal ini dikarenakan sinyal non-stasioner lebih sulit diolah daripada sinyal stasioner [8].

Ada dua jenis segmentasi sinyal yaitu, *constant segmentation* dan *adaptive segmentation*. *Constant segmentation* membagi sinyal ke dalam *window* dengan panjang yang sama. Kekurangan *constant segmentation* adalah *window* yang dihasilkan tidak stasioner [9]. *Constant segmentation* ada dua jenis yaitu *non-overlapping* dan *overlapping*. *Non-overlapping* membagi sinyal ke dalam *window* tanpa adanya tumpang tindih nilai dalam *window*. Sedangkan *overlapping* membagi sinyal ke dalam *window* dan terjadi tumpang tindih data sinyal antar *window*.

Adapative segmentation membagi sinyal menjadi *window* yang relatif stasioner dan memiliki panjang *window* yang berbeda-

beda. *Adaptive segmentation* secara otomatis menentukan batas *window* terhadap posisi sebenarnya dari transisi antara interval stasioner. Metode ini biasanya berdasarkan estimasi tingkat kesamaan *window* yang tetap dengan *window* yang memiliki panjang yang sama ditentukan oleh waktu sinyal. Nilai indeks kemiripan akan turun tajam jika jendela melewati batas *window*, memberi indikasi untuk pindah *window*.

2.4 *Wavelet Transform*

Wavelet Transform atau disebut Transformasi Wavelet adalah fungsi matematik yang membagi data menjadi beberapa komponen frekuensi yang berbeda, kemudian menganalisis setiap komponen frekuensi dengan resolusi yang disesuaikan dengan skalanya [10]. Fungsi dasar ψ(t), atau disebut juga *mother wavelet* adalah fungsi transformasi. Istilah *wavelet* berarti gelombang yang kecil. Hal tersebut mengarah pada kondisi bahwa fungsi ini memiliki panjang yang terbatas. Istilah *mother* menyiratkan bahwa fungsi dengan daerah pendukung yang berbeda yang digunakan pada proses transformasi berasal dari satu fungsi utama yaitu *mother wavelet*.

Mother wavelet ψ(t) dirumuskan pada persamaan (2.1) sebagai berikut:

$$\psi_{s,\tau}(t) = \frac{1}{\sqrt{s}}\psi\left(\frac{t-\tau}{s}\right) \tag{2.1}$$

Dimana, τ adalah pergeseran ψ sepanjang sinyal/waktu dan *s* adalah scale atau dilasi ψ. Dan $\frac{1}{\sqrt{s_0^j}}$ untuk menormalisasi energi. Fungsi utama dapat digunakan untuk menghasilkan seluruh keluarga wavelet dengan melakukan translasi dan penskalaan pada *mother wavelet*. Parameter translasi adalah τ dan parameter penskalaan adalah *s*. *Wavelet Transform* menggunakan fungsi *mother wavelet* ini dan melakukan dekomposisi sinyal *x(t)* ke dalam rangkaian fungsi *wavelet* skala $\psi(t)$. Keuntungan utama dari penggunaan *wavelet* adalah lokalisasi di dalam ruang. *Wavelet*

ada beberapa jenis diantaranya, *Haar Wavelet*, *Daubechies Wavelet* (db), *Coifmann wavelets* (coiflets), *Shanon Wavelet*, dan *Meyer Wavelet*.

2.4.1 *Continuous Wavelet Transform*

Continuous Wavelet Transform (CWT) adalah salah satu w*avelet transform* yang mengubah sinyal kontinu menjadi sinyal yang sangat redundan dari dua variabel kontinu: translasi (τ) dan penskalaan (*s*) [11]. Fungsi kontinu CWT untuk *time series* x(t) dan *mother wavelet* ψ(t) dirumuskan dengan persamaan (2.2) sebagai berikut:

$$W_{\psi}(s,t) = \int_{-\infty}^{+\infty} x(t)\psi_{s,\tau}^{*}(t)\,dt \qquad (2.2)$$

yang mana $W_{\psi}(s,t)$ adalah koefisien *wavelet transform* dan variabel *t* adalah waktu.

2.4.2 *Discrete Wavelet Transform*

Discrete Wavelet Transform (DWT) adalah salah satu *wavelet transform* yang merepresentasikan sinyal dalam domain waktu dan frekuensi. DWT berasal dari *Continuous Wavelet Transform* dengan koefisien diskrit. Dekomposisi sinyal merepresentasikan fitur sinyal yang berubah secara perlahan di *band* frekuensi rendah, dan sebaliknya, fitur yang berubah dengan cepat pada *band* frekuensi yang lebih tinggi.

Analisis sinyal dengan DWT dilakukan pada frekuensi yang berbeda dengan resolusi yang berbeda pula dengan mendekomposisi sinyal menjadi komponen detail dan komponen aproksimasi. Pada transformasi ini terjadi filterisasi dan *downsampling* yaitu pengurangan koefisien pada fungsi genap. Proses filterisasi dengan melewati *highpass filter* dan *lowpass filter*. Komponen detail merepresentasikan komponen frekuensi tinggi sinyal yang diperoleh dari *highpass filter* dan komponen aproksimasi merepresentasikan komponen frekuensi rendah sinyal yang diperoleh dari *lowpass filter*. Jika dilakukan dekomposisi sinyal pada komponen aproksimasi maka akan diperoleh

komponen wavelet dalam level dekomposisi yang berbeda. Proses dekomposisi ini disebut analisis mutiresolusional *wavelet*. *Downsampling* adalah proses mengurangi laju sampling sinyal.

DWT dan CWT memiliki rumus yang sama. Perbedaannya, untuk DWT, $s = 2^j$, $\tau = k^* 2^j$, dimana $(j,k) \in Z^2$. Sehingga *mother wavelet* dirumuskan seperti pada persamaan (2.3) sebagai berikut [11]:

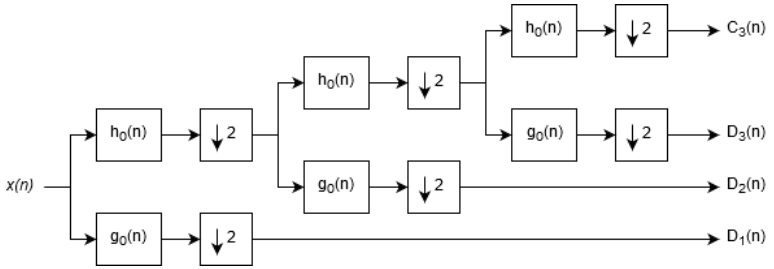
$$\psi_{s,\tau}(t) = \frac{1}{\sqrt{2^j}} \psi \left(\frac{t - k^* 2^j}{2^j} \right) \tag{2.3}$$

Time series x(n) dapat di dekomposisikan menjadi dua *band* frekuensi menggunakan rumus DWT seperti persamaan (2.4) dan (2.5) sebagai berikut.

$$y_h[m] = \sum_n x[n] h_0[2m - n] \tag{2.4}$$

$$y_g[m] = \sum_n x[n] g_0[2m - n] \tag{2.5}$$

Dimana $y_h[.]$ dan $y_g[.]$ adalah output dari *high-pass* dan *low-pass filter* dengan $h_0[.]$ dan $g_0[.]$ sebagai respons impuls setelah sub-sampling. Komponen $y_g[.]$ dapat didekomposisikan dengan filter yang sama. Proses ini memberikan resolusi frekuensi yang baik pada frekuensi rendah.



Gambar 2.2 Dekomposisi 3 level DWT

Gambar 2.2 menunujukkan dekomposisi sinyal dengan 3 level *Discrete Wavelet Transform* dengan 2 *channel filter* rekursif, yaitu *low pass analysis filter* $h_0(n)$ dan *high pass analysis filter* $g_0(n)$ serta operator *down sampling* oleh faktor 2. Sinyal didekomposisi menjadi 4 *subband* frekuensi yaitu satu koefisien aproksimasi $C_3(n)$ dan 3 koefisien detail, $D_3(n)$, $D_2(n)$, dan $D_1(n)$.

2.5 *Permutation Entropy*

Permutation Entropy (PE) diperkenalkann oleh Christoph Bandt dan Bernd Pompe [12]. *Permutation Entropy* adalah metode yang memperkirakan kompleksitas parameter untuk deret waktu (*time series*) berdasarkan perbandingan nilai-nilai tetangga dari setiap titik/nilai dan memetakannya ke dalam pola permutasi tertentu. Metode ini digunakan untuk mendefenisikan pengukuran kompleksitas deret waktu yang mudah dihitung untuk semua tipe deret waktu, baik itu teratur, tidak teratur, dan memiliki *noise*.

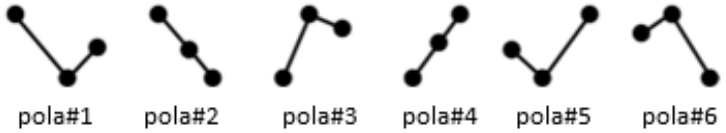
Berdasarkan teori diatas, jika diberikan data *time series* x(t) dimana $t = 1,2,3,\ldots,T$ yang dapat dipetakan ke sekuens berdimensi *m*. Pengambilan nilai-nilai sekuens sebanyak *m* nilai dari *time series* dirumuskan dengan persamaan (2.6) berikut:

$$X_t = \{x_t, x_{t+\tau}, x_{t+2\tau}, \ldots, x_{t+(m-1)\tau}\} \qquad (2.6)$$

PE memiliki dua parameter yaitu *m* dan τ. Parameter *m* merepresentasikan jumlah nilai dalam setiap sekuens. Nilai optimum *m* yang dapat diuji coba adalah 3-8 dan jumlah permutasi *m* (m!) harus lebih kecil dari panjang *time series* [12]. Parameter τ atau *delay time* merepresentasikan jumlah nilai yang dilewati/dilompat oleh setiap nilai dalam sekuens. Dari *time series* dengan panjang *T* diperoleh sejumlah $N = T - (m-1)\tau$ sekuens.

Parameter *m* menentukan jumlah pattern/pola permutasi sekuens yang mungkin terjadi. Jumlah pola permutasi yang mungkin terjadi adalah sejumlah permutasi *m* (*m*!). Jika *m* = 3 maka jumlah pola permutasi adalah 6 dan pola permutasi dalam bentuk angka yaitu, 012, 021, 102, 120, 201, dan 210. Pola

permutasi dalam bentuk sinyal ditunjukkan oleh **Gambar 2.3**. pola#1 merepresentasikan pola angka 201, pola#2 merepresentasikan pola angka 210, pola#3 merepresentasikan pola angka 021, pola#4 merepresentasikan pola angka 012, pola#5 merepresentasikan pola angka 102, dan pola#6 merepresentasikan pola angka 120.



Gambar 2.3 Pola permutasi untuk *m* = 3 [13]

Sekuens sejumlah *N* kemudian akan dicocokkan ke pola permutasi. Nilai sekuens sejumlah *m* akan diurutkan secara *ascending*/meningkat dan index nilai-nilai sekuens setelah diurutkan menjadi pola permutasi. Sekuens yang memiliki pola angka 201 akan memiliki urutan index nilai yaitu $x_{t+2} < x_t < x_{t+1}$ dan pola angka 120 akan memiliki urutan index nilai yaitu $x_{t+1} < x_{t+2} < x_t$.

Sekuens j yang memiliki pola permutasi π_i diberikan nilai 1 dan jika bukan pola permutasi π_i diberikan nilai 0. Probabilitas setiap pola permutasi $P(\pi i)$ dapat dihitung dengan persamaan (2.7) berikut ini:

$$P(\pi i) = \frac{\sum_{j<N} 1_{u:type(u)=\pi_i}(X_j)}{\sum_{j<N} 1_{u:type(u)=\pi}(X_j)} \quad (2.7)$$

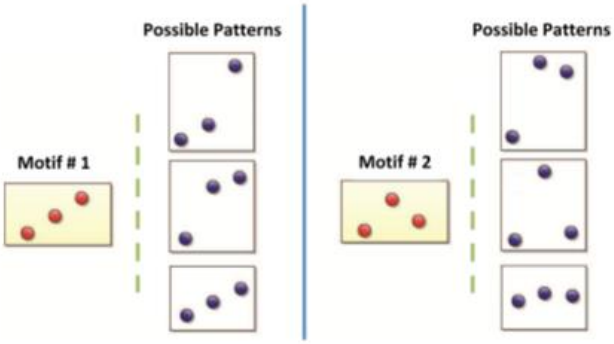
Setelah diperoleh probabilitas setiap pola permutasi, maka nilai entropi $H(m,\tau)$ dari sebuah *time series* dihitung dengan persamaan (2.8) sebagai berikut:

$$H(m,\tau) = -\sum p(\pi_i) \ln p(\pi_i) \quad (2.8)$$

Dimana logaritma (ln) menggunakan basis 2.

2.6 *Weighted Permutation Entropy*

Weighted Permutation Entropy (WPE) diperkenalkan oleh Bilal Fadlallah dan Jos*é* Pr*í*ncipe. Metode WPE merupakan modifikasi dari konsep *Permutation Entropy* (PE). WPE mengubah konsep PE dalam menangani pola permutasi dengan menambahkan informasi amplitudo [14]. PE memiliki kelemahan yaitu, sekuens dengan informasi amplitudo berbeda memiliki pattern/pola permutasi dan nilai PE yang sama. Pola permutasi dari fluktuasi kecil yang disebabkan oleh *noise* seharusnya tidak memiliki nilai PE yang sama. **Gambar 2.4** menunjukkan pola permutasi yang sama dari sekuens berdimensi m = 3 yang berbeda.



Gambar 2.4 Dua contoh sekuens dengan vektor 3 dimensi yang memiliki pattern/pola permutasi yang sama [14]

PE memiliki keterbatasan yaitu tidak dapat menunjukkan perbedaan antara pola permutasi tertentu yang berbeda dan kurangnya sensitivitas terhadap *noise*. Metode ini tergantung pada perbedaan amplitudo dan varians pola permutasi dan proses penentuan bobot/weight setiap vektor yang diekstraksi saat menghitung frekuensi yang berasosiasi dengan setiap pola permutasi. WPE dapat menangani *noise* pada sinyal yang mengalami perubahan mendadak pada *magnitude*. Konsep WPE ini adalah menyimpan informasi amplitudo yang dibawa oleh sinyal.

Berdasarkan konsep PE, jika diberikan data *time series* x(t) dimana $t = 1,2,3,\ldots,T$ yang dapat dipetakan ke sekuens berdimensi *m*. Pengambilan nilai-nilai sekuens sebanyak *m* nilai dari *time series* dirumuskan pada persamaan (2.6). Seperti halnya PE, WPE memiliki dua parameter yaitu *m* dan τ. Kemudian *time series* dibentuk menjadi sekuens berdimensi *m* dan setiap sekuens dipetakan ke pola permutasi yang sesuai.

Sekuens j yang dipetakan terhadap pola permutasi π_i mempunyai bobot/*weight* yang berbeda-beda. Probabilitas setiap pola permutasi merupakan penjumlahan bobot sekuens pada pola yang sama dibagi jumlah keseluruhan bobot sekuens. Probabilitas setiap pola permutasi $P\omega(\pi i)$ dapat dihitung dengan persamaan (2.9) berikut ini:

$$P\omega(\pi i) = \frac{\sum_{j<N} 1_{u:type(u)=\pi_i}(X_j).\omega_j}{\sum_{j<N} 1_{u:type(u)=\pi}(X_j).\omega_j} \tag{2.9}$$

Untuk menghitung bobot sekuens, setiap komponen nilai sekuens dibandingkan dengan varians atau energi dari sekuens tetangga. Varians atau energi $\overline{X}_j$ dapat dihitung dengan persamaan (2.10) sebagai berikut:

$$\overline{X}_j = \frac{1}{m} \sum_{t=1}^{m} x_{j+(t+1)\tau} \tag{2.10}$$

Kemudian bobot sekuens ω_j dihitung dengan menggunakan persamaan (2.11) sebagai berikut:

$$\omega_j = \frac{1}{m} \sum_{t=1}^{m} (x_{j+(t+1)\tau} - \overline{X}_j)^2 \tag{2.11}$$

Setelah diperoleh probabilitas setiap pola permutasi menggunakan persamaan (2.9), maka entropi dari *time series* dihitung dengan persamaan (2.12) sebagai berikut.

$$H\omega(m,\tau) = -\sum P_\omega(\pi_i) \ln P_\omega(\pi_i) \tag{2.12}$$

Dimana logaritma (ln) menggunakan basis 2. Untuk menormalisasi nilai WPE dalam rentang nilai 0 sampai 1, maka digunakan 1/ln(m!) ke rumus *Weighted Permutation Entropy*, sehingga persamaan (2.12) dapat dituliskan menjadi persamaan (2.13) berikut:

$$\mathrm{H}\omega(\mathrm{m},\tau) = -\frac{1}{\ln(m!)}\sum P_{\omega}(\pi_{\mathrm{i}})\ln P_{\omega}(\pi_{\mathrm{i}}) \qquad (2.13)$$

2.7 *Support Vector Machine*

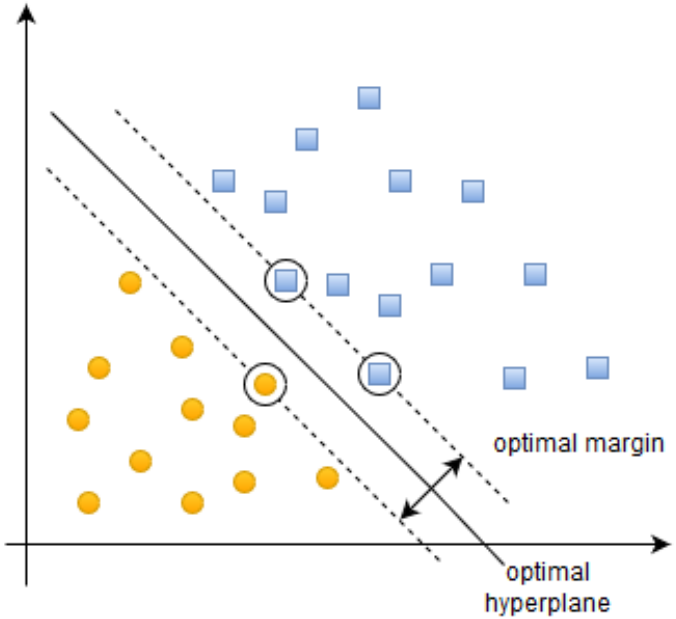
Support Vector Machine (SVM) adalah metode *learning machine* yang bekerja berdasarkan prinsip *Structural Risk Minimization* (SRM). SVM menyediakan metode klasifikasi yang dibangun dengan baik dan *powerful* untuk menganalisis data dan menemukan pemisah antar kelas yang berbeda dengan resiko yang minimal [15]. SVM diperkenalkan oleh Vapnik pada tahun 1995. Beberapa aplikasi yang dibangun berdasarkan algoritma SVM diantaranya yaitu, *time series forecasting*, pengenalan tulisan tangan, prediksi kebangkrutan, identifikasi dan pengenalan wajah, serta untuk tujuan biologis dan medis.

Pada algoritma SVM, model klasifikasi dibangun dengan memetakan data training menjadi kelas yang berbeda yang dipisahkan oleh *hyperplane*. Usaha untuk mencari *hyperplane* pemisah terbaik antara kedua kelas dapat ditentukan dengan menemukan *margin* maksimal. *Margin* adalah jarak antar *support vector* kelas yang berbeda. Sedangkan *support vector* adalah data *training* yang paling dekat dengan *hyperplane* [16]. **Gambar 2.5** menunjukkan contoh pemisahan data *training* ke dalam dua kelas dalam ruang dimensi. Data *training* yang menjadi *support vector* pada **Gambar 2.5** berada di dalam lingkaran hitam.

Diberikan sebuah data seperti pada persamaan (2.14):

$$(y_1, x_1),\ \ldots, (y_l, x_l)\ ;\ y_i \in \{1, -1\} \qquad (2.14)$$

dimana *y* adalah variabel label kelas, *x* adalah variabel data fitur. data dapat dipisahkan secara *linear* jika ada vektor *w* dan skalar *b*



Gambar 2.5 Contoh pemisahan kelas dalam ruang dua dimensi.

sehingga pertidaksamaan (2.15) dan (2.16) valid untuk semua data pada persamaan (2.14).

$$w.x_i + b \geq 1 \qquad if\ y_i = 1, \tag{2.15}$$

$$w.x_i + b \leq -1 \qquad if\ y_i = -1, \tag{2.16}$$

Pertidaksamaan (2.15) dan (2.16) dapat disederhanakan seperti pada pertidaksamaan (2.17) berikut ini:

$$y_i(w.x_i + b) - 1 \geq 0 \qquad i = 1,2,\ldots,l \tag{2.17}$$

Hyperplane yang optimal dirumuskan pada persamaan (2.18) sebagai berikut:

$$w_0.x + b_0 = 0 \tag{2.18}$$

Hyperplane yang optimal diperoleh dengan memaksimalkan *margin*. *Margin* dirumuskan oleh $\frac{2}{||w||^2}$. Untuk memperoleh *margin* maksimal adalah dengan meminimasi $||w||$ dan mempertimbangkan konstrain pertidaksamaan (2.17) yang diperoleh dengan *Quadratic Programming* (QP). Masalah ini dapat dipecahkan dengan teknik komputasi *Lagrange Multiplier* [16] dan dirumuskan dalam persamaan (2.19) berikut ini:

$$L(w, b, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{l} \alpha_i^0 [y_i(x_i . w + b - 1)] \qquad (2.19)$$

dimana $\alpha_i^0 \geq 0$ dan $\alpha > 0$ berlaku untuk *support vector*. Nilai optimal dari persamaan (2.18) dapat dihitung dengan meminimlkan nilai L terhadap *w* dan *b*, dan memaksimalkan nilai L terhadap α_i. Vektor w_0 yang menentukan *hyperplane* yang optimal merupakan kombinasi *training vector* yang dirumuskan pada persamaan (2.20) berikut ini:

$$w_0 = \sum_{i=1}^{l} y_i \alpha_i^0 x_i \qquad i = 1,2,\ldots,l \qquad (2.20)$$

2.7.1 *Soft Margin Hyperplane*

Ada kasus dimana data *training* tidak dapat dipisahkan tanpa *error* sehingga kedua kelas tidak terpisah secara sempurna oleh *hyperplane*. Pada kasus ini data *training* akan dipisahkan dimana jumlah nilai *error* paling minimal. Pendekatan untuk perbaikan klasifikasi SVM adalah dengan *soft margin*. *Soft margin* memiliki variabel *slack* ξ yang mengukur derajat misklasifikasi untuk setiap data *training*. Fungsi penalti ditambahkan sebagai jumlah variabel *slack*, untuk memberi penalti pada data yang misklasifikasi dan membuat *trade-off* antara generalisasi dan *error* ketika mencari *hyperplane* pemisah terbaik. Penalti pada data yang salah klasifikasi dapat dikontrol dengan nilai konstan *C*. Bobot penalti *C* lebih lanjut akan memberi nilai penalti pada data training yang berbeda dengan proporsi yang berbeda. Fleksibilitas ini dapat dengan mudah mengarah pada model yang *over-fit* [15].

Untuk membangun *soft margin hyperplane* pemisah dengan memaksimalkan fungsi pada persamaan (2.20) berikut:

$$\Phi = \frac{1}{2} w.w + C\left(\sum_{i=1}^{l} \xi_i\right) \quad (2.21)$$

dan dengan mempertimbangkan konstrain pada persamaan (2.21) dan persamaan (2.22) berikut:

$$y_i(w.x_i + b) - 1 \geq 0 - \xi_i, \qquad i = 1,2,\ldots,l \quad (2.22)$$

$$\xi_i \geq 0, \qquad i = 1,2,\ldots,l \quad (2.23)$$

Nilai *C* yang tinggi akan memberi penalti yang luas pada data yang misklasifikasi sehingga generalisasi model menjadi buruk. Nilai $C = \infty$ akan mengarah pada *hard-margin*. Sebaliknya nilai C yang rendah akan memberikan penalti yang ringan pada data yang misklasifikasi sehingga hasil pemisahan menjadi keliru [15].

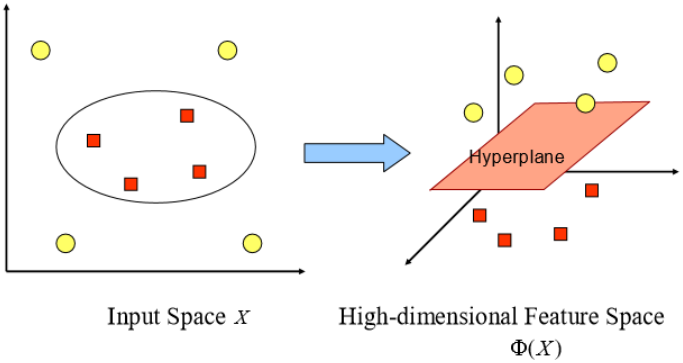
2.7.2 Fungsi Kernel

Ada masalah di dunia nyata yang tidak dapat dipisahkan secara *linear*. Fungsi SVM dimodifikasi dengan memasukkan fungsi kernel. Untuk kasus *non-linear*, fungsi kernel digunakan untuk memetakan *input space X* ke ruang fitur berdimensi yang lebih tinggi *R*, dimana data dapat lebih mudah dipisahkan. Pada ruang vektor *R*, *hyperplane* yang memisahkan kedua kelas dikonstruksikan [15]. **Gambar 2.6** menunjukkan data pada *input space X* berdimensi dua tidak dapat dipisahkan secara *linear*. Kemudian data dipetakan ke ruang fitur berdimensi lebih tinggi yaitu dimensi tiga. Sehingga data dapat dipisahkan secara *linear* oleh sebuah *hyperplane* [17].

Ada beberapa kernel yang digunakan pada SVM yaitu, *linear kernel, polynomial kernel, Radial Basis Function kernel* (RBF), dan *sigmoid kernel*. Kernel optimal yang digunakan untuk setiap masalah tergantung pada masalah klasifikasi dan data yang tersedia, dan setiap kernel memiliki parameter tertentu yang dapat dikontrol untuk mendapatkan performa terbaik [15]. Fungsi kernel adalah fungsi yang hanya berhubungan dengan melakukan

perkalian *dot product* pada data [1]. Hyperplane yang optimal menggunakan fungsi kernel dirumuskan dengan persamaan (2.24) berikut ini:

$$f(x) = \sum_{i=1}^{l} y_i \alpha_i K(x_i, x) + b \qquad (2.24)$$



Gambar 2.6 *Input space X* dipetakan ke ruang fitur berdimensi tinggi [17]

Linear kernel atau kernel *linear* adalah fungsi kernel yang paling sederhana. Fungsi kernel *linear* dirumuskan pada persamaan (2.25), dimana tidak ada parameter yang perlu untuk dioptimalkan.

$$K(x, x') = (x, x') \qquad (2.25)$$

Polynomial kernel atau kernel polinomial adalah kernel non-stationary. Kernel ini cocok digunakan apabila data dinormalisasi. Parameter yang harus dikontrol pada kernel ini adalah parameter gamma (σ), nilai konstan r dan polynomial degree d (dimana default d=3 dan r=0) [18]. Fungsi kernel polinomial dirumuskan pada persamaan (2.26) sebagai berikut:

$$K(x, x') = (\sigma(x, x') + r)^d , \qquad \sigma > 0 \qquad (2.26)$$

Radius Basis Function (RBF) atau *Gaussian kernel* adalah kernel yang pengukuran *distance* diratakan oleh fungsi *radial* atau fungsi eksponensial. Kernel ini secara *non-linear* memetakan data ke ruang berdimensi lebih tinggi, tidak seperti kernel *linear*, dapat mengatasi kasus ketika hubungan antara kelas label dan fitur tidak *linear*. Fungsi kernel RBF dirumuskan pada persamaan (2.27) sebagai berikut:

$$K(x, x') = exp(-\sigma\|x - x'\|^2) , \quad \sigma > 0 \tag{2.27}$$

Parameter yang dapat diatur pada kernel ini adalah parameter *C* dan *gamma* (σ). Jika parameter *gamma* terlalu besar maka eksponensial akan berlaku hampir seperti *linear* dan proyeksi ke dimensi yang lebih tinggi akan kehilangan kemampuan *non-linear*. Sebaliknya, jika parameter *gamma* terlalu kecil maka fungsi akan kekurangan aturan dan *hyperplane* sensitif pada *noise* di data *training* [18].

Sigmoid kernel harus memenuhi teorema Mercer dan mengharuskan bahwa kernel harus positif *definite*. Sehingga parameter *gamma* (σ), r harus dipilih secara tepat, jika tidak, hasilnya bisa menurun secara drastis [18]. Fungsi kernel *sigmoid* dirumuskan pada persamaan (2.28) sebagai berikut:

$$K(x, x') = \tanh(\sigma(x, x') + r) \tag{2.28}$$

Dimana *gamma* adalah parameter penskalaan untuk sampel *input* dan r adalah parameter *shifting* yang mengontrol *threshold* pemetaan (default $r = 0$).

2.8 *Confusion Matrix*

Confusion Matrix adalah sebuah matrix yang menunjukkan data klasifikasi aktual dan prediksi. *Confusion Matrix* berukuran *n x n* dimana *n* adalah jumlah kelas yang berbeda [19]. **Tabel 2.1** menunjukkan *confusion matrix* untuk dua kelas, dimana *n* = 2.

Berikut keterangan isi tabel *confusin matrix*:

1. TP adalah jumlah data dari kelas aktual *True* dan diklasifikasikan dengan benar sebagai kelas prediksi Positif.
2. TN adalah jumlah data dari kelas aktual *True* namun diklasifikasikan sebagai kelas prediksi Negatif.
3. FP adalah jumlah data dari kelas *False* namun diklasifikasikan sebagai kelas prediksi Positif.
4. FP adalah jumlah data dari kelas *False* dan diklasifikasikan dengan benar sebagai kelas prediksi Negatif.

Pengukuran performa yang umum digunakan adalah akurasi. Akurasi dapat dihitung dengan persamaan (2.29) berikut ini:

$$Accuracy = \frac{TP+TN}{TP+FN+TN+FP} \quad (2.29)$$

Tabel 2.1 *Confusion Matrix* untuk klasfikasi dua kelas

		Kelas Prediksi	
		Positif	Negatif
Kelas Aktual	True	TP	TN
	False	FP	FN

2.9 *K-Fold Cross-Validation*

Cross-validation adalah metode statistik untuk mengevaluasi dan membandingkan pembelajaran dengan membagi data menjadi dua segmen, satu segmen sebagai data training yang digunakan untuk melatih model klasifikasi satu segmen lain sebagai data testing yang digunakan untuk memvalidasi model [20]. Segmen data training dan data testing harus disilangkan secara berturut-turut sehingga setiap partisi data memiliki kesempatan untuk divalidasi.

Dalam *k-fold cross-validation* data dipartisi menjadi k partisi dengan ukuran yang sama. Kemudian dilakukan iterasi sebanyak k kali untuk melakukan *training* dan *testing*. Pada setiap

iterasi k-1 partisi data digunakan untuk *training* dan satu sisanya digunakan sebagai data *testing*. Dalam *data mining* dan *machine learning k-fold cross-validation* yang sering digunakan adalah k = 10 [20]. Ilustrasi *k-fold cross-validation* ditunjukkan pada **Gambar 2.7** berikut.`

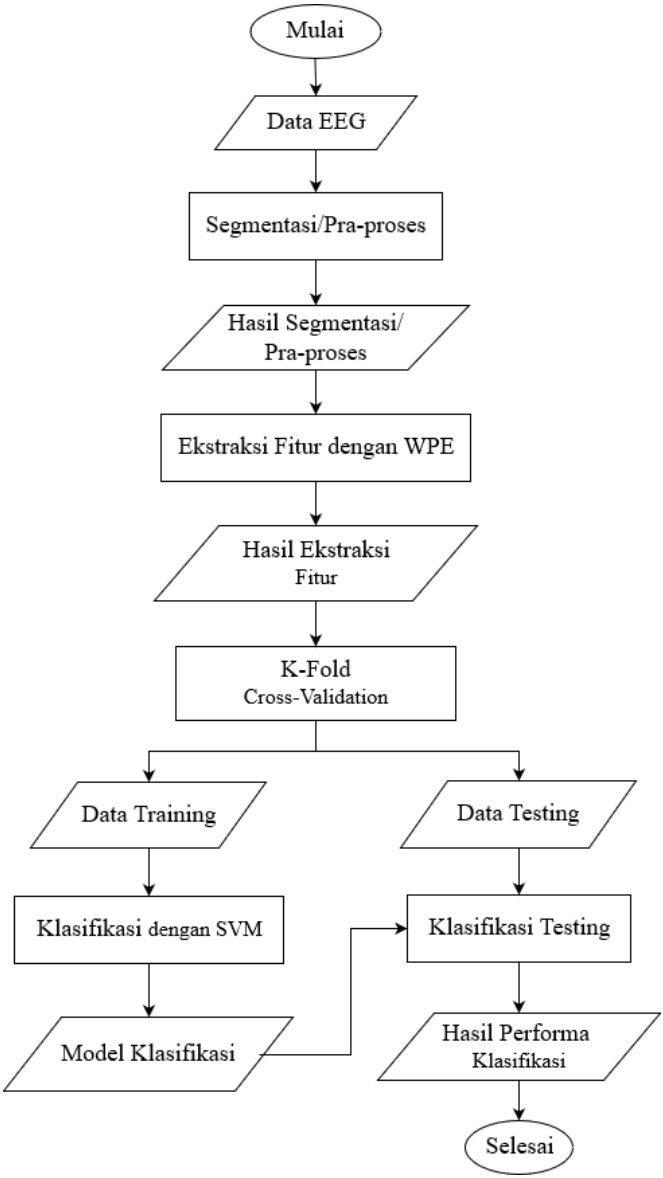
Gambar 2.7 Ilustrasi 10-*fold cross validation*

BAB III
PERANCANGAN SISTEM

Pada bab ini akan dijelaskan mengenai rancangan sistem perangkat lunak yang akan dibuat. Perancangan yang dijelaskan meliputi data dan proses. Data yang dimaksud adalah data yang akan diolah dalam perangkat lunak kemudian digunakan sebagai pembelajaran maupun pengujian sehingga tujuan tugas akhir ini bisa tercapai. Proses yaitu tahap-tahap yang ada dalam sistem sebagai pengolah data meliputi *Segmentation*/*Pre-Processing*, *Weighted Permutation Entropy*, dan *Support Vector Machine*.

3.1 Desain Umum Sistem

Rancangan perangkat lunak yang akan dibuat untuk klasifikasi serangan epilepsi menggunakan *segmentation*/*pre-processing*, *Weighted Permutation Entropy*, dan *Support Vector Machine*. Proses pertama yang dilakukan adalah mengolah data EEG hasil rekaman sinyal otak manusia dengan menggunakan metode *segmentation*/*pre-processing*. Pada tugas akhir ini, ada tiga skenario uji coba pengolahan data awal yaitu *non-overlapping segmentation*, *overlapping segmentation*, dan *pre-processing* dengan *Discrete Wavelet Transform*. *Segmentation* dilakukan untuk membagi data rekaman EEG menjadi beberapa *window* dengan panjang yang sama. *Pre-processing* menggunakan *Discrete Wavelet Transform* yaitu mendekomposisi data rekaman EEG menjadi beberapa *window* sesuai *band* frekuensi. Data EEG didekomposisi menjadi 5 *band* frekuensi sesuai rentang frekuensi aktivitas elektrik otak manusia yaitu, 0-4 Hz, 4-8 Hz, 8-15 Hz, 15-30 Hz, dan 30-60 Hz. Selanjutnya hasil *segmentation* diekstraksi untuk mendapatkan nilai fitur, kemudian data hasil ekstraksi fitur diklasifikasi menggunakan *Support Vector Machine* (SVM). Begitu juga hasil *pre-processing* diekstraksi untuk mendapatkan nilai fitur, kemudian data hasil ekstraksi fitur diklasifikasi menggunakan *Support Vector Machine*.



Gambar 3.1 Diagram alir rancangan perangkat lunak secara umum

Proses ekstraksi fitur pada setiap *window* hasil *segmentation* maupun *pre-procesing*, dilakukan dengan menghitung nilai entropi *window*. Nilai entropi setiap *window* merepresentasikan satu fitur. Sehingga satu data EEG memiliki jumlah fitur sebanyak jumlah *window*. Keluaran dari tahap ekstraksi fitur adalah dataset baru yang akan digunakan untuk tahap selanjutnya yaitu proses klasifikasi menggunakan SVM. Data tersebut dibagi menjadi dua bagian yaitu data *training* dan data *testing*. Data *training* digunakan untuk membuat model klasifikasi dan data *testing* digunakan untuk menguji model klasifikasi. Dari proses klasifikasi tersebut akan diperoleh nilai performa sistem. Data *training* dan data *testing* dilakukan dengan menggunakan metode *k-fold cross-validation*. **Gambar 3.1** menunjukkan diagram alir desain umum perangkat lunak.

3.2 Data

Pada sub bab ini akan dijelaskan mengenai data yang digunakan sebagai masukan perangkat lunak. Data tersebut akan diolah dan dilakukan pengujian sehingga menghasilkan data keluaran yang diharapkan.

3.2.1 Data Masukan

Data masukan adalah data yang digunakan sebagai masukan awal dari sistem. Pada pembuatan perangkat lunak klasifikasi serangan epilepsi, data yang digunakan diiambil dari "Klinik für Epileptologie, Universität Bonn" [21]. Data tersebut berisi lima dataset rekaman EEG yaitu set Z, O, N, F, dan S. Setiap set berisi 100 data sinyal otak *single-channel* yang masing-masing data berdurasi 23,6 detik dan menggunakan frekuensi 173,61 Hz. Setiap data memiliki 4097 nilai.

Set Z dan O direkam dari lima orang sehat, dimana set Z direkam dengan mata terbuka sedangkan set O direkam dengan mata tertutup. Set N, F, dan S direkam dari lima penderita epilepsi. Set N dan F direkam saat penderita tidak mengalami serangan epilepsi. Perbedaan set N dan F adalah lokasi pengambilan data

rekaman EEG. Set N direkam dari bagian *hippocampal formation*, sedangkan set F direkam dari *epileptogenic zone*. Set S direkam dari penderita yang sedang mengalami serangan epilepsi dan diambil dari bagian *epileptogenic zone* [22]. Set S merupakan satu-satunya dataset yang tergolong kelas serangan epilepsi sedangkan set Z, O, N, dan F tergolong kelas bukan serangan epilepsi. Berikut contoh sinyal set Z, O, N, F, dan S ditunjukkan oleh **Gambar 3.2**.

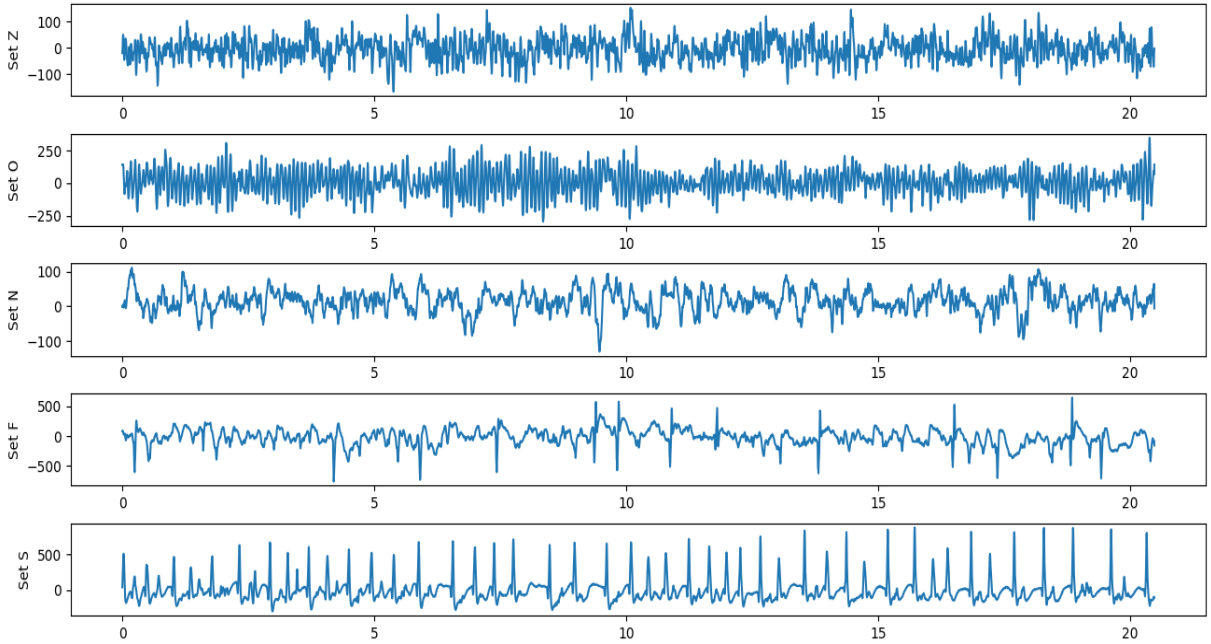
3.2.2 Data Keluaran

Data masukan akan diproses dengan menggunakan *segmentation*/*pre-processing*, *Weighted Permutation Entropy* (WPE), dan *Support Vector Machine* (SVM). Untuk pengolahan awal data rekaman EEG akan diproses menjadi beberapa *window*. Hasil pengolahan awal diekstraksi untuk mendapatkan nilai fitur. Selanjutnya untuk proses klasifikasi, data fitur akan dibagi menjadi dua bagian yaitu data *training* dan data *testing*. Hasil dari proses klasifikasi adalah nilai performa perangkat lunak yang dibangun yaitu nilai *accuracy*.

3.3 *Segmentation*

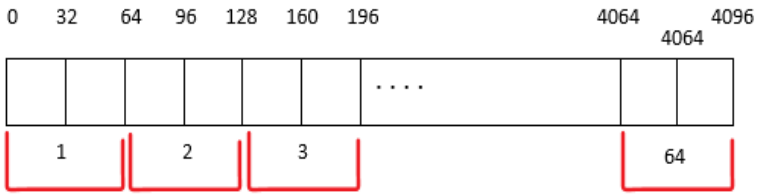
Segmentation atau bisa juga disebut dengan segmentasi merupakan metode untuk membagi data menjadi beberapa *window*. Pada perangkat lunak ini, jenis segmentasi yang akan digunakan adalah *constant segmentation*, dimana data akan dibagi menjadi beberapa *window* dengan panjang data di setiap *window* adalah sama. Pada perangkat lunak ini, proses segmentasi data akan dijadikan sebagai skenario uji coba, dimana data akan dibagi ke dalam *window* secara *non-overlapping* dan *overlapping*.

Pada tahap ini data masukan yang digunakan adalah data EEG yang diperoleh dari website. Setiap data EEG memiliki panjang 4097 nilai dan akan dibagi ke dalam beberapa *window* dengan panjang masing-masing 64 nilai. Untuk segmentasi secara *non-overlapping*, data akan bergeser ke depan sebesar 64 nilai sehingga tidak ada tumpang tindih nilai antar *window*.

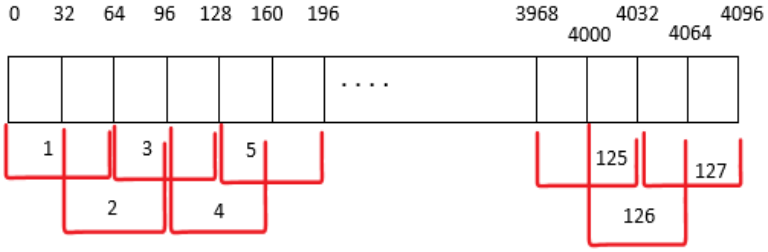


Gambar 3.2 Contoh data masukan masukan setiap set

Untuk segmentasi secara *overlapping* data akan bergeser ke depan sebesar 32 nilai, sehingga sebesar 32 nilai akan tumpang tindih antar *window*. Oleh karena itu, setiap data EEG yang disegmentasi secara *non-overlapping* akan memiliki 64 *window* dan data EEG yang disegmentasi secara *overlapping* akan memiliki 127 *window*. **Gambar 3.3** menunjukkan ilustrasi segmentasi pada setiap data rekaman EEG secara *non-overlapping dan* **Gambar 3.4** menunjukkan ilustrasi segmentasi pada setiap data rekaman EEG secara dan *overlapping*. 5 *window* pertama hasil segmentasi secara *non-overlapping* pada setiap set ditunjukkan oleh **Tabel 3.1**. 5 *window* pertama hasil segmentasi secara *overlapping* pada setiap set ditunjukkan oleh **Tabel 3.2**.

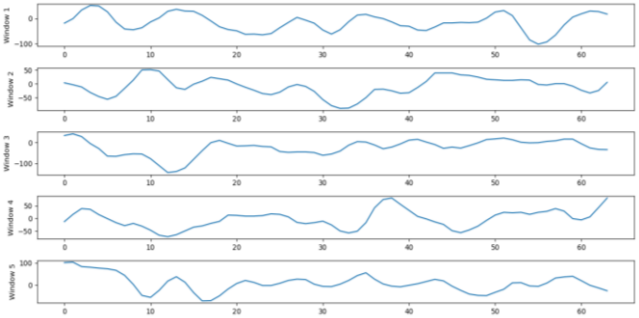
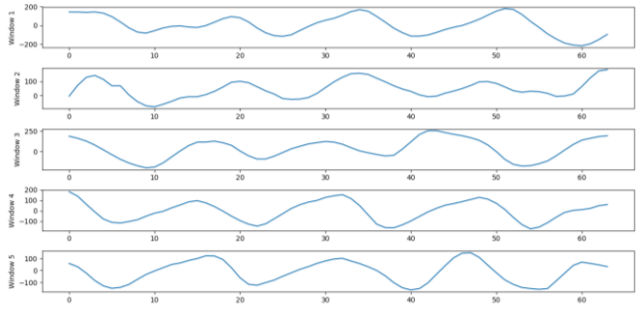
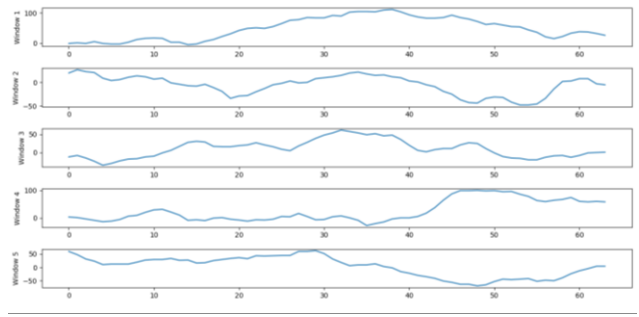


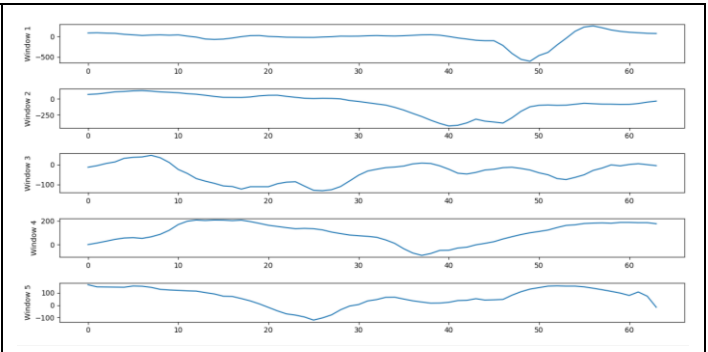
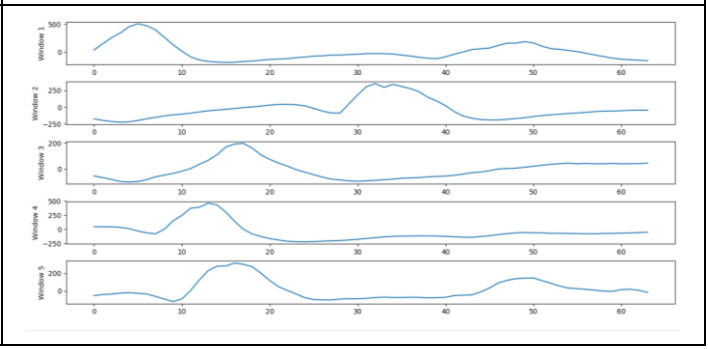
Gambar 3.3 Pembagian *window* segmentasi secara *non-overlapping*



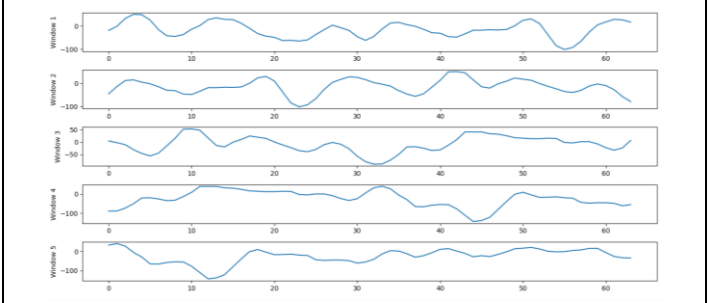
Gambar 3.4 Pembagian *window* segmentasi secara *overlapping*

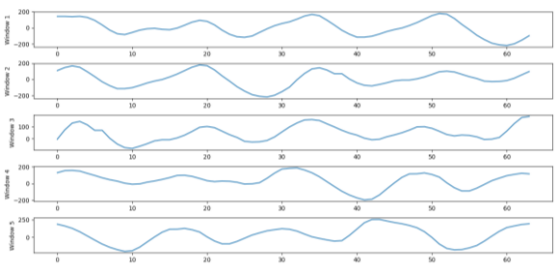
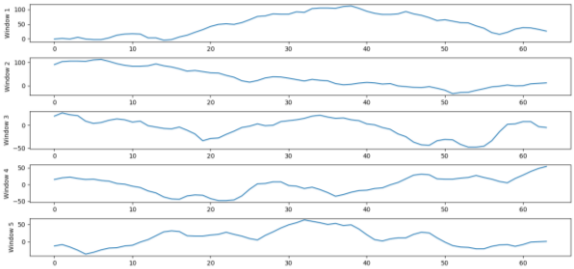
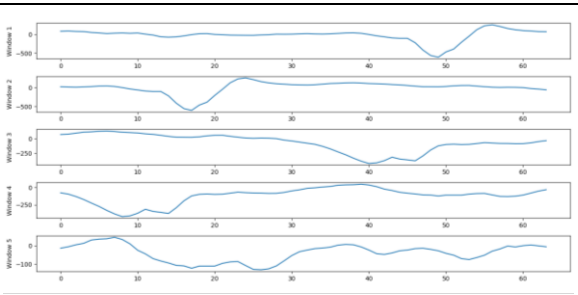
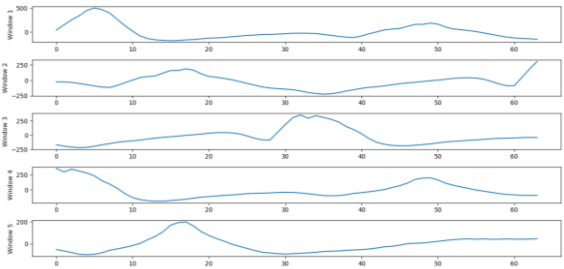
Tabel 3.1 Segmentasi secara *non-overlapping*

Set	Hasil segmentasi secara *non-overlapping*
Z	
O	
N	

F	
S	

Tabel 3.2 segmentasi secara *overlapping*

Set	Hasil segmentasi secara *overlapping*
Z	

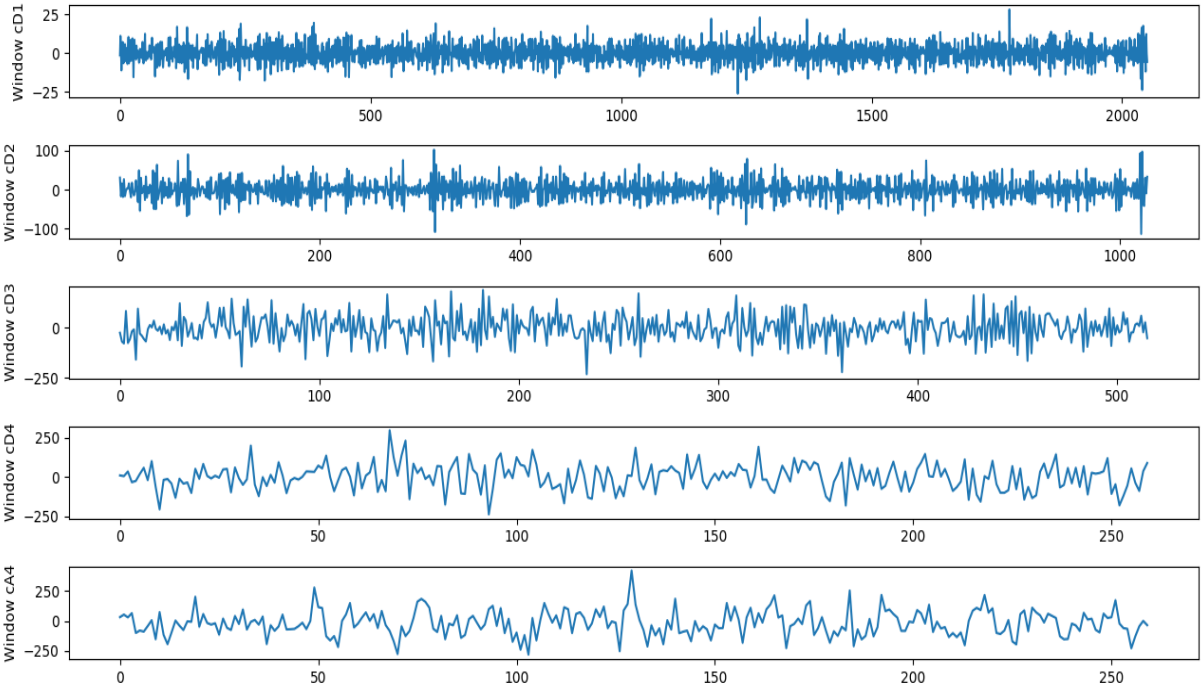
O	
N	
F	
S	

3.4 *Discrete Wavelet Transform*

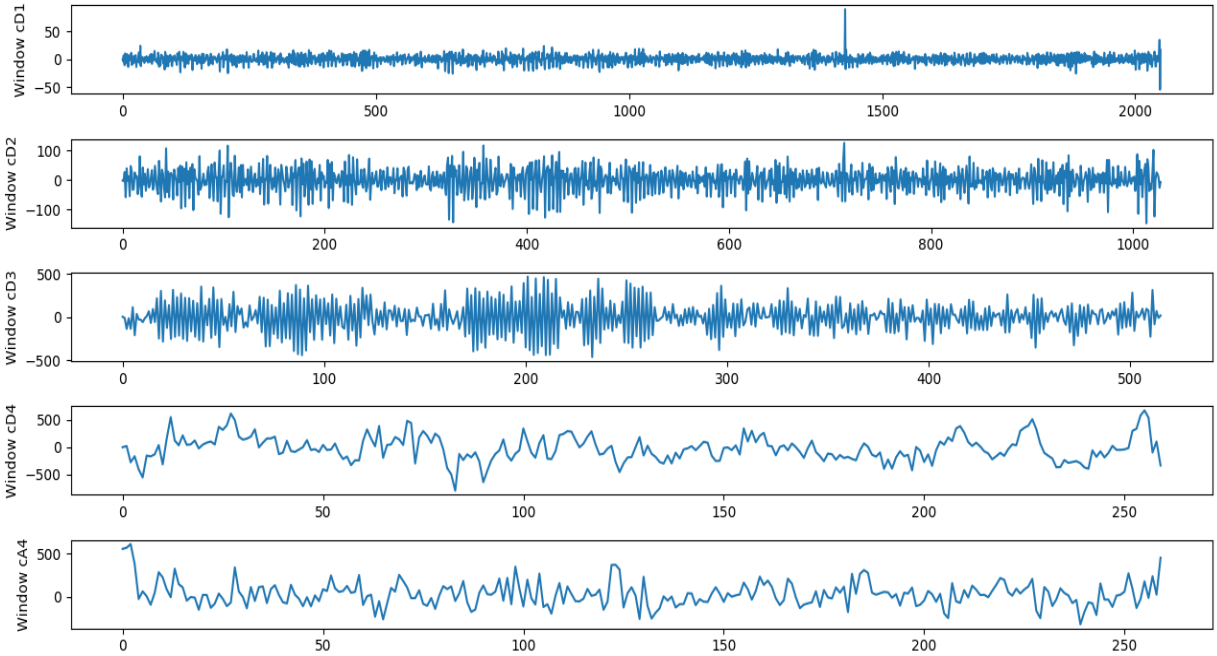
Discrete Wavelet Transform (DWT) merupakan metode untuk mendekomposisi sinyal menjadi dua bagian yang memiliki panjang data yang sama pada tiap levelnya. Dua bagian ini terdiri dari dua *band* frekuensi yang berbeda yang dipisahkan oleh dua jenis filter yaitu *highpass filter* dan *lowpass filter*. DWT melakukan dekomposisi pada *mother wavelet* atau biasa disebut sinyal utama sehingga *lowpass filter* menghasilkan koefisien aproksimasi dan *highpass filter* menghasilkan koefisien detail. Untuk dekomposisi level selanjutnya, koefisien aproksimasi akan berfungsi sebagai *mother wavelet*.

Pada tahap ini data masukan yang digunakan adalah data EEG yang berfungsi sebagai *mother wavelet*. Data EEG ini berada pada rentang 0-60 Hz dan data ini akan dibagi sesuai *band* frekuensi otak manusia yaitu, *delta*, *theta*, *alpha*, *beta*, dan *gamma*. Maka dapat disimpulkan bahwa satu data EEG akan didekomposisi ke dalam 5 *window* sesuai *band* frekuensi yaitu, 0-4 Hz, 4-8 Hz, 8-15 Hz, 15-30 Hz, dan 30-60 Hz.

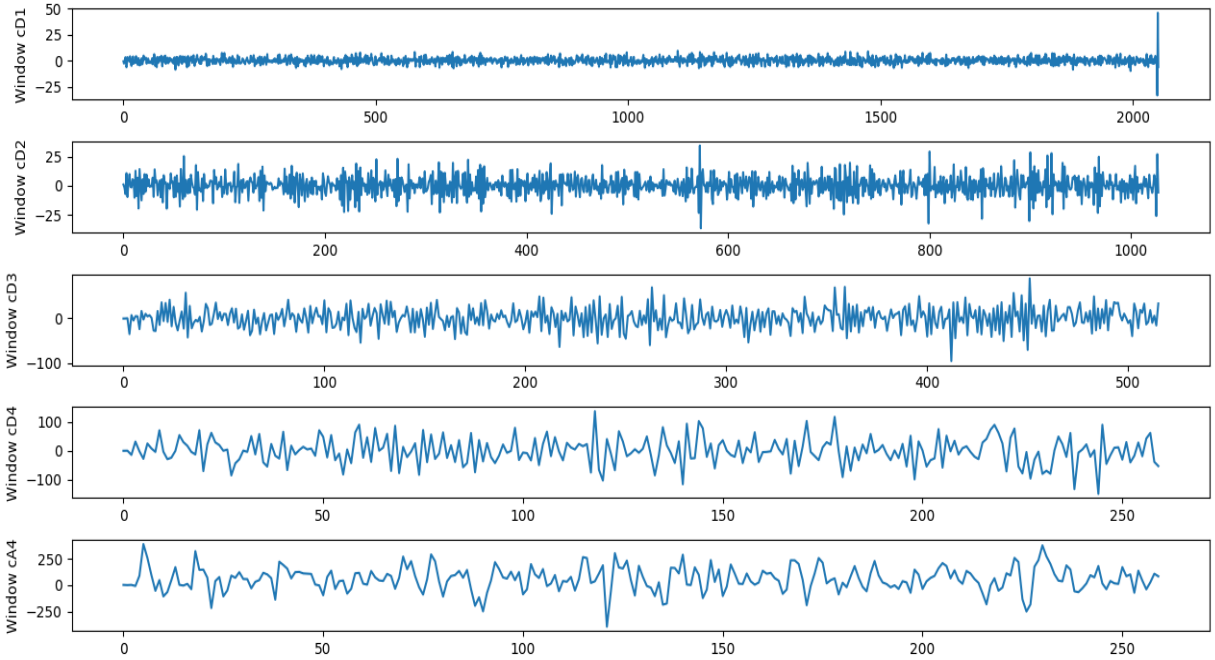
Pada level pertama satu data EEG berfungsi sebagai *mother wavelet* yang menghasilkan koefisien aproksimasi disingkat cA1 (0-30 Hz) dan koefisien detail disingkat cD1 (30-60 Hz). Pada level dua, koefisien aproksimasi dari level pertama akan berfungsi sebagai *mother wavelet* dan akan menghasilkan koefisien aproksimasi cA2 (0-15 Hz) dan koefisien detail cD2 (15-30 Hz) begitu seterusnya sampai level empat. Sehingga di hasil akhir akan diperoleh 5 *window* yaitu *window* satu berisi data cA4, *window* dua berisi data cD4, *window* tiga berisi data cD3, *window* empat berisi data cD2, dan *window* lima berisi data cD1 yang koresponden terhadap *band* frekuensi 0-4 Hz, 4-8 Hz, 8-15 Hz, 15-30 Hz, dan 30-60 Hz secara berurutan. **Gambar 3.5**, **Gambar 3.6**, **Gambar 3.7**, **Gambar 3.8**, dan **Gambar 3.9** menunjukkan koefisien *wavelet* hasil dekomposisi satu data EEG pada 5 window dari setiap set menggunakan 4 level *Discrete Wavelet Transform* dan fungsi *wavelet Daubechies* (db3).



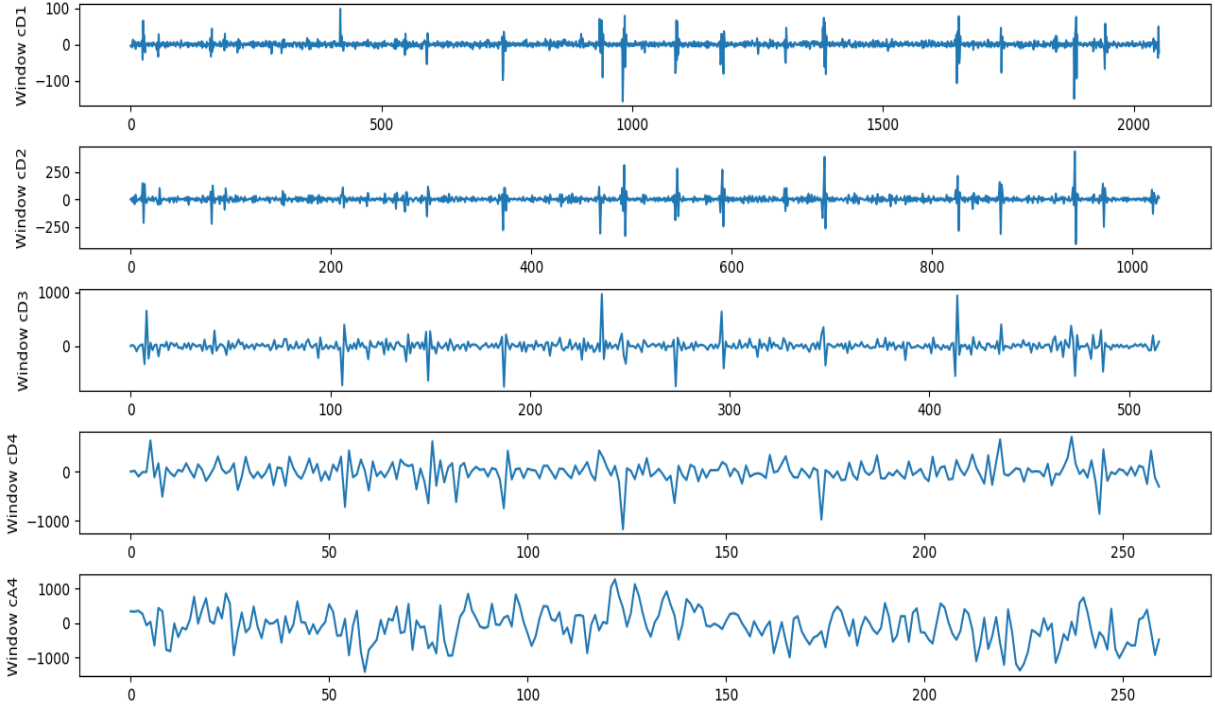
Gambar 3.5 Koefisien *wavelet* untuk sinyal set Z yang didekomposisi menggunakan 4 level DWT



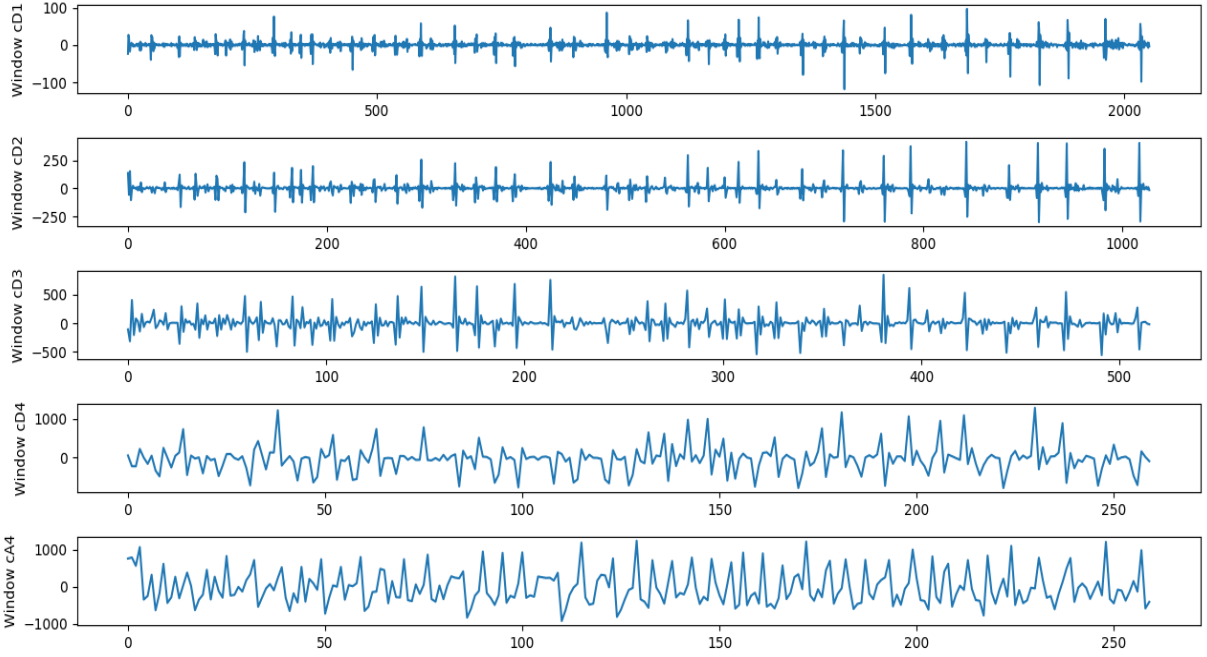
Gambar 3.6 Koefisien *wavelet* untuk sinyal set O yang didekomposisi menggunakan 4 level DWT



Gambar 3.7 Koefisien *wavelet* untuk sinyal set N yang didekomposisi menggunakan 4 level DWT



Gambar 3.8 Koefisien *wavelet* untuk sinyal set F yang didekomposisi menggunakan 4 level DWT



Gambar 3.9 Koefisien *wavelet* untuk sinyal set S yang didekomposisi menggunakan 4 level DWT

3.5 Ekstraksi Fitur dengan *Weighted Permutation Entropy*

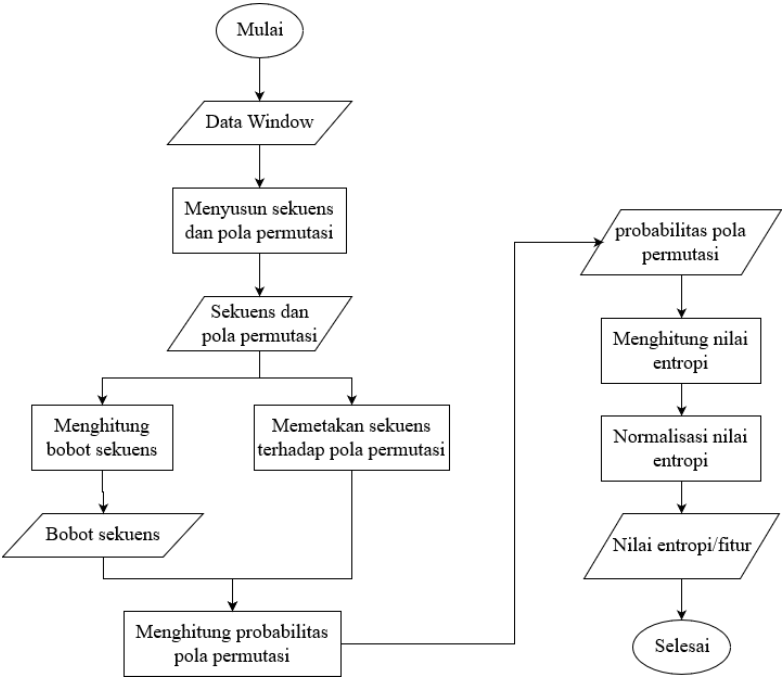
Weighted permutation entropy (WPE) pada tugas akhir ini bertujuan untuk mengekstraksi fitur data EEG. Metode WPE mengukur kompleksitas dan ketidakteraturan *time series* dengan membandingkan nilai tetangga dan amplitudo berbeda pada setiap setiap pola permutasi. WPE memperkirakan distribusi entropi pada kelompok permutasi dari sampel waktu dan perubahan pada amplitudonya. Sehingga variasi WPE sebagai fungsi waktu dapat dengan efektif menunjukkan perubahan pada data yang sesungguhnya.

Pada tahap ekstraksi fitur data masukan yang digunakan adalah data *window* hasil segmentasi atau data *window* hasil dekomposisi *Discrete Wavelet Transform*. Setiap data rekaman EEG yang disegmentasi secara *non-overlapping* memiliki 64 *window*, sedangkan yang disegmentasi secara *overlapping* memiliki 127 *window*, dan yang didekomposisi dengan *Discrete Wavelet Transform* memiliki 5 *window*. Setiap data *window* akan diekstraksi hingga memperoleh satu nilai entropi yang merepresentasikan nilai fitur.

Proses pertama pada tahapan ekstraksi fitur dimulai dengan membagi data *window* menjadi beberapa sekuens, yang mana setiap sekuens memiliki panjang sekuens *m* dan pengambilan nilai dari data *window* ke dalam setiap sekuens tergantung pada nilai *time delay* (τ) seperti dijelaskan sebelumnya dengan menggunakan persamaan (2.6). Kemudian menyusun pola permutasi yang tergantung pada panjang sekuens. Permutasi adalah susunan unsur-unsur yang berbeda dalam urutan tertentu yang disebut dengan pola permutasi. Dalam hal ini, jumlah unsur ditentukan oleh panjang sekuens. Sehingga dapat disimpulkan bahwa panjang sekuens akan memiliki panjang yang sama dengan panjang pola permutasi. Pada proses pertama dihasilkan sejumlah N sekuens berdimensi *m* dan pola permutasi sejumlah permutasi *m* (*m*!) berdimensi *m*.

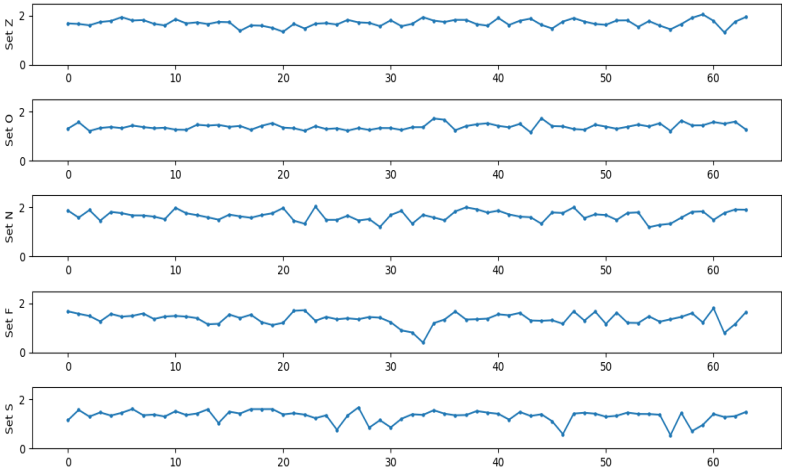
Setiap sekuens memiliki informasi amplitudo yang berbeda dan penting. Sehingga pada proses kedua, nilai informasi

amplitudo atau bobot sekuens akan dihitung. Bobot sekuens diperoleh dengan membandingkan nilai sekuens dengan nilai rata-rata sekuens tetangga seperti dijelaskan sebelumnya dengan persamaan (2.11). Nilai rata-rata sekuens diperoleh dengan persamaan (2.10). Selanjutnya setiap sekuens yang terbentuk dipetakan ke pola permutasi yang sesuai. Sehingga sejumlah N sekuens akan terbagi ke dalam kelompok-kelompok pola permutasi dan setiap sekuens akan berada pada satu pola permutasi. Kemudian probabilitas setiap pola permutasi dihitung dengan menjumlahkan bobot sekuens yang berada dalam kelompok pola permutasi tersebut seperti dijelaskan sebelumnya dengan persamaan (2.9). Pada proses kedua diperoleh hasil akhir berupa nilai distribusi probabilitas setiap pola permutasi.

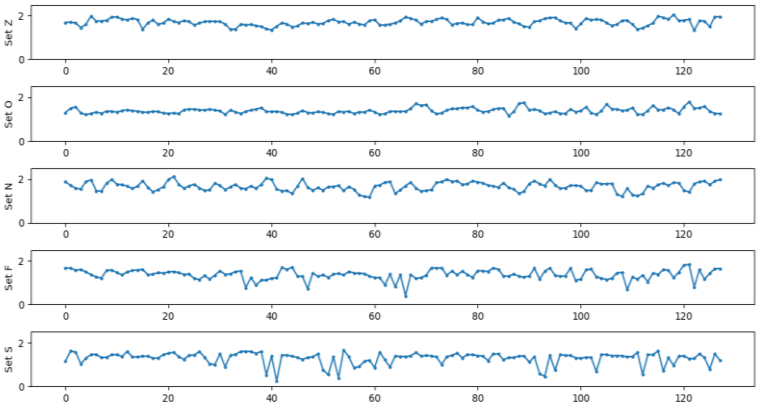


Gambar 3.10 Diagram alir proses ekstraksi fitur

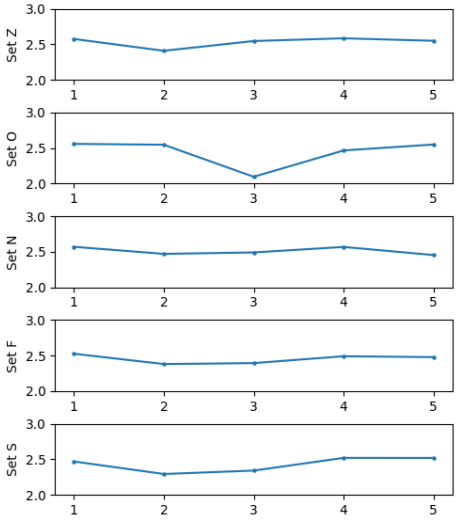
Pada proses terakhir, nilai distribusi probabilitas setiap pola permutasi akan digunakan untuk menghitung nilai entropi. Nilai entropi dihitung dengan menggunakan persamaan (2.12). Nilai entropi yang diperoleh merupakan nilai akhir dari proses ekstraksi fitur satu data *window*. Sehingga hasil ekstraksi fitur pada setiap data *window* merepresentasikan satu nilai fitur. Jumlah fitur setiap data EEG akan sama dengan jumlah *window* yang diperoleh pada pengolahan awal data rekaman EEG. Setiap data rekaman EEG yang disegmentasi secara *non-overlapping* akan memiliki 64 *window* sehingga akan memiliki 64 fitur, sedangkan yang disegmentasi secara *overlapping* akan memiliki 127 *window* sehingga akan memiliki 127 fitur, dan yang didekomposisi dengan *Discrete Wavelet Transform* memiliki 5 *window* sehingga akan memiliki 5 fitur. Garis besar proses ekstraksi fitur pada setiap data window dapat dilihat pada **Gambar 3.10**. **Gambar 3.11**, **Gambar 3.12**, dan **Gambar 3.13** menunjukkan satu data fitur setiap set hasil ekstraksi fitur menggunakan WPE.



Gambar 3.11 Hasil ekstraksi fitur pada data hasil segmentasi *non-overlapping*



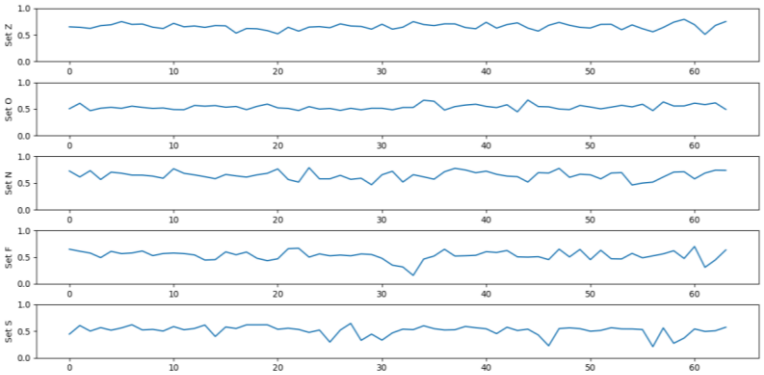
Gambar 3.12 Hasil ekstraksi fitur pada data hasil segmentasi *overlapping*



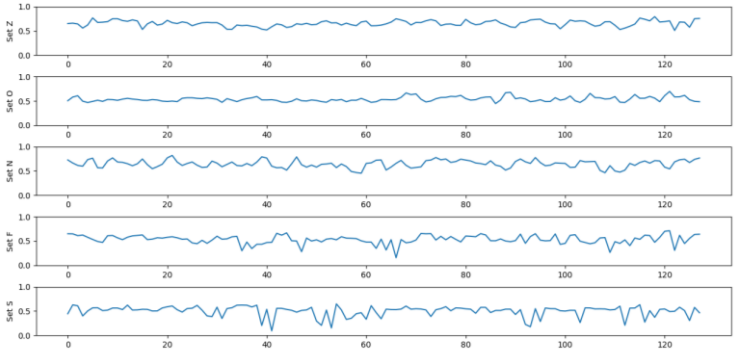
Gambar 3.13 Hasil ekstraksi fitur pada data hasil dekomposisi dengan *Discrete Wavelet Transform*

3.6 Klasifikasi dengan *Support Vector Machine*

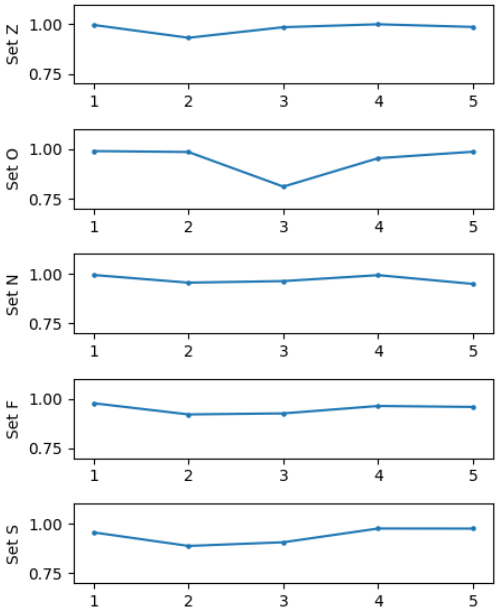
Support Vector Machine (SVM) merupakan salah satu teknik *supervised learning* atau pembelajaran terawasi yang paling umum digunakan, karena sangat efektif untuk banyak masalah terutama klasifikasi dua kelas. Klasifikasi dilakukan dengan cara mencari *hyperplane* terbaik yang memisahkan data menjadi dua kelas berbeda.



Gambar 3.14 Hasil normalisasi data fitur pada segmentasi *non-overlapping*



Gambar 3.15 Hasil normalisasi data fitur pada segmentasi *overlapping*



Gambar 3.16 Hasil normalisasi data fitur pada dekomposisi dengan *Discrete Wavelet Transform*

Data masukan yang digunakan untuk proses klasifikasi adalah data hasil ekstraksi fitur yang telah dinormalisasi dan dikelompokkan menjadi 4 kelompok data yaitu, set Z dan S, set O dan S, set N dan S, serta set F dan S. Sehingga setiap kelompok data klasifikasi akan terdiri dari 200 data fitur. Jumlah fitur data rekaman EEG berbeda-beda tergantung pada proses pengolahan awal data. Data rekaman EEG yang disegmentasi secara *non-overlapping* memiliki 64 fitur, sedangkan yang disegmentasi secara *overlapping* memiliki 127 fitur, dan pengolahan awal dengan dekomposisi dengan *Discrete Wavelet Transform* memiliki 5 fitur. Nilai data fitur yang dihasilkan pada proses ekstraksi fitur memiliki skala yang berbeda dan distribusi nilai yang tidak merata. Oleh karena itu, perlu dilakukan normalisasi nilai data fitur. Normalisasi bertujuan untuk menormalkan nilai data fitur pada

skala yang sama dan menjamin konvergensi berat dan bias stabil. Untuk menormalisasi nilai fitur digunakan persamaan (2.13). **Gambar 3.14**, **Gambar 3.15**, dan **Gambar 3.16** menunjukkan hasil normalisasi satu data fitur setiap set.

Data keluaran yang dihasilkan pada tahap klasifikasi ada dua kelas yaitu serangan epilepsi atau bukan serangan epilepsi.

3.7 Uji Performa

Pengujian tingkat akurasi *clasifier* pada tugas akhir ini menggunakan *k-fold cross-validation*. *K-fold cross-validation* membagi dataset menjadi sebanyak k-buah partisi kemudian melakukan sebanyak k-percobaan *training* dan *testing*. Dalam setiap percobaan, satu *fold* digunakan sebagai data *testing* dan sisanya digunakan sebagai data *training*.

Dalam tugas akhir ini jumlah k yang digunakan adalah sebanyak 10 *fold*. Dataset hasil ekstraksi fitur akan dibagi secara acak menjadi 10 partisi. Pengacakan data ini bertujuan untuk menghindari bias pada data. Kemudian akan dilakukan 10 kali percobaan, dimana pada setiap percobaan 9 partisi data digunakan sebagai data *training* untuk membentuk model *classifier* dan satu sisanya digunakan sebagai data *testing*. Pada setiap testing akan diperoleh akurasi performa model *classifier*. Akurasi akhir dari pengujian ini adalah rata-rata akurasi dari 10 percobaan.

BAB IV
IMPLEMENTASI

Bab ini membahas mengenai implementasi dari perancangan yang sudah dilakukan pada bab sebelumnya. Implementasi berupa kode sumber untuk membangun program. Sebelum masuk ke penjelasan implementasi, akan ditunjukkan terlebih dahulu lingkungan untuk melakukan implementasi.

4.1 Lingkungan Implementasi

Lingkungan implementasi perangkat lunak yang akan dibangun menggunakan spesifikasi perangkat keras dan perangkat lunak seperti yang ditunjukkan pada **Tabel 4.1** berikut.

Tabel 4.1 Lingkungan implementasi perangkat lunak

Perangkat	Jenis Perangkat	Spesifikasi
Perangkat Keras	Prosesor	Intel(R) Core(TM) i7-4720HQ CPU @ 2.60GHz (8 CPUs), 2.6GHz
	Memori	8 GB
Perangkat Lunak	Sistem Operasi	Windows 10 64 bit
	Perangkat Pengembang	Pycharm

4.2 Implementasi

Pada sub bab implementasi akan menjelaskan bagaimana pembangunan perangkat lunak secara detail dan menampilkan kode sumber yang digunakan mulai dari tahap *segmentation/preprocessing*, ekstraksi fitur hingga klasifikasi.

4.2.1 Implementasi *Segmentation* pada Data EEG

Segmentation atau segmentasi adalah proses membagi data sinyal EEG menjadi beberapa *window* dengan panjang yang sama. Data masukan untuk proses segmentasi ini adalah data rekaman

EEG kelima set Z, O, N, F, dan S yang masing-masing set berjumlah 100 data. Sehingga total data adalah 500 data rekaman EEG. Setiap data rekaman EEG memiliki panjang 4097. Sehingga dapat disimpulkan bahwa ukuran data masukan awal adalah 500 x 4097.

Keluaran dari proses segmentasi pada setiap data rekaman EEG adalah beberapa *window* dengan panjang data tiap *window* adalah 64 nilai. Pada segmentasi secara *non-overlapping*, data akan bergeser sebanyak 64 nilai seperti pada variabel *shiftwindow* sehingga tidak ada data *window* yang tumpang tindih. Implementasi segmentasi secara *non-overlapping* pada data rekaman EEG dilakukan seperti pada **Kode sumber 4.1** baris 10-13. Data setiap *window* disimpan dalam bentuk matriks bernama *dataWindow* berukuran 1 x 64 pada baris ke-12. Setiap data rekaman EEG yang disegmentasi secara *non-overlapping* menghasilkan matriks berukuran *K* x *L*, yang merepresentasikan jumlah *window* dan panjang *window* secara berturut-turut. Jumlah data rekaman EEG adalah 500, maka hasil akhir dari segmentasi secara *non-overlapping* akan berupa matriks berdimensi 3 yaitu 500 x *K* x *L*, dimana *K* = 64 dan *L* = 64.

Sedangkan untuk segmentasi secara *overlapping*, data akan bergeser sebanyak 32 nilai seperti pada variabel *shiftwindow* sehingga terjadi tumpang tindih data sebanyak 32 nilai antar *window*. Implementasi segmentasi secara *overlapping* pada data rekaman EEG dilakukan seperti pada **Kode sumber 4.2** baris 10-13. Data setiap *window* disimpan dalam bentuk matriks bernama *dataWindow* berukuran 1 x 64 pada baris ke-12. Setiap data rekaman EEG yang disegmentasi secara *overlapping* menghasilkan matriks berukuran *K* x *L*, yang merepresentasikan jumlah *window* dan panjang *window* secara berturut-turut. Jumlah data rekaman EEG adalah 500, maka hasil akhir dari segmentasi secara *overlapping* akan berupa matriks berdimensi 3 yaitu 500 x *K* x *L*, dimana *K* = 127 dan *L* = 64.

1	for filename in os.listdir(folder):
2	data = [float(i) for i in tmp]

```
3     panjang_data = len(data)
4
5     panjangWindow = 64
6     shiftWindow = 64
7     batasWindow = 64
8     fiturSNO = []
9
10    for i in xrange(0, panjang_data, shiftWindow):
11        if (i + shiftWindow) < panjang_data:
12            dataWindow = data[i:batasWindow]
13            batasWindow = batasWindow + shiftWindow
```

Kode sumber 4.1 segmentasi secara *non-overlapping*

```
1   for filename in os.listdir(folder):
2     data = [float(i) for i in tmp]
3     panjang_data = len(data)
4
5     panjangWindow = 64
6     shiftWindow = 32
7     batasWindow = 64
8     fiturSO = []
9
10    for i in xrange(0, panjang_data, shiftWindow):
11        if (i + shiftWindow) < panjang_data:
12            dataWindow = data[i:batasWindow]
13            batasWindow = batasWindow + shiftWindow
```

Kode sumber 4.2 segmentasi secara *overlapping*

4.2.2 Implementasi Dekomposisi sinyal dengan *Discrete Wavelet Transform*

Discrete Wavelet Transform (DWT) adalah proses dekomposisi sinyal sesuai *band* frekuensi. Keluaran dari proses dekomposisi setiap data sinyal adalah beberapa *window* sesuai *band* frekuensi yang sesuai. Pada tugas akhir ini, metode DWT menggunakan *library pywt* yaitu fungsi *wavedec*. Fungsi *wavedec* adalah fungsi yang digunakan untuk mendekomposisi data secara

multilevel. Pada tugas akhir ini, data rekaman EEG akan didekomposisi sebanyak 4 level untuk mendapatkan 5 *band* frekuensi.

Pada **Kode sumber 4.3** baris 3-4, data didekomposisi menjadi 5 *window* dengan fungsi *wavedec*. Data *window* disimpan dalam variabel berbentuk matriks cA4, cD4, cD3, cD2, dan cD1 berukuran 1 x *L*, dimana *L* adalah panjang *window*. Kelima *window* memiliki panjang yang berbeda-beda, kecuali *window* cD4 dan cA4 yang memiliki panjang yang sama karena pada level yang sama. Setiap data rekaman EEG yang didekomposisi menghasilkan matriks berukuran *K* x *L*, yang merepresentasikan jumlah *window* dan panjang *window* secara berturut-turut. Jumlah data rekaman EEG adalah 500, maka hasil akhir dari segmentasi secara *overlapping* akan berupa matriks berdimensi 3 yaitu 500 x K x *L* dimana *K* = 5.

```
1  for filename in os.listdir(folder):
2      data = [float(k) for k in tmp]
3      coeffs = wavedec(data, 'db3', level=4)
4      cA4, cD4, cD3, cD2, cD1 = coeffs
5
```

Kode sumber 4.3 dekomposisi menggunakan *discrete wavelet transform*

4.2.3 Implementasi Ekstraksi Fitur dengan *Weighted Permutation Entropy*

Weighted Permutation Entropy (WPE) adalah metode untuk menghitung nilai entropi dengan mengukur kompleksitas dan ketidakteraturan *time series* dengan membandingkan nilai tetangga dan amplitudo berbeda pada setiap setiap pola permutasi. Proses ekstraksi fitur pada setiap *window* diperoleh dengan menghitung nilai entropi. Keluaran dari proses ekstraksi fitur satu *window* adalah satu nilai entropi yang merepresentasikan satu fitur.

Data masukan pada tahap ekstraksi fitur adalah data *window* hasil pengolahan awal. Ada 3 skenario pengolahan awal yang akan dibandingkan yaitu segmentasi secara *non-overlapping*,

segmentasi secara *overlapping*, dan dekomposisi dengan *Discrete Wavelet Transform*. Ketiga hasil pengolahan awal ini akan diekstraksi secara terpisah dan akan menghasilkan 3 data fitur yang berbeda-beda.

Pada pengolahan awal menggunakan segmentasi *non-overlapping* diperoleh data keluaran berupa matriks berukuran 500 x 64 x 64 yang akan digunakan sebagai data masukan ke proses ekstraksi fitur. Sedangkan pengolahan awal menggunakan segmentasi *overlapping* diperoleh data keluaran berukuran 500 x 127 x 64 yang akan dijadikan sebagai data masukan ke proses ekstraksi fitur. Data masukan untuk proses ekstraksi fitur adalah setiap satu *window* 1 x 64 yang disimpan pada variabel *DataWindow*. Setiap data memiliki *K window*, untuk segmentasi secara *non-overlapping K* = 64 sedangkan segmentasi secara *overlapping K* = 127. Sehingga dilakukan iterasi sebanyak 500 x *K*. Implementasi untuk melakukan parsing setiap data *window*, parameter panjang sekuens (*m*), dan *time delay* ke fungsi ekstraksi fitur *weightedPermutationEntropy* dilakukan seperti pada **Kode sumber 4.4** baris 5.

```
1  for i in range(500):
2      for j in range(K):
3          m =3
4          time_lag = 1
5          wpeWindow= eFitur.weightedPermutationEntropy(dataWindow, m, time_lag)
```

Kode sumber 4.4 data window diparsing untuk proses ekstraksi fitur

Pada pengolahan awal menggunakan dekomposisi dengan *Discrete Wavelet Transform* diperoleh data keluaran berupa matriks berukuran 500 x 5 x *L* yang akan digunakan sebagai data masukan ke proses ekstraksi fitur. Data masukan untuk proses ekstraksi fitur adalah setiap satu window yang disimpan pada variabel cA4, cD4, cD3, cD2, dan cD1. setiap satu *window* 1 x *L*

yang disimpan pada variabel *DataWindow*. Setiap data memiliki *K* = 5 *window*. Implementasi untuk melakukan parsing setiap data *window*, parameter panjang sekuens (*m*), dan *time delay* ke fungsi *weightedPermutationEntropy* dilakukan seperti pada **Kode sumber 4.5** baris 4-8.

```
1  for i in range(500):
2      m = 3
3      time_lag = 1
4      WPEcA4 = eFitur.weightedPermutationEntropy(cA4, m, time_lag)
5      WPEcD1 = eFitur.weightedPermutationEntropy(cD1, m, time_lag)
6      WPEcD2 = eFitur.weightedPermutationEntropy(cD2, m, time_lag)
7      WPEcD3 = eFitur.weightedPermutationEntropy(cD3, m, time_lag)
8      WPEcD4 = eFitur.weightedPermutationEntropy(cD4, m, time_lag)
```

Kode sumber 4.5 data window dari proses dekomposisi diparsing untuk proses ekstraksi fitur

Fungsi *weightedPermutationEntropy* memiliki nilai data window, parameter panjang sekuens (*m*) dan *time delay* (*time_lag*). Parameter *m* akan menentukan panjang sekuens yang akan dibentuk, pola permutasi dan jumlah pola permutasi yang dihasilkan. Proses pertama pada tahap ekstraksi fitur adalah membentuk sekuens dan pola permutasi yang diimplementasikan pada **Kode sumber 4.6**. Pada baris 2, Pola permutasi berupa susunan angka dibentuk seperti dijelaskan di bab sebelumnya. Selanjutnya pada baris 10-13, data masukan yaitu data *window* akan dibentuk menjadi sejumlah sekuens dengan panjang *m* nilai. Hasil keluaran dari proses ini adalah sekuens dan pola permutasi.

```
1  def weightedPermutationEntropy(self, time_series, m, time_lag):
```

```
2   array_pi_m = np.array(list(itertools.permutations(range(m))))
3   jumlah_pi_m = len(array_pi_m)
4
5   panjang_time_series = len(time_series)
6   N = panjang_time_series - (m-1) * time_lag
7
8   SequenceXj = [[]for j in range(N)]
9   NeighborSequence = [[]for j in range(N)]
10  for j in range(N):
11      for i in range(1, m+1):
12          xSequence = j+(i-1)*time_lag
13          SequenceXj[j].append(time_series[xSequence])
14
15          xNeighbor = j+(i+1)*time_lag
16          if xNeighbor<panjang_time_series:
17              NeighborSequence[j].append(time_series[xNeighbor])
18  MeanSequenceXj = [sum(i) / len(i) for i in NeighborSequence]
```

Kode sumber 4.6 membentuk pola permutasi dan sekuens

Konsep WPE adalah membandingkan setiap nilai sekuens dengan energi/varians sekuens tetangga lalu menggabungkan nilai amplitudo tersebut sebagai probabilitas pola permutasi. Pada proses pertama diperoleh sekuens dan pola permutasi, selanjutnya pada proses kedua dilakukan penghitungan probabilitas setiap pola permutasi. Pada **Kode sumber 4.6**, sekuens dibentuk dan pada baris 15-18 diperoleh varians sekuens tetangga. Implementasi perhitungan bobot sekuens dilakukan pada **Kode sumber 4.7**. Iterasi dilakukan sebanyak N, jumlah sekuens, untuk menghitung bobot masing-masing sekuens.

```
1   for j in range(N):
2       weightj = 0
3       for i in range(1, m+1):
```

```
4           xWeight = j+(i-1)*time_lag  # paper: xWeight = j+(i+1)*time_lag  # penemu: xWeight = j+(i-1)*time_lag
5           if xWeight<panjang_time_series:
6               IndexWeightSequence[j].append(xWeight)
7           NilaiIndexWeightSequence[j].append(time_series[xWeight])
8               w = time_series[xWeight] - MeanSequenceXj[j]
9               PenguranganWeight[j].append(w)
10          KuadratPenguranganWeight[j].append(np.power(w, 2))
11              weightj = weightj + np.power(w, 2)
12          weightj = weightj/3
13          WeightOfSequence.append(weightj)
```

Kode sumber 4.7 Menghitung bobot sekuens

Jumlah pola permutasi dan pola permutasi ditentukan oleh parameter panjang sekuens (*m*). Untuk setiap pola permutasi akan dilakukan penghitungan nilai probabilitas. Implementasi Perhitungan probabilitas pola permutasi dilakukan pada pada **Kode sumber 4.8**. Pemetaan sekuens terhadap pola permutasi diperoleh dengan melakukan *ascending sorting* terhadap nilai sekuens dan menyimpan indeks *sorting* nilai sekuens seperti pada baris 3. Selanjutnya bobot sekuens pada pola permutasi yang sama akan dijumlahkan. Kemudian pada baris 4-12, probabilitas setiap tipe permutasi dihitung. Hasil keluaran dari proses kedua ini adalah probabilitas setiap pola permutasi.

```
1       probability = [0] * jumlah_pi_m
2       for j in range(N):
3           sorted_index_array= np.array(np.argsort(time_series[j:j+m*time_lag:time_lag], kind='quicksort'))
4           for i in range(jumlah_pi_m):
5               if abs(array_pi_m[i] - sorted_index_array).any() == 0:
6                   probability[i] = probability[i] + WeightOfSequence[j]
7
```

```
8       sumP = 0
9       for j in range(jumlah_pi_m):
10          sumProbability = np.sum(WeightOfSequence)
11          probability[j] = probability[j]/sumProbability
12          sumP = sumP + probability[j]
```

Kode sumber 4.8 menghitung probabilitas pola permutasi

Proses terakhir pada tahap ekstraksi fitur adalah menghitung nilai entropi data *window*. Implementasi perhitungan nilai entropi dilakukan pada **Kode sumber 4.9**. Keluaran akhir dari proses ekstraksi fitur ini adalah nilai entropi *window* yang merepresentasikan satu nilai fitur. Selanjutnya, nilai fitur ini akan menjadi data masukan pada proses klasifikasi menggunakan SVM, sehingga sangat baik jika dilakukan normalisasi nilai pada nilai fitur untuk menormalkan data ekstraksi fitur pada rentang yang sama seperti pada baris 4.

```
1       SoWPE = 0
2       for j in range(jumlah_pi_m):
3           SoWPE = SoWPE + probability[j] * math.log(probability[j] if probability[j]>0 else 1,2)
4       SoWPE = - SoWPE/math.log(jumlah_pi_m, 2)
5
6     return SoWPE
```

Kode sumber 4.9 menghitung nilai entropi *window*

Nilai ekstraksi fitur satu data rekaman EEG pada data hasil segmentasi akan disimpan pada matriks bernama *fiturSNO* **Kode sumber 4.10**. Sedangkan nilai ekstraksi fitur satu data rekaman EEG pada data hasil dekomposisi DWT akan disimpan pada variabel *fiturWavelet* pada **Kode sumber 4.11**. *fiturSNO* dan *fiturWavelet* berukuran 1 x M, dimana M merupakan jumlah fitur (K) ditambah satu untuk menyimpan kelas data. Untuk data segmentasi secara *non-overlapping* M = 65, untuk segmentasi secara overlapping M = 128, dan untuk dekomposisi M = 6.

Implementasi untuk menyimpan data fitur dilakukan pada **Kode sumber 4.10**. Setelah proses ekstraksi fitur pada satu data

EEG selesai, label disematkan pada kolom terakhir matriks *fiturSNO* begitu juga pada matriks *fiturWavelet*. Proses pemberian label dan *filtering* pada data fitur pada data hasil segmentasi dan dekomposisi melalui proses yang sama seperti pada baris 2-5. Set S akan dilabeli kelas serangan epilepsi (1) dan kelas lainnya dilabeli kelas bukan serangan (-1).

Proses segmentasi/dekomposisi dan tahap ekstraksi fitur untuk semua data dilakukan secara bersamaan meskipun proses klasifikasi pada setiap set data dilakukan secara terpisah. Oleh karena itu, dilakukan proses *filtering* data. *Filtering* data yang dimaksud adalah menggabungkan data fitur dari set yang sama dalam satu variabel. Setiap data fitur akan digabungkan ke set masing-masing seperti pada baris 7-16. Misal data rekaman EEG dari set S akan dimasukkan ke variabel datasetS. Setiap set akan memiliki matriks berukuran $100 \text{ x } M$. Untuk keperluan proses klasifikasi masing-masing set Z, O, N, F akan diklasifikasikan terhadap set S, yaitu klasifikasi Z dan S, O dan S, N dan S, serta F dan S. Pada baris 18-21, set akan digabungkan sehingga menghasilkan 4 set baru yang masing-masing berukuran $200 \text{ x } M$. Kemudian masing-masing set akan disimpan dalam file berekstensi .csv.

Data rekaman EEG yang disegmentasi secara *non-overlapping* kemudian diekstrkasi akan menghasilkan 4 set data yang masing-masing set berukuran 200 x 65. Data rekaman EEG yang disegmentasi secara *overlapping* kemudian diekstraksi akan menghasilkan 4 set data yang masing-masing set berukuran 200 x 128. Data rekaman EEG yang didekomposisi dengan DWT kemudian diekstraksi akan menghasilkan 4 set data yang masing-masing set berukuran 200 x 6.

1	fiturSNO.append(wpeWindow)
2	if filename[0] == 'S':
3	fiturSNO.append(1)
4	else:
5	fiturSNO.append(-1)
6	

```
7     if filename[0] == 'S':
8         datasetS.append(fiturSNO)
9     elif filename[0] == 'F':
10        datasetF.append(fiturSNO)
11    elif filename[0] == 'N':
12        datasetN.append(fiturSNO)
13    elif filename[0] == 'O':
14        datasetO.append(fiturSNO)
15    elif filename[0] == 'Z':
16        datasetZ.append(fiturSNO)
17
18  datasetF = datasetF + datasetS
19  datasetN = datasetN + datasetS
20  datasetO = datasetO + datasetS
21  datasetZ = datasetZ + datasetS
```

Kode sumber 4.10 Proses menyimpan data fitur

```
10    fiturWavelet.append(WPEcA4)
11    fiturWavelet.append(WPEcD1)
12    fiturWavelet.append(WPEcD2)
13    fiturWavelet.append(WPEcD3)
14    fiturWavelet.append(WPEcD4)
```

Kode sumber 4.11 Proses menyimpan data fitur satu rekaman EEG pada data hasil dekomposisi DWT

4.2.4 Implementasi Klasifikasi dengan *Support Vector Machine*

Setelah diperoleh data fitur dari data rekaman EEG pada tahap ekstraksi fitur, *Support Vector Machine* digunakan untuk proses klasifikasi. Pada tugas akhir ini, metode klasifikasi SVM mengunakan *library sklearn*. Data masukan yang digunakan adalah data ekstraksi fitur yang telah disimpan pada file berekstensi *.csv*. Setiap file data ekstraksi fitur berisi 200 data, dimana 100 data merupakan kelas bukan serangan epilepsi dan 100 data lagi merupakan kelas serangan epilepsi. Setiap data menyimpan nilai fitur dan label kelas.

Implementasi klasifikasi dengan SVM dilakukan seperti pada **Kode sumber 4.12**. Klasifikasi SVM menggunakan *library sklearn* akan memparsing data fitur dan label kelas dengan parameter yang berbeda. Oleh karena itu data masukan akan disimpan ke dalam dua matriks, yaitu matriks yang menyimpan data fitur dan matriks yang menyimpan label kelas. Pada baris 2-3 data fitur disimpan dalam matriks *X* dan label kelas disimpan dalam matriks *y*. Proses pengklasifikasian data dengan *linear* SVM akan dilakukan dengan memanggil fungsi *svm.SVC()* sedangkan untuk *non-linear* SVM dilakukan dengan memanggil fungsi *svm.NuSVC()*. Pembentukan model klasifikasi dengan data *training* dilakukan seperti pada baris 14-15. Pengujian model klasifikasi dengan data *testing* dilakukan pada baris 17.

Untuk *linear* SVM ada beberapa parameter yang digunakan untuk mengoptimalkan pemodelan klasifikasi yaitu parameter optimasi (*C*) dan fungsi kernel serta parameter untuk fungsi kernel seperti *gamma* untuk kernel *polinomial*, RBF, dan *sigmoid*. *Non-linear* SVM menggunakan parameter yang sama, namun pada *library sklearn* parameter optimasi variabel *C* diganti dengan variabel *nu*. Secara matematis *nu* dan *C* adalah sama. Parameter *nu* mengontrol jumlah *support vector* dan *training error*. Parameter *nu* adalah batas atas pada bagian *training error* dan batas bawah bagian *support vector*. Nilai parameter optimasi variabel *C* pada *linear* SVM yang dapat dicobakan harus lebih besar dari 0 sedangkan parameter optimasi variabel *nu* pada *non*-linear SVM yang dapat dicobakan berada pada rentang nilai lebih besar dari 0 dan lebih kecil sama dengan 1 ($0 < nu \leq 1$). Jenis klasifikasi SVM, jenis fungsi kernel, nilai parameter optimasi (*C* atau *nu*), dan *gamma* akan dijadikan sebagai skenario uji coba.

```
1  for filename in os.listdir(folder):
2      X = np.array(df.drop(["['col0'", " 'class']"], 1))
3      y = np.array(df[" 'class']"])
4
5      n_splits = 10
```

```
6   sss = StratifiedShuffleSplit(n_splits=n_splits, test_size=0.1, random_state=0)
7   sss.get_n_splits(X, y)
8
9   meanaccuracy = 0
10  for train_index, test_index in sss.split(X, y):
11      X_train, X_test = X[train_index], X[test_index]
12      y_train, y_test = y[train_index], y[test_index]
13
14      clf = svm.SVC(kernel='rbf', C=10, gamma=1)
15      clf.fit(X_train, y_train)
16
17      accuracy = clf.score(X_test, y_test)
18      meanaccuracy = meanaccuracy + accuracy
19
20  meanaccuracy = meanaccuracy / n_splits
21  print meanaccuracy
```

Kode sumber 4.12 proses klasifikasi data menggunakan SVM

4.2.5 Implementasi *K-Fold Cross-Validation*

Metode *k-fold cross-calidation* dilakukan untuk menghindari bias saat melakukan uji performa sistem. *K-fold cross-validation* digabungkan dengan pengambilan data *training* dan *testing* secara acak. *K-fold cross-validation* yang umum digunakan pada percobaan ini adalah k=10. Sehingga data akan dibagi menjadi 10 *fold* dan setiap *fold* akan berisi 20 data. Jumlah data pada set adalah 200, maka jumlah data yang akan dijadikan sebagai data *training* sebanyak 180 dan 20 sisanya dijadikan sebagai data *testing*.

Pada **Kode sumber 4.12** baris 6-7 dilakukan pengambilan indeks data secara acak untuk setiap *fold*. Kemudian dilakukan iterasi sebanyak k. Pada baris 11-12 data dibagi menjadi data *training* dan data *testing* sesuai indeks yang telah diacak. Proses pengujian sistem dilakukan pada baris 17. Disetiap iterasi akan diperoleh akurasi model klasifikasi. Akurasi akhir dari klasifikasi satu set data adalah rata-rata akurasi setiap iterasi.

[*Halaman ini sengaja dikosongkan*]

BAB V
UJI COBA DAN EVALUASI

Dalam bab ini dibahas mengenai skenario uji coba pada perangkat lunak yang telah dibangun. Setelah itu, hasil uji coba akan dievaluasi kinerjanya sehingga dapat diputuskan apakah perangkat lunak ini mampu menyelesaikan permasalahan yang telah dirumuskan diawal. Secara garis besar, bab ini berisi pembahasan mengenali lingkungan pengujian, data pengujian, dan uji kinerja.

5.1 Lingkungan Uji Coba

Lingkungan pengujian pada uji coba permasalahan klasifikasi serangan epilepsi menggunakan *segmentation/preprocessing, Discrete Wavelet Transform, Weighted Permutation Entropy,* dan *Support Vector Machine* menggunakan spesifikasi perangkat keras dan perangkat lunak seperti yang ditunjukkan pada **Tabel 5.1**.

Tabel 5.1 Spesifikasi lingkungan uji coba

Perangkat	Jenis Perangkat	Spesifikasi
Perangkat Keras	Prosesor	Intel(R) Core(TM) i7-4720HQ CPU @ 2.60GHz (8 CPUs), 2.6GHz
	Memori	8 GB
Perangkat Lunak	Sistem Operasi	Windows 10 64 bit
	Perangkat Pengembang	Pycharm

5.2 Data Uji Coba

Data yang digunakan pada tugas akhir ini adalah data rekaman sinyal otak manusia (EEG) yang diunduh dari website milik "Klinik für Epileptologie, Universität Bonn". Pada Tugas Akhir ini, data yang digunakan adalah kelima set Z, O, N, F, dan

S. Setiap set terdiri dari 100 data, sehingga total data ada 500. Dari kelima dataset ini, set S adalah satu-satunya set untuk kelas serangan epilepsi sedangkan empat set lainnya adalah kelas bukan serangan epilepsi. Maka set S akan diklasifikasikan terhadap ke-empat set klasifikasi lainnya, yaitu set Z dengan S, set O dengan S, set N dengan S, dan set F dengan S.

Data awalan ini, akan masuk ke proses pengolahan data yaitu *segmentation* atau segmentasi dan dekomposisi dengan *Discrete Wavelet Transform*. Kedua perlakuan berbeda pada tahap pengolahan data akan menjadi skenario uji coba. Selanjutnya data hasil segmentasi dan dekomposisi akan diekstraksi untuk mendapatkan nilai fitur data. Kemudian data ekstraksi fitur ini akan di proses ke tahap klasifikasi.

5.3 Skenario Uji Coba

Sebelum melakukan uji coba, perlu ditentukan skenario yang akan digunakan dalam uji coba. Melalui skenario uji coba ini, perangkat akan diuji apakah sudah berjalan dengan benar, bagaimana performa pada masing-masing skenario dan perbandingan performa antara skenario mana yang memiliki hasil paling baik.

Pada Tugas akhir ini, terdapat berbagai macam skenario uji coba, yaitu:

1. Perhitungan performa berdasarkan parameter panjang sekuens (*m*) pada tahap ekstraksi fitur.
2. Perhitungan performa berdasarkan parameter *time delay* pada tahap ekstraksi fitur.
3. Perhitungan performa berdasarkan parameter optimasi pada tahap klasifikasi.
4. Perhitungan performa berdasarkan jenis fungsi kernel pada tahap klasifikasi.
5. Perhitungan performa berdasarkan jumlah *k-fold cross validation.*
6. Perhitungan performa berdasarkan pra-proses.

5.3.1 Skenario Uji Coba Perhitungan Performa Berdasarkan Parameter Panjang Sekuens (*m*) pada Tahap Ekstraksi Fitur

Skenario uji coba pertama adalah perhitungan performa parameter panjang sekuens (*m)* pada tahap ekstraksi fitur. Parameter *m* yang dapat dicobakan pada umumnya adalah 3-8 dan untuk nilai *m* yang dapat dicobakan untuk setiap *time series* tergantung pada panjang *time series* [12]. Nilai *m* yang dapat dicobakan harus memenuhi persyaratan bahwa nilai permutasi *m* (*m*!) lebih kecil dari panjang *time series*. Dalam pembangunan perangkat lunak ini, data *window* dari proses pengolahan data awal digunakan sebagai *time series* untuk ekstraksi fitur. Pada tahap pengolahan awal yaitu segmentasi dan dekomposisi, data dibagi menjadi beberapa *window*. Panjang *window* hasil segmentasi adalah 64 sehingga panjang sekuens *m* yang dapat dicobakan untuk *window* ini adalah 3 dan 4. Sehingga untuk skenario uji coba ini, panjang sekuens *m* yang akan dujicobakan pada semua *window* adalah 3 dan 4.

Perhitungan performa parameter panjang sekuens *m* akan diujicobakan pada data dengan pengolahan awal segmentasi secara *overlapping*. Jenis klasifikasi yang digunakan adalah *linear* SVM. Nilai parameter *delay time* yang digunakan adalah 1. Hal ini didasari pada penelitian sebelumnya pada pemrosesan EEG bahwa nilai *time delay* di-set dengan nilai 1 [5]. Nilai optimasi pada *linear* SVM *C* = 1. Nilai *k-fold cross-validation* yang digunakan adalah 10. Hal ini dikarenakan nilai k=10 sering digunakan [20]. Fungsi kernel yang akan digunakan adalah kernel RBF dengan parameter *gamma* di-set *auto*, yaitu $\frac{1}{jumlah\ fitur}$ akan digunakan sebagai koefsien kernel.

Hasil performa parameter panjang sekuens *m* dapat dilihat pada **Tabel 5.2**. Kesimpulan yang diperoleh dari uji coba perhitungan performa parameter panjang sekuens *m* bahwa akurasi rata-rata terbaik diperoleh ketika nilai *m* adalah 4 yaitu sebesar 79,13%.

Tabel 5.2 Perhitungan performa parameter panjang sekuens (*m*) pada proses ekstraksi fitur

Dataset	Panjang Sekuens (m)	
	3 (%)	4 (%)
Z versus S	94,50	96,00
O versus S	78,50	78,50
N versus S	76,00	76,00
F versus S	64,00	66,00
Rata-rata	78,25	79,13

5.3.2 Skenario Uji Coba Perhitungan Performa Berdasarkan Parameter *Time Delay* pada Tahap Ekstraksi Fitur

Skenario uji coba yang kedua adalah perhitungan performa parameter *time delay* pada proses ekstraksi fitur. Parameter *time delay* ini mempengaruhi nilai-nilai yang akan diambil dari *time series* ke setiap sekuens. Parameter *time delay* yang akan diujicobakan untuk setiap *time series* adalah 1, 2, dan 3.

Perhitungan performa parameter *time delay* akan diujicobakan pada data dengan pengolahan awal segmentasi secara *overlapping*. Pada uji coba ini, parameter yang digunakan adalah parameter yang pada uji coba sebelumnya menghasilkan akurasi rata-rata terbaik. Berdasarkan uji coba, nilai parameter yang menghasilkan akurasi terbaik adalah panjang sekuens $m = 4$. Untuk parameter yang belum diujicoba nilai parameter yang akan digunakan yaitu jenis klasifikasi linear SVM dengan nilai optimasi pada *linear* SVM $C = 1$, nilai *k-fold cross-validation* $k = 10$ dan fungsi kernel RBF dengan parameter *gamma* di-set *auto*.

Hasil performa parameter *time delay* dapat dilihat pada **Tabel 5.3**. Kesimpulan yang diperoleh dari uji coba perhitungan performa parameter *time delay* bahwa akurasi rata-rata terbaik diperoleh ketika nilai *time delay* adalah 1 yaitu sebesar 79,13%.

Tabel 5.3 Perhitungan performa parameter *time delay* pada proses ekstraksi fitur

Dataset	time delay		
	1 (%)	2 (%)	3 (%)
Z versus S	96,00	91,50	91,50
O versus S	78,50	84,00	87,00
N versus S	76,00	59,50	60,50
F versus S	66,00	62,00	69,00
Rata-rata	79,13	74,25	77,00

5.3.3 Skenario Uji Coba Perhitungan Performa Berdasarkan Parameter Optimasi pada Tahap Klasifikasi

Skenario uji coba ketiga adalah perhitungan performa parameter optimasi. Parameter optimasi mengontrol misklasifikasi *trade-off* pada sampel *training* terhadap kesederhanaan permukaan *hyperplane*.

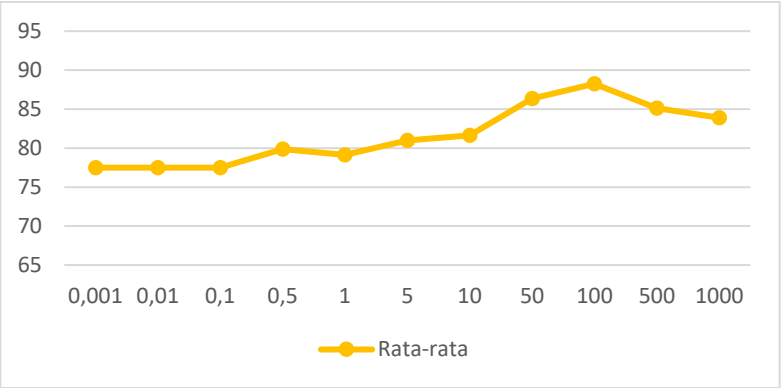
Pada tugas akhir ini, implementasi tahap klasifikasi dengan SVM menggunakan *library sklearn*, dimana variabel parameter optimasi untuk *linear* SVM dan *non-linear* SVM berbeda. variabel *C* digunakan sebagai parameter optimasi pada *linear* SVM dan variabel *nu* digunakan sebagai parameter optimasi pada *non-linear* SVM. Parameter *C* dan parameter *nu* secara matematis adalah sama. Nilai yang akan diujicoba yaitu $C > 0$ dan $0 < nu \leq 1$.

Perhitungan performa parameter optimasi *C* akan diujicobakan pada data dengan pengolahan awal segmentasi secara *overlapping*. Pada uji coba ini, parameter yang digunakan adalah parameter yang pada uji coba sebelumnya menghasilkan akurasi rata-rata terbaik. Berdasarkan uji coba, nilai parameter yang menghasilkan akurasi terbaik adalah panjang sekuens $m = 4$, *time delay* = 1. Untuk parameter yang belum diujicoba nilai parameter yang akan digunakan yaitu nilai *k-fold cross-validation* $k = 10$ dan fungsi kernel RBF dengan parameter *gamma* di-set *auto*. Hasil performa parameter *C* pada *linear* SVM dapat dilihat pada **Tabel**

5.4. Grafik performa parameter optimasi *C* ditunjukkan pada **Gambar 5.1**.

Tabel 5.4 Perhitungan performa parameter *C* pada *linear* SVM

Optimasi (*C*)	Dataset				Rata-rata
	Z vs S	O vs S	N vs S	F vs S	
0,001	93,00	75,00	75,50	66,50	77,50
0,01	93,00	75,00	75,50	66,50	77,50
0,1	93,00	75,00	75,50	66,50	77,50
0,5	96,00	77,50	79,50	66,50	79,88
1	96,00	78,50	76,00	66,00	79,12
5	96,00	80,00	80,00	68,00	81,00
10	96,00	79,50	80,00	71,00	81,63
50	96,50	78,00	84,00	87,00	86,38
100	97,50	83,00	84,50	88,00	88,25
500	93,50	82,50	86,00	78,50	85,13
1000	92,00	81,50	86,50	73,50	83,88

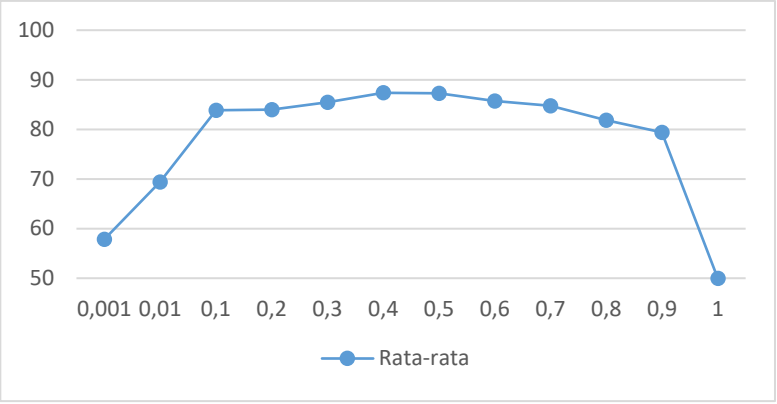


Gambar 5.1 Grafik performa parameter *C* pada *linear* SVM

Perhitungan performa parameter optimasi *nu* akan diujicoba pada data dengan pengolahan awal segmentasi secara *overlapping*. Jenis klasifikasi yang digunakan yaitu *non-linear* SVM. Pada uji coba ini, parameter yang digunakan adalah parameter yang pada uji coba sebelumnya menghasilkan akurasi rata-rata terbaik. Berdasarkan uji coba, nilai parameter yang menghasilkan akurasi terbaik adalah panjang sekuens m = 4, *time delay* = 1. Untuk parameter yang belum diujicoba nilai parameter yang akan digunakan yaitu nilai *k-fold cross-validation* k = 10 dan fungsi kernel RBF dengan parameter *gamma* di-set *auto*. Hasil performa parameter *nu* pada *non-linear* SVM dapat dilihat pada **Tabel 5.5**. Grafik performa parameter optimasi *nu* ditunjukkan pada **Gambar 5.2**.

Tabel 5.5 Perhitungan performa parameter *nu* pada *non-linear* SVM

Optimasi (*nu*)	Dataset				Rata-rata
	Z vs S	O vs S	N vs S	F vs S	
0,001	67,00	46,00	61,00	57,50	57,88
0,01	81,50	50,00	78,00	68,00	69,38
0,1	95,50	72,50	88,00	79,50	83,88
0,2	96,00	78,00	86,00	76,00	84,00
0,3	96,00	82,00	84,00	80,00	85,50
0,4	95,00	83,00	84,00	86,50	87,38
0,5	95,00	82,50	83,50	88,00	87,25
0,6	95,50	80,00	81,00	86,00	85,75
0,7	96,00	80,00	79,50	84,00	84,75
0,8	96,00	79,00	79,50	73,00	81,88
0,9	96,00	78,00	77,00	66,50	79,38
1	50,00	50,00	50,00	50,00	50,00



Gambar 5.2 Grafik performa parameter *nu* pada *non-linear* SVM

Kesimpulan yang diperoleh dari uji coba parameter optimasi adalah akurasi rata-rata terbaik pada *linear* SVM diperoleh ketika nilai *C* = 100 dengan akurasi terbaik sebesar 88,25% sedangkan pada *non-linear* SVM diperoleh ketika nilai *nu* = 0,4 dengan akurasi terbaik sebesar 87,38%. Klasifikasi dengan linear SVM memberikan rata-rata akurasi terbaik dibandingkan dengan non-linear SVM.

5.3.4 Skenario Uji Coba Perhitungan Performa Berdasarkan Jenis Fungsi Kernel

Skenario uji coba yang keempat adalah perhitungan performa fungsi kernel yang berbeda-beda yaitu kernel *linear*, kernel *polinomial*, kernel *Radial Basis Function* (kernel RBF), dan kernel *sigmoid*. Selain itu, parameter *gamma* juga akan diujicobakan. Parameter *gamma* adalah koefisien kernel untuk kernel RBF, *polinomial*, dan *sigmoid*.

Perhitungan performa fungsi kernel dan *gamma* akan diujicobakan pada data dengan pengolahan awal segmentasi secara *overlapping*. Pada uji coba ini, parameter yang digunakan adalah

parameter yang pada uji coba sebelumnya menghasilkan akurasi rata-rata terbaik. Berdasarkan uji coba, nilai parameter yang menghasilkan akurasi terbaik adalah panjang sekuens *m* = 4, *time delay* =1, jenis klasifikasi linear SVM dengan nilai parameter optimasi *C* = 100. Untuk parameter yang belum diujicoba nilai parameter yang akan digunakan yaitu nilai *k-Fold Cross-Validation* k = 10.

Hasil performa fungsi kernel pada *linear* SVM dapat dilihat pada **Tabel 5.6**. Hasil performa parameter gamma pada kernel RBF dapat dilihat pada **Tabel 5.7**. Hasil performa parameter gamma pada kernel polinomial dan sigmoid dapat dilihat pada sub bab lampiran. Grafik performa parameter *gamma* pada RBF ditunjukkan pada **Gambar 5.3.** Berdasarkan hasil uji coba yang ditunjukkan pada **Tabel 5.6** dan **Tabel 5.7** dapat disimpulkan bahwa fungsi kernel yang menghasilkan akurasi rata-rata terbaik adalah kernel RBF dengan nilai gamma = 2 yaitu sebesar 91,88%.

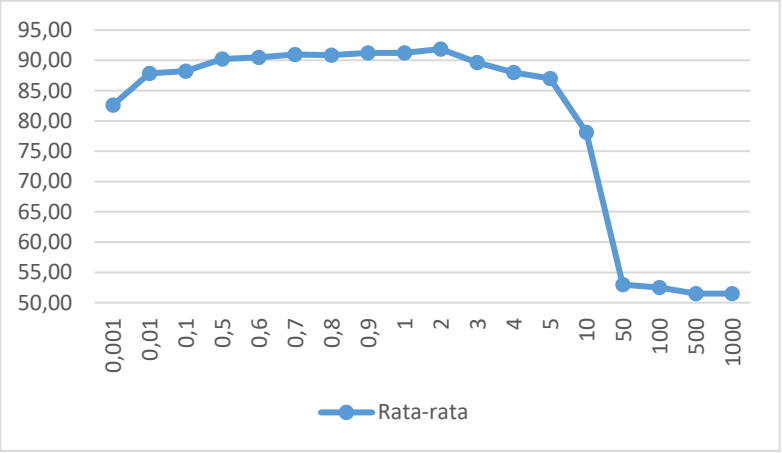
Tabel 5.6 Perhitungan performa fungsi kernel pada *linear* SVM

Fungsi Kernel	Dataset				Rata-rata
	Z vs S	O vs S	N vs S	F vs S	
Linear	92,00	78,00	79,50	72,50	80,50
Poly-nomial	97,50	78,00	82,00	81,50	84,75
RBF	97,50	82,50	94,00	93,50	91,88
Sigmoid	96,50	80,50	82,00	84,50	85,88

Tabel 5.7 Perhitungan performa parameter *gamma* pada fungsi kernel RBF

Gamma	Dataset				Rata-rata
	Z vs S	O vs S	N vs S	F vs S	
0,001	96,00	80,50	80,00	74,00	82,63
0,01	97,50	82,00	83,50	88,50	87,88
0,1	94,00	80,00	88,00	91,00	88,25
0,5	96,00	77,00	94,00	94,00	90,25

0,6	96,00	77,00	95,00	94,00	90,50
0,7	96,50	78,50	95,00	94,00	91,00
0,8	96,50	78,50	94,50	94,00	90,88
0,9	96,50	80,00	94,50	94,00	91,25
1	96,50	80,00	94,50	94,00	91,25
2	97,50	82,50	94,00	93,50	91,88
3	96,00	83,00	89,50	90,00	89,63
4	91,00	82,00	89,00	90,00	88,00
5	88,50	80,50	89,50	89,50	87,00
10	80,50	72,00	80,00	80,00	78,13
50	53,00	53,00	53,00	53,00	53,00
100	52,50	52,50	52,50	52,50	52,50
500	51,50	51,50	51,50	51,50	51,50
1000	51,50	51,50	51,50	51,50	51,50



Gambar 5.3 Grafik performa parameter *gamma* pada fungsi kernel RBF

5.3.5 Skenario Uji Coba Perhitungan Performa Berdasarkan Jumlah k-Fold Cross-Validation

Pada skenario uji coba yang kelima akan diuji performa jumlah *k* yang berbeda-beda. *K-fold cross-validation* berfungsi untuk menghindari bias pada sistem klasifikasi data. Nilai *k* yang akan diujicobakan pada tugas akhir ini adalah 5 dan 10.

Perhitungan performa nilai *k-fold cross-validation* akan diujicobakan pada data dengan pengolahan awal yang berbeda yaitu segmentasi secara overlapping. Pada uji coba ini, parameter yang digunakan adalah parameter yang pada uji coba sebelumnya menghasilkan akurasi rata-rata terbaik. Berdasarkan uji coba, nilai parameter yang menghasilkan akurasi terbaik adalah panjang sekuens *m* = 4, *time delay* = 1, jenis klasifikasi *linear* SVM nilai parameter optimasi *C* = 100, dan kernel RBF dengan *gamma* = 2.

Tabel 5.8 Perhitungan performa *k-fold cross-validation*

Dataset	***k-fold***	
	5 (%)	10 (%)
Z versus S	96,50	97,50
O versus S	79,50	82,50
N versus S	93,50	94,00
F versus S	93,00	93,50
Rata-rata	90,63	91,88

Hasil performa *k-fold cross-validation* dapat dilihat pada **Tabel 5.8**. Berdasarkan hasil uji coba yang ditunjukkan pada **Tabel 5.8** dapat disimpulkan bahwa nilai *k* = 10 memberikan rata-rata akurasi terbaik yaitu sebesar 91,88%.

5.3.6 Skenario Uji Coba Perhitungan Performa Berdasarkan Pra-proses

Pada skenario uji coba yang ke-enam akan diuji performa pra-proses yang berbeda-beda. Pra-proses yang akan diujicoba adalah segmentasi secara non-overlapping (SNO), segmentasi

secara overlapping (SO), dan dekomposisi dengan *Discrete Wavelet Transform* (DWT). Pada uji coba ini, parameter yang digunakan adalah parameter yang pada uji coba sebelumnya menghasilkan akurasi rata-rata terbaik. Berdasarkan uji coba, nilai parameter yang menghasilkan akurasi terbaik adalah panjang sekuens *m* = 4, *time delay* = 1, jenis klasifikasi *linear* SVM dengan nilai parameter optimasi *C* = 100, dan kernel RBF dengan *gamma* = 2. Hasil performa fungsi kernel pada *linear* SVM dapat dilihat pada **Tabel 5.9**.

Tabel 5.9 Perhitungan performa pra-proses

Dataset	Pra-proses		
	SNO (%)	SO (%)	DWT (%)
Z versus S	96,50	97,50	92,50
O versus S	77,50	82,50	93,00
N versus S	94,00	94,00	96,00
F versus S	91,50	93,50	84,50
Rata-rata	89,88	91,88	91,50

Berdasarkan hasil uji coba yang ditunjukkan pada **Tabel 5.9** dapat disimpulkan bahwa pra-proses terbaik adalah segmentation secara overlapping dengan rata-rata akurasi terbaik sebesar 91,88%.

5.4 Evaluasi Umum Skenario Uji Coba

Berdasarkan skenario uji coba yang telah dilakukan dengan mengubah nilai parameter, dapat diketahui bahwa sistem klasifikasi serangan epilepsi menggunakan *Weighted Permutation Entropy* dan *Support Vector Machine* menghasilkan rata rata akurasi terbaik sebesar 91,88%. Parameter-parameter yang digunakan yaitu panjang sekuens 4, parameter time delay 1, parameter optimasi C sebesar 100 pada linear SVM, parameter fungsi kernel RBF dengan gamma 2, nilai k-fold yang digunakan yaitu 10, dan pengolahan awal segmentasi secara *overlapping*. Perbandingan akurasi akhir sistem pada tiap dataset ditunjukkan

pada **Tabel 5.10**. Berdasarkan **Tabel 5.10** dapat disimpulkan bahwa akurasi klasifikasi Z dan S terbaik yaitu sebesar 97,50%, akurasi klasifikasi O dan S terbaik yaitu sebesar 82,50%, akurasi klasfikasi N dan S terbaik yaitu sebesar 94,00%, akurasi klasfikasi F dan S terbaik yaitu sebesar 93,50%.

Tabel 5.10 Performa sistem

Dataset	Akurasi (%)
Z versus S	97,50
O versus S	82,50
N versus S	94,00
F versus S	93,50
Rata-rata	91,88

[*Halaman ini sengaja dikosongkan*]

BAB VI
KESIMPULAN DAN SARAN

Bab ini membahas mengenai kesimpulan yang dapat diambil dari hasil uji coba yang telah dilakukan sebagai jawaban dari rumusan masalah. Selain itu juga terdapat saran yang ditujukan untuk pengembangan penelitian lebih lanjut.

6.1 Kesimpulan

Kesimpulan yang diperoleh dari uji coba dan evaluasi adalah sebagai berikut:

1. Klasifikasi serangan epilepsi dan bukan serangan epilepsi terbaik antara set S terhadap set Z, O, N, dan F diperoleh dengan pengolahan awal segmentasi *overlapping* dengan akurasi rata-rata sebesar 91,88%.
2. Implementasi metode *Weighted Permutation Entropy* dan *Support Vector Machine* mampu mengklasifikakan aktivitas serangan epilepsi paling baik pada set S terhadap set Z dengan akurasi sebesar 97,50%, diikuti oleh set N dengan akurasi sebesar 94,00%, set F dengan akurasi sebesar 93,50%, dan terakhir set O dengan akurasi sebesar 82,50%.
3. Pemilihan parameter panjang sekuens (m) sama dengan 4 menghasilkan rata-rata akurasi terbaik yaitu sebesar 81.4%.
4. Pemilihan parameter *time delay* sama dengan 1 menghasilkan rata-rata akurasi terbaik yaitu sebesar 81.4%.
5. Klasifikasi dengan *linear* SVM menghasilkan rata-rata akurasi terbaik dibandingkan dengan *non-linear* SVM. Rata-rata akurasi terbaik diperoleh ketika nilai optimasi C sama dengan 100 yaitu sebesar 89,88%.
6. Pemilihan fungsi kernel RBF dengan nilai *gamma* sama dengan 2 menghasilkan rata-rata akurasi terbaik yaitu sebesar 91,88%.
7. Pemilihan nilai k=10 pada *k-fold cross-validation* memberikan akurasi terbaik sebesar 91,88%.

6.2 Saran

Saran yang dapat diberikan untuk pengembangan lebih lanjut dari tugas akhir ini antara lain:

1. Menggunakan algoritma *Grid Search* atau *Random Search* dalam pemilihan nilai parameter optimasi *C* dan *gamma* terbaik.
2. Membuat GUI aplikasi agar lebih mudah digunakan.

DAFTAR PUSTAKA

[1] N. S. Tawfik, S. M. Youssef, dan M. Kholief, "A hybrid automated detection of epileptic seizures in EEG records," *Comput. Electr. Eng.*, vol. 53, hal. 177–190, Jul 2016.

[2] "Epilepsi," *Wikipedia bahasa Indonesia, ensiklopedia bebas*. 30-Okt-2016.

[3] S. B. Wilson, M. L. Scheuer, C. Plummer, B. Young, dan S. Pacia, "Seizure detection: correlation of human experts," *Clin. Neurophysiol. Off. J. Int. Fed. Clin. Neurophysiol.*, vol. 114, no. 11, hal. 2156–2164, Nov 2003.

[4] "WHO | Epilepsy," *WHO*. [Daring]. Tersedia pada: http://www.who.int/mediacentre/factsheets/fs999/en/. [Diakses: 30-Mei-2017].

[5] N. Nicolaou dan J. Georgiou, "Detection of epileptic electroencephalogram based on Permutation Entropy and Support Vector Machines," *Expert Syst. Appl.*, vol. 39, no. 1, hal. 202–209, Jan 2012.

[6] A. T. Tzallas, M. G. Tsipouras, dan D. I. Fotiadis, "The use of time-frequency distributions for epileptic seizure detection in EEG recordings," *Conf. Proc. Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. IEEE Eng. Med. Biol. Soc. Annu. Conf.*, vol. 2007, hal. 3–6, 2007.

[7] M. Teplan, "FUNDAMENTALS OF EEG MEASUREMENT," vol. 2, 2002.

[8] H. Azami, K. Mohammadi, dan B. Bozorgtabar, "An Improved Signal Segmentation Using Moving Average and Savitzky-Golay Filter," *J. Signal Inf. Process.*, vol. 03, no. 01, hal. 39, Feb 2012.

[9] J. M. Barreiro, F. Martin-Sanchez, V. Maojo, dan F. Sanz, *Biological and Medical Data Analysis: 5th International Symposium, ISBMDA 2004, Barcelona, Spain, November 18-19, 2004, Proceedings*. Springer, 2004.

[10] A. Graps, "An Introduction to Wavelets."

[11] R. Haddadi, E. Abdelmounim, M. E. Hanine, dan A. Belaguid, "Discrete Wavelet Transform based algorithm for recognition

of QRS complexes," in *2014 International Conference on Multimedia Computing and Systems (ICMCS)*, 2014, hal. 375–379.

[12] C. Bandt dan B. Pompe, "Permutation Entropy: A Natural Complexity Measure for Time Series." [Daring]. Tersedia pada: https://www.researchgate.net/publication/11364831_Permutation_Entropy_A_Natural_Complexity_Measure_for_Time_Series. [Diakses: 30-Mei-2017].

[13] "Weighted-Permutation Entropy Analysis of Resting State EEG from Diabetics with Amnestic Mild Cognitive Impairment (PDF Download Available)." [Daring]. Tersedia pada: https://www.researchgate.net/publication/307851235_Weighted-Permutation_Entropy_Analysis_of_Resting_State_EEG_from_Diabetics_with_Amnestic_Mild_Cognitive_Impairment. [Diakses: 03-Jun-2017].

[14] "Weighted-permutation entropy: A complexity measure for time series incorporating amplitude information," *ResearchGate*. [Daring]. Tersedia pada: https://www.researchgate.net/publication/236051016_Weighted-permutation_entropy_A_complexity_measure_for_time_series_incorporating_amplitude_information. [Diakses: 30-Mei-2017].

[15] P. Gaspar, J. Carbonell, dan J. L. Oliveira, "On the parameter optimization of Support Vector Machines for binary classification," *J. Integr. Bioinforma.*, vol. 9, no. 3, hal. 201, Jul 2012.

[16] C. Cortes dan V. Vapnik, "Support-Vector Machine," *Kluwer Acad. Publ. Boston*, hal. 273–297, 1995.

[17] R. Kurnia, "Support Vector Machine –Teori dan Aplikasinya dalam Bioinformatika 1."

[18] R. Amami, D. B. Ayed, dan N. Ellouze, "Practical Selection of SVM Supervised Parameters with Different Feature Representations for Vowel Recognition," *ArXiv150706020 Cs*, Jul 2015.

[19] "Confusion Matrix-based Feature Selection. (PDF Download Available)," *ResearchGate*. [Daring]. Tersedia pada: https://www.researchgate.net/publication/220833270_Confusion_Matrix-based_Feature_Selection. [Diakses: 14-Jun-2017].

[20] P. Refaeilzadeh, L. Tang, dan H. Liu, "Cross-Validation."

[21] "EEG time series downlaod page." [Daring]. Tersedia pada: http://epileptologie-bonn.de/cms/front_content.php?idcat=193&lang=3.

[22] R. G. Andrzejak, K. Lehnertz, F. Mormann, C. Rieke, P. David, dan C. E. Elger, "Indications of nonlinear deterministic and finite-dimensional structures in time series of brain electrical activity: Dependence on recording region and brain state," vol. 64, Nov 2001.

[*Halaman ini sengaja dikosongkan*]

LAMPIRAN

1. Performa parameter gamma pada fungsi kernel polinomial

Gamma	Klasifikasi Dataset				Rata-rata
	Z vs S	O vs S	N vs S	F vs S	
0,001	60,00	60,50	66,50	64,50	62,88
0,005	96,50	77,00	74,50	66,00	78,50
0,006	96,00	78,00	77,00	65,50	79,13
0,007	96,00	78,50	79,00	66,00	79,87
0,008	96,00	78,00	80,50	67,00	80,38
0,009	96,00	79,00	82,00	66,50	80,88
0,01	96,50	80,50	82,00	69,50	82,13
0,02	**97,50**	**78,00**	**82,00**	**81,50**	**84,75**
0,03	93,50	80,50	78,00	81,00	83,25
0,04	91,50	79,50	80,50	76,00	81,88
0,05	91,50	81,00	82,00	75,00	82,38
0,1	91,50	77,50	80,50	75,00	81,13
0,5	91,50	77,50	80,50	75,00	81,13
1	91,50	77,50	80,50	75,00	81,13
5	91,50	77,50	80,50	75,00	81,13
10	91,50	77,50	80,50	75,00	81,13
50	91,50	77,50	80,50	75,00	81,13
100	91,50	77,50	80,50	75,00	81,13
500	91,50	77,50	80,50	75,00	81,13
1000	91,50	77,50	80,50	75,00	81,13

2. Performa parameter gamma pada fungsi kernel sigmoid

Gamma	Klasifikasi Dataset				Rata-rata
	Z vs S	O vs S	N vs S	F vs S	
0,001	96,00	80,00	79,00	69,00	81,00
0,005	96,00	78,50	82,00	82,50	84,75
0,006	96,50	78,00	82,00	84,50	85,25
0,007	96,50	78,00	83,00	84,00	85,38
0,008	96,50	78,50	82,00	84,50	85,38
0,009	96,50	79,00	82,00	85,00	85,63
0,01	**96,50**	**80,50**	**82,00**	**84,50**	**85,88**
0,02	97,00	77,50	69,50	77,50	80,38
0,03	38,50	67,00	60,00	63,50	57,25
0,04	3,00	51,00	48,50	56,00	39,63
0,05	3,00	43,50	36,00	46,50	32,25
0,1	3,00	27,00	23,50	43,00	24,13
0,5	3,00	31,00	48,50	49,50	33,00
1	17,00	37,00	34,50	39,00	31,88
5	50,00	50,00	50,00	50,00	50,00
10	50,00	50,00	50,00	50,00	50,00
50	50,00	50,00	50,00	50,00	50,00
100	50,00	50,00	50,00	50,00	50,00
500	50,00	50,00	50,00	50,00	50,00
1000	50,00	50,00	50,00	50,00	50,00

BIODATA PENULIS

Lophita Y Napitupulu lahir pada tanggal 10 Juli 1995 di Sosordolok. Penulis menempuh pendidikan mulai dari SDN 173543 Sosordolok (2001 - 2007), SMPN 4 Balige (2007 - 2010), SMAN 2 Balige (2010 - 2013). Saat ini penulis sedang menempuh pendidikan perguruan tinggi di Institut Teknologi Sepuluh Nopember Surabaya di jurusan Teknik Informatika Fakultas Teknologi Informasi angkatan tahun 2013.

Selama kuliah di Institut Teknologi Sepuluh Nopember, penulis mengikuti kegiatan organisasi dengan berpartisipasi sebagai staff Departemen Kewirausahaan BEM Fakultas Teknologi Informasi 2015-2016. Selain itu penulis juga memiliki pengalaman kepanitiaan sebagai staff National Seminar of Technology 2014, staff Dana dan Usaha SCHEMATICS 2015, staff Doa Pemerhati dan Konsumsi Panitia Natal-Paskah Persekutuan Mahasiswa Kristen ITS 2014-2015, Bendahara Panitia Natal-Paskah Persekutuan Mahasiswa Kristen ITS 2015-2016.

Penulis memiliki bidang minat Komputasi Cerdas Visi (KCV). Komunikasi dengan penulis dapat melalui email: **lophitan@gmail.com**.