TUGAS AKHIR – KI1502

DETEKSI SINYAL P300 MENGGUNAKAN METODE BATCH NORMALISATION NEURAL NETWORK

Adenuar Purnomo
NRP 05111440000079

Dosen Pembimbing I
Prof. Ir. Handayani Tjandrasa, M.Sc., Ph.D.

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

DEPARTEMEN INFORMATIKA
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2018

TUGAS AKHIR – KI1502

DETEKSI SINYAL P300 MENGGUNAKAN METODE BATCH NORMALISATION NEURAL NETWORK

Adenuar Purnomo
NRP 05111440000079

Dosen Pembimbing I
Prof. Ir. Handayani Tjandrasa, M.Sc., Ph.D.

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

DEPARTEMEN INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI DAN KOMUNIKASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA 2018

[Halaman ini sengaja dikosongkan]

UNDERGRADUATE THESES – KI1502

P300 SIGNAL DETECTION USING BATCH NORMALISATION NEURAL NETWORK

Adenuar Purnomo
NRP 05111440000079

Supervisor I
Prof. Ir. Handayani Tjandrasa, M.Sc., Ph.D.

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

DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA 2018

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

DETEKSI SINYAL P300 MENGGUNAKAN METODE BATCH NORMALISATION NEURAL NETWORK

TUGAS AKHIR

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

Oleh
Adenuar Purnomo
NRP : 05111440000079

Disetujui oleh Dosen Pembimbing Tugas Akhir:

1. Prof. Ir. Handayani Tjandrasa, M.Sc., Ph.D. .....
   NIP. 194908231976032001 (Pembimbing 1)

2. Dini Adni Navastara, S.Kom., M.Sc. .....
   NIP. 19851017201504200 (Pembimbing 2)

SURABAYA
JUNI, 2018

[Halaman ini sengaja dikosongkan]

DETEKSI SINYAL P300 MENGGUNAKAN METODE BATCH NORMALISATION NEURAL NETWORK

Nama Mahasiswa	**: Adenuar Purnomo**
NRP	**: 05111440000079**
Departemen	**: Informatika FTIK-ITS**
Dosen Pembimbing 1	**: Prof. Ir. Handayani Tjandrasa, M.Sc., Ph.D.**
Dosen Pembimbing 2	**: Dini Adni Navastara, S.Kom., M.Sc.**

Abstrak

Brain-computer interface (BCI) adalah mekanisme komunikasi antara sinyal electroencephalograph (EEG) dan komputer dimana sehingga sistem dapat menangkap niat otak tanpa melibatkan neuron motorik dan muskular. Mendeteksi sinyal P300 dari EEG adalah kunci untuk menafsirkan maksud dari pengguna. Convolutional Neural Network (CNN) dapat mendeteksi sinyal P300 dengan baik, namun CNN cenderung overfitting.

Untuk mengatasi masalah ini diimplementasikan sebuah metode yang bernama Batch Normalisation Neural Network (BNNN) untuk mendeteksi sinyal P300. Sinyal inputan pertama-tama dipotong terlebih dahulu sepanjang 0-667ms setelah stimulus, setelah itu difilter menggunakan Butterworth Filter. Jumlah data masing-masing kelas pada data inputan tidak seimbang, sehingga dilakukan duplikasi untuk menyeimbangkannya. Kemudian dilakukan normalisasi menggunakan min-max normalization, dan dilakukan signal averaging. Data kemudian diperhalus menggunakan Haar Wavelet Transformation dan inverse Haar Wavelet Transformation. Barulah data dapat diklasifikasi menggunakan Batch Normalisation Neural Network. Performa metode klasifikasi yang diusulkan diuji dengan membandingkan kelas

sinyal hasil prediksi dengan kelas sinyal ground truth menggunakan perhitungan akurasi, specificity, dan sensitivity.

Dilakukan beberapa uji coba untuk mencari parameter terbaik dari jumlah sinyal yang di averaging, level smoothing, jumlah epoch, nilai dropout probability, jumlah kernel fully connected layer, dan jumlah kernel dari convolutional layer. Dari beberapa skenario uji coba didapatkan hasil akurasi terbaik sebesar 70.80% untuk data BCI competition 2 dataset 2B, 84.03% untuk data BCI competition 3 dataset 2 subjek A dan 86.85% untuk data BCI competition 3 dataset 2 subjek B. Dari hasil ujicoba tersebut, BNNN dapat memberikan hasil yang baik untuk beberapa dataset.

Kata kunci: EEG; P300 event-related potential; CNN; BNNN .

P300 SIGNAL DETECTION USING BATCH NORMALISATION NEURAL NETWORK

Nama Mahasiswa	**: Adenuar Purnomo**
NRP	**: 05111440000079**
Departemen	**: Informatika FTIK-ITS**
Dosen Pembimbing 1	**: Prof. Ir. Handayani Tjandrasa, M.Sc., Ph.D.**
Dosen Pembimbing 2	**: Dini Adni Navastara, S.Kom., M.Sc.**

Abstract

Brain-computer interface (BCI) is a communication mechanism between electroencephalograph (EEG) signals and a computer, such that the system can capture the brain intention without involving motoric and muscular neurons. Detecting the P300 signal from electroencephalograph (EEG) is the key to interpreting the intent of the user. Convolutional Neural Network (CNN) can detect P300 signal well, but CNN tends to overfitting.

To solve this problem, a method called Batch Normalization Neural Network (BNNN) is implemented to detect P300 signals. The input signal is firstly cut 0-667ms after the stimulus, after that input signal is filtered using the Butterworth Filter. The amount of data each class on the input data is not balanced, so duplication is done to balance it.Then normalization is performed using min-max normalization, after that averaging signal is performed . The data is then refined using Haar Wavelet Transformation and inverse Haar Wavelet Transformation. Only then can the data be classified using the Batch Normalisation Neural Network. The performance of the proposed classification method is tested by comparing the

predicted signal class with the ground truth signal class using accuracy, specificity, and sensitivity calculations.

There are several tests to find the best parameters of the number of averaging signal, smoothing level, epoch number, dropout probability value, number of kernel fully connected layer, and number of kernels from the convolutional layer. From several test scenarios, the best accuracy result is 70.80% for BCI competition 2 dataset 2B data, 84.03% for BCI competition 3 dataset 2 subject A and 86.85% for BCI competition 3 dataset 2 subject B. From the result of the test, BNNN can give good results for some datasets.

Keywords: EEG; P300 event-related potential; CNN; BNNN.

KATA PENGANTAR

Puji syukur kepada Tuhan YME, yang telah melimpahka rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul **"DETEKSI SINYAL P300 MENGGUNAKAN METODE BATCH NORMALISATION NEURAL NETWORK".** Dengan pengerjaan Tugas Akhir ini, penulis bisa belajar lebih banyak untuk memperdalam dan meningkatkan apa yang telah didapatkan penulis selama menempuh perkuliahan di Teknik Informatika ITS. Dengan Tugas Akhir ini penulis juga dapat menghasilkan suatu implementasi dari apa yang telah penulis pelajari.

Selesainya Tugas Akhir ini tidak lepas dari bantuan dan dukungan beberapa pihak. Sehingga pada kesempatan ini penulis mengucapkan syukur dan terima kasih kepada:

1. Keluarga penulis yang telah memberikan dukungan moral dan material serta do'a yang tak terhingga untuk penulis. Serta selalu memberikan semangat dan motivasi pada penulis dalam mengerjakan Tugas Akhir ini.
2. Ibu Prof. Ir. Handayani Tjandrasa, M.Sc., Ph.D. selaku pembimbing I yang telah membantu, membimbing, dan memotivasi penulis dalam menyelesaikan Tugas Akhir ini dengan sabar.
3. Ibu Dini Adni Navastara, S.Kom., M.Sc. selaku pembimbing II yang juga telah membantu, membimbing, dan memotivasi kepada penulis dalam mengerjakan Tugas Akhir ini.
4. Bapak Dr. Darlis Herumurti, S.Kom., M.Kom. selaku Kepala Departemen Informatika ITS.
5. Bapak Radityo Anggoro, S.Kom., M.Sc. selaku koordinator Tugas Akhir, dan segenap dosen Informatika yang telah memberikan ilmunya.

6. Teman-teman angkatan 2014 yang yang telah membantu, berbagi ilmu, menjaga kebersamaan, dan memberi motivasi kepada penulis.
7. Serta semua pihak yang yang telah turut membantu penulis dalam menyelesaikan Tugas Akhir ini.

Penulis menyadari bahwa Tugas Akhir ini masih memiliki banyak kekurangan. Sehingga dengan kerendahan hati, penulis mengharapkan kritik dan saran dari pembaca untuk perbaikan ke depannya.

Surabaya, Juni 2018

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 dijelaskan beberapa hal dasar yang meliputi latar belakang, tujuan, permasalahan, batasan masalah, metodologi serta sistematika penulisan tugas akhir, sehingga gambaran umum permasalahan dan pemecahan yang diambil dapat dipahami dengan baik.

1.1 Latar Belakang

Pasien yang menderita *amyotrophic lateral sclerosis* (ALS), penyakit Parkinson atau cacat motor lainnya biasanya memerlukan pendekatan komunikasi langsung. Sebuah *brain–computer interface* (BCI) dapat menjembatani otak seseorang secara langsung ke dunia luar tanpa memerlukan aktivitas otot dari subjek [1]. *Brain-computer interface* (BCI) adalah mekanisme komunikasi antara sinyal *electroencephalograph* (EEG) dan komputer dimana sehingga sistem dapat menangkap niat otak tanpa melibatkan neuron motorik dan muskular. Mendeteksi sinyal P300 dari EEG adalah kunci untuk menafsirkan maksud dari pengguna. P300 *event-related potential* (ERP) adalah salah satu sinyal EEG yang umum digunakan dalam *speller systems*. Namun P300 ERP yang didapat sering memiliki rasio *signal-to-noise* yang sangat rendah, sehingga rangsangan harus diulang untuk meningkatkan kualitas *speller*[2], namun secara lebih mendasar, klasifikasi P300 yang benar berdasarkan pengulangan yang sedikit lebih penting untuk membangun sistem speller yang lebih cepat [3].

Penelitian klasifikasi sinyal P300 terdahulu lebih berfokus kepada optimisasi teknik pembelajaran tradisional seperti SVM dan LDA. Rakotomamonjy dan Guigue [4] mengelompokkan 17 klasifikasi SVM, mencapai rata-rata akurasi 96,5% dalam prediksi karakter, berada di peringkat teratas dalam kompetisi BCI 2004 Data Set II [5]. Metode yang diusulkan menyaring sinyal menjadi 0,1-10 Hz kemudian melakukan *down-sampling* dan merata-rata untuk mengurangi *noise* dan variabilitas. Kemudian, digunakan

algoritma ecursive channel elimination untuk memilih jumlah channel terbaik untuk subjek yang berbeda. Li dan teman-teman [6] menerapkan ICA untuk koreksi artefak okular sebelum klasifikasi SVM. Bostanov [7] menggunakan LDA untuk mendapatkan hasil yang sebanding dengan SVM dalam kompetisi BCI II [8].

Selama dekade terakhir, deep learning, sebagai sub bidang machine learning, telah membuat kemajuan mengesankan dalam memecahkan masalah dunia nyata seperti visi komputer. Arsitektur seperti VGG net [9] dan Resnet [10] telah diusulkan, diperkuat dengan kemampuan untuk menangkap fitur data berdimensi banyak yang hierarkis. Namun, aplikasi deep learning pada deteksi sinyal EEG masih di tahap awal [11]. Cecotti dan Graser [12] mengembangkan Convolutional Neural Network (CNN) berlapis empat untuk mencapai hasil yang sebanding dengan SVM pada penelitian milik Rakotomamonjy dan Guigue. Karakteristik spasial dan temporal dari sinyal input P300 masing-masing diekstrak oleh lapisan pertama dan kedua dari CNN dan kemudian dikirim ke lapisan klasifikasi. Baru-baru ini, Langkvist dan teman-teman [13] mengusulkan Deep Belief Network(DBN) untuk melakukan ekstraksi fitur otomatis pada data mentah. Wulsin dan teman-teman [14] memodelkan bentuk gelombang EEG saluran tunggal dengan DBN untuk klasifikasi dan deteksi anomali dalam paradigma semi-supervised. Stober dan teman-teman [15] menggabungan CNN dengan lapisan output DLSVM untuk mendapatkan klasifikasi ritme di antara individu. Namun, belum banyak metode yang tepat yang diusulkan untuk klasifikasi P300 yang cepat dalam real world speller systems. CNN dapat mendeteksi sinyal P300 dengan baik, namun CNN cenderung overfitting.

Dalam tugas akhir ini, penulis mengimplementasikan sebuah metode untuk mendeteksi sinyal P300. Tujuan penulis adalah memanfaatkan *Batch Normalisation Neural Network* (BNNN) untuk mendeteksi sinyal P300. Motivasi di balik tujuan ini adalah bahwa BNNN dapat digunakan untuk menangkap ciri-ciri sinyal EEG. Penulis juga mengusulkan penggunaan Batch

Normalisation dan Dropout untuk meningkatkan penggeneralisasian model kami dan menggunakan aktivasi ReLU di lapisan konvolusi untuk mempercepat pelatihan. CNN dan Batch Normalisation digabung menjadi sebuah model yang dinamakan BNNN (Batch Normalisation Neural Network). Model yang penulis usulkan menggabungkan ekstraksi fitur dan klasifikasi menjadi satu proses sehingga membutuhkan proses yang lebih sedikit

Hasil tugas akhir ini diharapkan dapat memberikan metode yang lebih baik untuk deteksi sinyal P300 sehingga dapat menyelesaikan permasalahan di atas dengan optimal dan diharapkan dapat memberikan kontribusi pada perkembangan ilmu pengetahuan dan teknologi informasi.

1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam Tugas Akhir ini adalah sebagai berikut:

1. Bagaimana menyiapkan data sinyal pada tahap *preprocessing* untuk diklasifikasi?
2. Bagaimana melakukan deteksi sinyal P300 menggunakan metode *Batch Normalisation Neural Network*?
3. Bagaimana mencari parameter arsiterktur *Batch Normalisation Neural Network* terbaik?
4. Bagaimana menghitung akurasi, *specificity* dan *sensitivity* hasil deteksi sinyal P300 menggunakan metode *Batch Normalisation Neural Network*?

1.3 Batasan Masalah

Batasan dalam Tugas Akhir ini, yaitu:

1. Metode untuk deteksi sinyal P300 yang digunakan adalah *Batch Normalisation Neural Network*.
2. Percobaan akan dilakukan menggunakan Matlab.
3. Dataset yang digunakan untuk percobaan adalah dataset yang tersedia untuk umum (publik) yaitu dataset BCI III dataset II dan BCI II dataset IIB yang diambil dari *Berlin Brain-Computer Interface*.

1.4 Tujuan

Tujuan dari pembuatan tugas akhir ini yaitu untuk mendeteksi sinyal P300 menggunakan metode *Batch Normalisation Neural Network* dan mengevaluasi hasil deteksi menggunakan akurasi, *specificity,* dan *sensitivity.*

1.5 Manfaat

Tugas akhir ini diharapkan dapat membantu pasien yang mengalami cacat motoric untuk berkomunikasi.

1.6 Metodologi

Tahapan-tahapan yang dilakukan dalam pengerjaan Tugas Akhir ini adalah sebagai berikut:

1. Penyusunan proposal Tugas Akhir.

   Tahap awal untuk memulai pengerjaan Tugas Akhir adalah penyusunan proposal Tugas Akhir. Proposal Tugas Akhir yang diajukan memiliki gagasan yang sama dengan Tugas Akhir ini. Penyusunan proposal Tugas Akhir dilaksanakan untuk merumuskan masalah serta melakukan penetapan rancangan dasar dari sistem yang akan dikembangkan dalam pelaksanaan Tugas Akhir ini.
2. Studi literatur

   Pada tahap ini dilakukan pencarian dan pemahaman informasi dan literatur baik melalui media internet maupun dengan melakukan studi pustaka pada jurnal-jurnal ilmiah cetak yang diperlukan untuk tahap implementasi program.
3. Analisis dan perancangan perangkat lunak

   Tahap ini meliputi perancangan sistem berdasarkan studi literatur dan pembelajaran konsep teknologi dari perangkat lunak yang ada. Tahap ini mendefinisikan alur dari implementasi. Langkah-langkah yang dikerjakan juga didefinisikan pada tahap ini. Pada tahapan ini dibuat *prototype* sistem, yang merupakan rancangan dasar dari sistem yang akan dibuat.

4. Implementasi perangkat lunak

   Implementasi perangkat lunak merupakan tahap membangun rancangan program yang telah dibuat. Program yang dibuat menggunakan Matlab R2018a.

5. Pengujian dan evaluasi

   Pada tahap ini dilakukan uji coba pada data yang telah dikumpulkan. Tahap ini digunakan untuk mengevaluasi kinerja program serta mencari masalah yang mungkin timbul saat program dievaluasi serta melakukan perbaikan jika terdapat kesalahan pada program.

6. Penyusunan buku Tugas Akhir.

   Pada tahap ini disusun buku yang memuat dokumentasi mengenai perancangan, pembuatan serta hasil dari implementasi perangkat lunak yang telah dibuat.

1.7 Sistematika Penulisan Laporan Tugas Akhir

Buku Tugas Akhir ini bertujuan untuk mendapatkan gambaran dari pengerjaan Tugas Akhir ini. Secara garis besar, buku Tugas Akhir terdiri atas beberapa bagian seperti berikut ini:

Bab I Pendahuluan

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

Bab II Tinjauan Pustaka

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

Bab III Desain Sistem

Bab ini berisi tentang rancangan sistem yang akan dibangun dan disajikan dalam bentuk diagram alir.

Bab IV Implementasi

Bab ini membahas mengenai implementasi dari rancangan yang telah dibuat pada bab sebelumnya. Penjelasan berupa *code* yang digunakan untuk proses implementasi.

Bab V Uji Coba Dan Evaluasi

Bab ini menjelaskan mengenai kemampuan perangkat lunak dengan melakukan pengujian kebenaran dan pengujian kinerja dari perangkat lunak yang telah dibuat sesuai dengan data yang diujikan.

Bab VI Kesimpulan Dan Saran

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

BAB II
TINJAUAN PUSTAKA

Bab ini berisi penjelasan teori-teori yang berkaitan dengan algoritma yang diajukan pada pengimplementasian program. Penjelasan ini bertujuan untuk memberikan gambaran secara umum terhadap program yang dibuat dan berguna sebagai penunjang dalam pengembangan perangkat lunak.

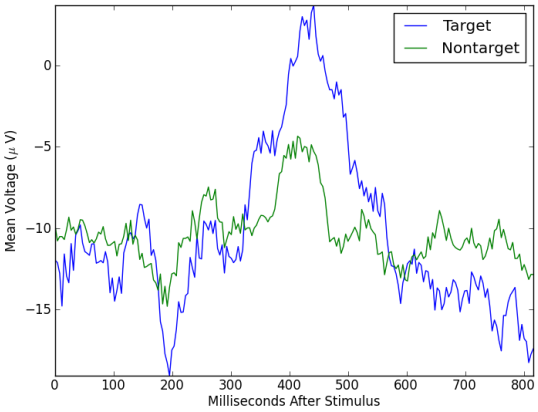
2.1 P300

P300 *Speller Paradigm*, yang juga disebut *Oddball Paradigm* pertama kali diajukan oleh Farwell dan Dochin. Matriks 6×6 huruf dengan baris dan kolom berpendar dalam urutan acak dipresentasikan kepada pengguna selama percobaan berlangsung. Ketika pengguna melihat sebuah huruf dalam matriks, sebuah ERP akan didapat jika huruf sasaran terkandung dalam baris atau kolom yang sedang berpendar. Dengan kondisi ini, ERP terdeteksi sebagai puncak positif EEG tegangan 300 ms setelah stimulus yang dinamakan P300. Setelah pengulangan *flashing*, kita dapat mengumpulkan pendeteksian sinyal P300 dan mengambil perpotongan dari baris dan kolom dengan sebagian besar pendeteksian sebagai karakter yang dimaksud. Contoh P300 Speller matrix dapat dilihat pada Gambar 2.1. Sedangkan perbedaan antara sinyal P300 dan bukan sinyal P300 dapat dilihat pada Gambar 2.2.

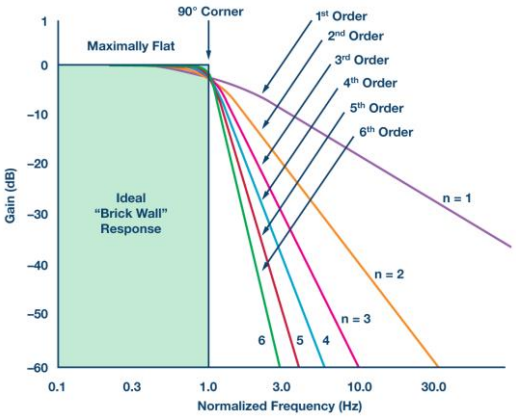
2.2 Butterworth Filter

Butterworth merupakan salah satu metode desain filter analog sistematis yang paling awal, dan masih merupakan salah satu yang paling banyak digunakan. Metode desain filter Butterworth adalah salah satu prosedur desain filter klasik. Metode desain filter klasik lainnya adalah Chebyshev Tipe I, Chebyshev Tipe II, eliptik (atau Cauer), dan Bessel[16]. Gambar grafik *gain* dari Butterworth Filter orde ke 1 sampai 5 dapat dilihat pada gambar 2.3

Gambar 2.1 P300 Speller Matrix [29]



Gambar 2.2 Perbedaan Sinyal P300(Target) dengan Sinyal Non-P300(Nontarget) [29]



Gambar 2.3 Grafik gain dari Butterworth Filter orde ke 1 sampai 5 [16]

2.3 Normalisasi

Normalisasi adalah teknik penskalaan atau teknik pemetaan atau tahap pra-pemrosesan [17]. Di mana, kita dapat menemukan rentang baru dari rentang satu yang ada. Ini dapat sangat membantu untuk memprediksi atau meramal [18]. Seperti yang diketahui ada banyak cara untuk memprediksi atau meramalkan tetapi semua bisa sangat berbeda satu sama lain. Jadi untuk mempertahankan banyaknya variasi prediksi dan peramalan, teknik Normalisasi diperlukan untuk membuatnya lebih akurat. Ada beberapa teknik normalisasi yang sudah ada seperti yaitu Min-Max, Zscore & Desimal. Rumus dari teknik normalisasi Min-Max dapat dilihat pada persamaan 2.1.

$$x = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{2.1}$$

2.4 Signal Averaging

Signal averaging menurut Tompkins dan Webster[19] adalah sebuah Teknik untuk memisahkan sinyal yang berulang dari

noise tanpa mendistorsi sinyal tersebut. Salah satu area aplikasi dominan dari *signal averaging* adalah di *electroencephalography*. EEG yang direkam dari elektroda kulit kepala sulit ditafsirkan sebagian karena terdiri dari penjumlahan aktivitas miliaran sel otak. Tidak mungkin untuk menyimpulkan aktivitas visual atau pendengaran otak dari EEG. Namun, jika kita menstimulasi bagian otak dengan kilatan cahaya atau klik akustik, respon yang timbul terjadi di wilayah otak yang memproses informasi untuk sistem sensorik yang dirangsang. Dengan menjumlahkan sinyal yang dibangkitkan segera setelah banyak rangsangan dan membaginya dengan jumlah total rangsangan, kita mendapatkan tanggapan yang dirata-ratakan rata-rata. Sinyal ini dapat mengungkapkan banyak hal tentang kinerja sistem sensorik. *Signal averaging* menjumlah satu set periode waktu dari sinyal dan *noise*. Jika waktu epochs selaras, bentuk gelombang sinyal menguat. Di sisi lain, *noise* yang ada menjadi samar. Dengan demikian, *signal-to-noise ratio* (SNR) meningkat[20]. Contoh dari *signal averaging* dapat dilihat pada gambar 2.5.

Signal averaging dapat dilakukan apabila sinyal dan noise memiliki karakteristik seperti berikut:

a. Bentuk gelombang sinyal harus berulang (meskipun tidak harus periodik). Artinya sinyal harus terjadi lebih dari satu kali tetapi tidak harus secara berkala.

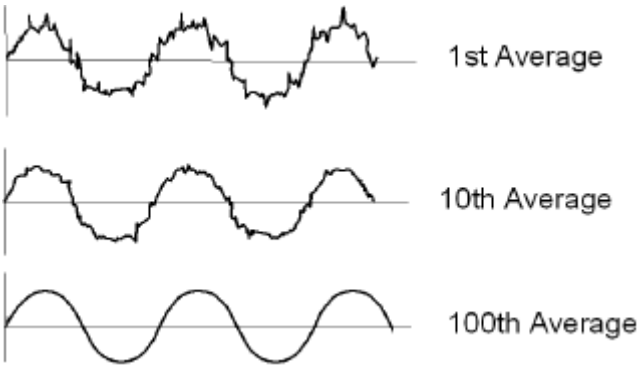
b. *Noise* harus acak dan tidak berkorelasi dengan sinyal. Acak berarti bahwa sinyal tidak bersifat periodik dan hanya dapat dideskripsikan secara statistik (misalnya, dengan mean dan variansnya).

c. Posisi temporal setiap gelombang sinyal harus diketahui secara akurat.

2.5 Haar Wavelet Transformation

Haar Wavelet adalah jenis wavelet yang paling sederhana. Dalam bentuk diskrit, Haar wavelet berhubungan dengan operasi matematika yang disebut transformasi Haar. Transformasi Haar berfungsi sebagai prototipe untuk semua transformasi wavelet

lainnya. Seperti semua transformasi wavelet, transformasi Haar menguraikan diskrit



Gambar 2.4 Contoh Signal Averaging pada 10 sinyal dan 100 sinyal [30]

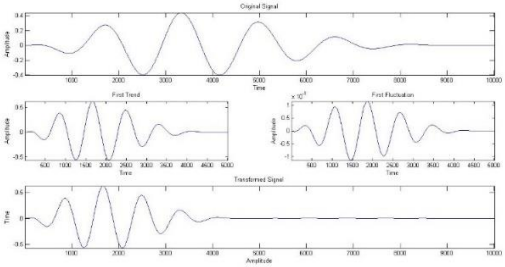
Sinyal menjadi dua sub-sinyal yang panjangnya setengah dari sinyal awal. Satu sub-sinyal adalah nilai rata-rata atau tren, sedangkan sub-sinyal lainnya adalah perbedaan atau fluktuasi yang berjalan. Transformasi Haar dilakukan dalam beberapa tahap, atau level[21].

Banyak metode telah dikembangkan untuk menghaluskan sinyal, dengan harapan bahwa kebisingan dapat ditekan dan pola yang signifikan dipertahankan dan terungkap. Ini telah berkisar dari rata-rata bergerak sederhana atau median bergerak ke metode kompleksitas matematika yang cukup besar. Wavelet tampaknya menawarkan pendekatan penghalusan yang relatif mudah digunakan, sementara beradaptasi dengan baik, dan secara otomatis, ke bentuk sinyal yang dihaluskan.[22] Contoh transformasi wavelet haar di level 1 dapat dilihat pada gambar 2.6.

2.6 Inverse Haar Wavelet Transformation

Inverse Haar Wavelet Transformation merupakan kebalikan dari Haar Wavelet Transformation. Inverse Haar Wavelet Transformation digunakan untuk merekonstruksi data

yang telah dipecah di dalam Haar Wavelet Transformation. Dalam Inverse Haar Wavelet Transformation data dapat direkonstruksi sampai satu level dibawah level yang digunakan pada Haar Wavelet Transformation [23].



Gambar 2.5 Contoh Haar Wavelet Transformation [31]

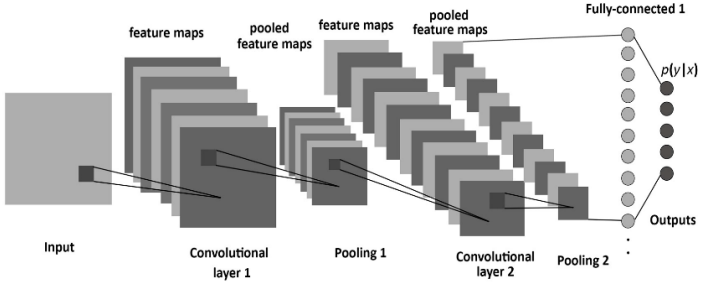
2.7 Deep Learning

Deep Learning adalah subfield dari *machine learning* yang berkaitan dengan algoritma yang terinspirasi oleh struktur dan fungsi otak yang disebut jaringan syaraf tiruan. Andrew Ng dari Coursera dan Chief Scientist di Baidu Research secara resmi mendirikan Google Brain yang pada akhirnya menghasilkan productisasi teknologi *deep learning* di sejumlah besar layanan Google. Dalam seminar tahun 2013 yang berjudul "*Deep Learning, Self-Taught Learning and Unsupervised Feature Learning*" dia menggambarkan ide *deep learning* seperti "menggunakan simulasi otak, berharap untuk membuat algoritma pembelajaran yang jauh lebih baik dan lebih mudah digunakan dan membuat kemajuan revolusioner dalam pembelajaran mesin dan AI." [24]

2.8 CNN

Convolutional Neural Network (CNN) terdiri dari satu atau lebih lapisan konvolusi dan kemudian diikuti oleh satu atau lebih lapisan yang saling terhubung seperti pada jaringan syaraf multilayer standar. Arsitektur CNN dirancang untuk memanfaatkan struktur 2D dari gambar masukan (atau input 2D

lainnya seperti sinyal ucapan). Hal ini dicapai dengan koneksi lokal dan bobot diikuti oleh beberapa bentuk penyatuan yang menghasilkan fitur invarian terjemahan. Manfaat lain dari CNN adalah CNN lebih mudah untuk dilatih dan memiliki lebih sedikit parameter daripada jaringan yang terhubung sepenuhnya dengan jumlah unit tersembunyi yang sama [25]. Gambaran arsitektur secara umum dapat dilihat pada gambar 2.7



Gambar 2.6 Arsitektur CNN [32]

2.9 Convolutional Layer

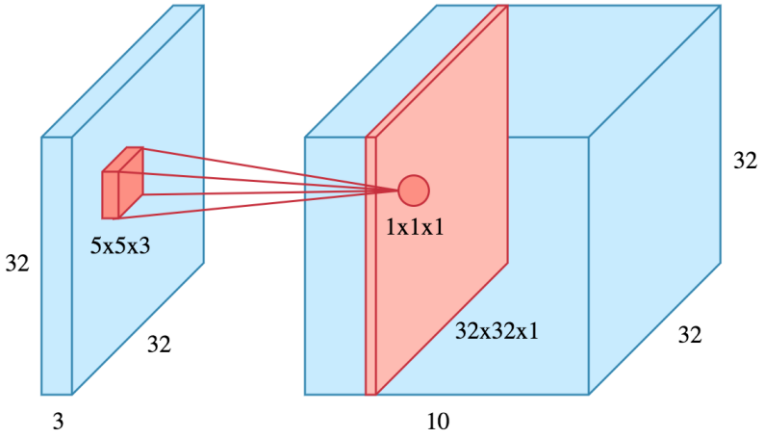
Lapisan konvolusional terdiri dari neuron yang terhubung ke daerah kecil dari input atau lapisan sebelumnya. Daerah ini disebut filter. Anda dapat menentukan ukuran wilayah ini menggunakan argumen input *filterSize*. Untuk setiap wilayah, perangkat lunak menghitung perkalian *dot* dari bobot dan masukan, dan kemudian menambahkan bias. Filter kemudian bergerak sepanjang input secara vertikal dan horizontal, mengulangi perhitungan yang sama untuk setiap wilayah. Ukuran langkah yang bergerak disebut *stride*. Anda dapat menentukan ukuran langkah ini dengan properti *Stride*. Daerah-daerah lokal yang terhubung dengan neuron mungkin tumpang tindih tergantung pada ukuran filter dan langkahnya.

Jumlah bobot yang digunakan untuk filter adalah *h*w*c*, di mana *h* adalah tinggi, dan *w* adalah lebar ukuran filter, dan *c* adalah jumlah saluran dalam input (misalnya, jika input adalah gambar berwarna, maka jumlah salurannya adalah tiga). Saat filter

bergerak sepanjang input, ia menggunakan kumpulan bobot dan bias yang sama untuk konvolusi, membentuk peta fitur. Lapisan konvolusi biasanya memiliki beberapa peta fitur, masing-masing dengan kumpulan bobot dan bias yang berbeda. Jumlah filter menentukan jumlah peta fitur.

Jumlah total parameter dalam lapisan konvolusional adalah *((h*w*c + 1)*Jumlah Filter)*, di mana *l* untuk bias. Tinggi dan lebar keluaran dari lapisan konvolusional adalah *(Ukuran Input - Ukuran Filter + 2 *Padding) / Stride + 1*. Nilai ini harus berupa bilangan bulat agar seluruh gambar tertutup sepenuhnya. Jika kombinasi parameter ini tidak menyebabkan gambar sepenuhnya tertutup, maka perangkat lunak secara default mengabaikan bagian gambar yang tersisa di sepanjang tepi kanan dan bawah dalam konvolusi.

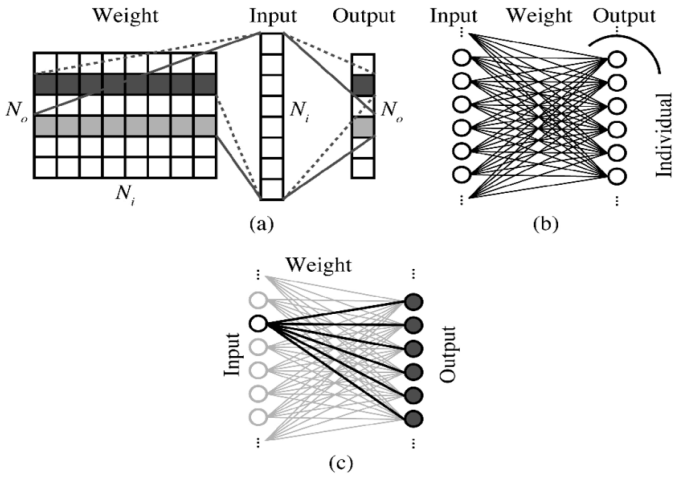
Jumlah total neuron dalam peta fitur, katakanlah Ukuran Peta, adalah produk dari tinggi dan lebar keluaran. Jumlah total neuron (ukuran output) dalam lapisan konvolusional, kemudian, adalah *Ukuran Peta * Jumlah Filter* [26]. Ilustrasi tentang Convolutional Layer dapat dilihat pada gambar 2.8.



Gambar 2.7 Contoh convolutional layer [33]

2.10 Fully Connected Layer

Fully Connected Layer mengalikan input dengan matriks bobot *W* dan kemudian menambahkan vektor bias *b*. Jika input ke lapisan adalah urutan (misalnya, dalam jaringan LSTM), maka Fully Connected Layer bertindak secara independen pada setiap langkah waktu. Sebagai contoh, jika layer sebelum layer yang terhubung sepenuhnya menghasilkan array *X* dengan ukuran *D-by-N-by-S*, maka layer yang terhubung sepenuhnya menghasilkan array *Z* dari ukuran *output Size-by-N-by-S*. Pada langkah waktu *t*, entri *Z* yang sesuai adalah $WX_t + b$, di mana X_t menunjukkan waktu langkah *t* dari *X* [27]. Default untuk bobot awal di perangkat lunak Matlab adalah distribusi Gaussian dengan mean 0 dan standar deviasi 0,01. Default untuk bias awal adalah 0. Ilustrasi Fully Connected Layer dapat dilihat pada gambar 2.9.



Gambar 2.8 Cara kerja Fully Connected Layer [34]

2.11 Batch Normalisation

Teknik untuk meningkatkan kinerja dan stabilitas jaringan syaraf tiruan, dan juga membuat arsitektur pembelajaran mendalam yang lebih canggih. Ide dari *Batch Normalisation* adalah

menormalkan masukan dari setiap lapisan sedemikian rupa sehingga mereka memiliki aktivasi keluaran rata-rata dari nol dan standar deviasi satu[28].

Tujuan di balik normalisasi batch adalah untuk mengoptimalkan pelatihan jaringan. Telah terbukti memiliki beberapa manfaat:

a. Belajar lebih cepat

Setiap iterasi pelatihan akan lebih lambat karena perhitungan normalisasi ekstra dan parameter tambahan saat melakukan proses pembelajaran. Namun, secara keseluruhan proses pelatihan berjalan lebih cepat.

b. Memungkinkan tingkat belajar yang lebih tinggi

Semakin dalam jaringan, gradien semakin kecil saat *back propagation*, dan memerlukan lebih banyak iterasi. Menggunakan *Batch Normalisation* memungkinkan tingkat belajar dengan akurasi yang jauh lebih tinggi, meningkatkan kecepatan melatih jaringan.

c. Membuat bobot lebih mudah diinisialisasi

Inisialisasi berat bisa sulit, terutama saat membuat jaringan yang lebih dalam. *Batch Normalisation* membantu mengurangi sensitivitas terhadap bobot awal awal.

d. Membuat lebih banyak fungsi aktivasi yang layak

Beberapa fungsi aktivasi tidak bekerja dengan baik dalam situasi tertentu. *Sigmoids* kehilangan gradien mereka dengan cepat, yang berarti tidak dapat digunakan di *deep networks*, dan ReLUs sering mati saat pelatihan (berhenti belajar), jadi kita harus berhati-hati dengan kisaran nilai yang diberikan ke dalamnya. Tetapi karena *batch normalisation* mengatur nilai yang masuk ke setiap fungsi aktivasi, *nonlinearities* yang tidak bekerja dengan baik di jaringan dalam cenderung menjadi layak lagi.

e. Menyederhanakan penciptaan jaringan yang lebih dalam

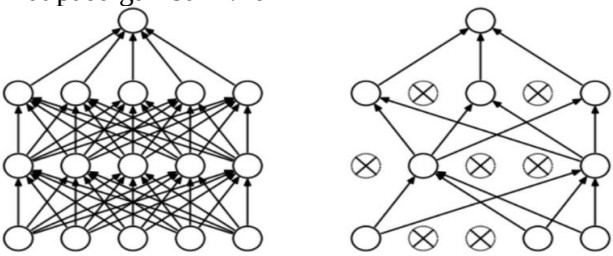
4 poin sebelumnya membuatnya lebih mudah dibangun dan lebih cepat untuk melatih jaringan syaraf yang lebih dalam, dan jaringan yang lebih dalam umumnya menghasilkan hasil yang lebih baik.

f. Menyediakan beberapa regularisasi

Batch Normalisation menambahkan sedikit *noise* pada jaringan, dan dalam beberapa kasus, (misalnya *Inception modules*) telah terbukti berhasil dan juga *dropout*. *Batch Normalisation* dapat dipertimbangkan sebagai sedikit regularisasi tambahan, yang memungkinkan Anda mengurangi beberapa *dropout* yang mungkin ditambahkan ke jaringan.

2.12 Dropout

Dropout adalah algoritma yang diperkenalkan untuk mengurangi kompleksitas adaptasi antar neuron, yang memaksa setiap neuron untuk mempelajari fitur yang lebih kuat. *Dropout* meningkatkan generasi jaringan dengan menggabungkan tiga pendekatan standar untuk menghindari *overfitting*: *ensemble* rata-rata, menambahkan *noise*, dan menambahkan regularisasi *terms* ke *error functions*. Secara khusus, *dropout* telah menyebabkan perbaikan besar pada beberapa masalah pengenalan citra dan suara yang sulit. Misalkan p menunjukkan tingkat *dropout* dari suatu *layer*, setiap neuron pada *layer* ini memiliki probabilitas 1-p untuk diisi nilai 0 selama tahap pelatihan *forward propagation*. Selama tahap pengujian, setiap neuron dipertahankan, sehingga keseluruhan model dapat diartikan sebagai yang rata-rata dari banyak pengklasifikasi. Nilai p yang tepat membantu neuron mempelajari fitur yang lebih berbeda dan kuat sehingga kurang memungkinkan untuk *overfitting* [29] . Ilustrasi algoritma dropout dapat dilihat pada gambar 2.10



Gambar 2.9 Ilustrasi algoritma Dropout [35]

[Halaman ini sengaja dikosongkan]

BAB III
DESAIN SISTEM

Tugas akhir ini berfokus pada klasifikasi terhadap masukan sinyal EEG yang digunakan, yang diperoleh dari *BCI Competition* II, dataset 2B dan *BCI Competition* III, dataset 2.

Bab ini membahas tentang desain dari sistem klasifikasi sinyal EEG dengan *Batch Normalisation Neural Network.* Deskripsi yang disampaikan meliputi lingkungan desain dilakukan dan desain sistem. Desain sistem sendiri meliputi desain data dan desain proses.

3.1 Lingkungan Desain dan Implementasi

Subbab ini akan menjelaskan mengenai lingkungan desain dan implementasi perangkat lunak yang akan dibangun. Spesifikasi perangkat keras yang digunakan dalam desain dan implementasi perangkat lunak adalah prosesor berjenis Intel(R) Core(TM) i7-4720HQ CPU @ 2.60GHz (8 CPUs), ~2.6GHz, dengan kapasitas memori (RAM) sebesar 8.192 GB.

Sistem operasi yang digunakan adalah Microsoft Windows 10 Home Single Language 64-bit, sedangkan perangkat lunak yang digunakan dalam proses implementasi adalah MATLAB R2018a.

3.2 Desain Sistem

Pada sub bab ini dijelaskan mengenai desain sistem klasifikasi terhadap data sinyal EEG menggunakan *Batch Normalisation Neural Network* dengan kombinasi *Haar Wavelet Transformation.* Desain sistem ini meliputi desain data dan desain proses. Desain data berisi penjelasan data yang diperlukan untuk dapat menerapkan algoritma ini, meliputi data masukan, data proses dan data keluaran. Desain proses berupa algoritma yang digunakan dalam sistem dan digambarkan dengan diagram alir.

3.2.1 Desain data

Pada sub bab ini dijelaskan mengenai perancangan data yang digunakan dalam pengoperasian perangkat lunak. Perancangan data merupakan hal penting untuk diperhatikan karena diperlukan data yang tepat dan sesuai agar perangkat lunak dapat beroperasi dengan benar. Data yang diperlukan dalam pengoperasian perangkat lunak ini terbagi menjadi tiga macam data, yaitu data masukan (*input*), data yang digunakan selama proses eksekusi perangkat lunak, dan data keluaran (*output*) yang merupakan hasil proses pengoperasian perangkat lunak.

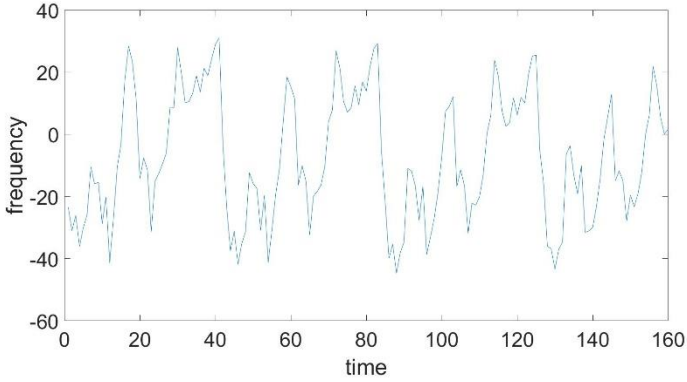
3.2.1.1 Data Masukan

Data masukan pada sistem ini adalah sinyal EEG yang diperoleh dari *BCI Competition* II dataset 2B. Sinyal yang tersedia, diambil dari hasil perekaman 64 *channel* elektroda. Matriks dataset mentah sinyal yang diperoleh berdimensi *S x C*, dimana *S* menandakan jumlah *samples*, dan *C* merupakan jumlah *channel*. Terdapat total 12930 sinyal pada dataset, dengan 10950 sinyal merupakan kelas 0 dan 1980 sinyal kelas 1. Kelas 0 merepresentasikan sinyal non-P300, sedangkan kelas 1 merepresentasikan sinyal P300. Sedangkan data dari *BCI Competition* III dataset 2 memiliki sinyal yang diambil dari hasil perekaman 64 *channel* elektroda. Matriks dataset mentah sinyal yang diperoleh berdimensi *T x S x C*, dimana *T* menandakan jumlah *trial*, S menandakan jumlah *samples*, dan *C* merupakan jumlah *channel*. Terdapat total 65750 sinyal pada dataset, dengan 55500 sinyal *merupakan* kelas 0 dan 10250 sinyal kelas 1. Kelas 0 merepresentasikan sinyal non-P300, sedangkan kelas 1 merepresentasikan sinyal P300. Gambar 3.1 merepresentasikan sampel data masukan sinyal ini.

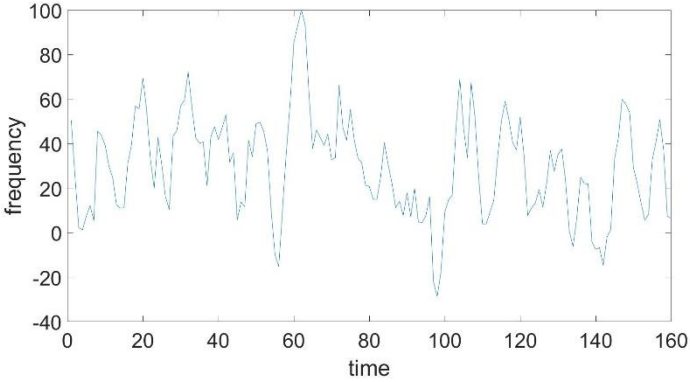
3.2.1.2 Data Proses

Terdapat beberapa tahapan dalam yang dilalui mulai dari proses noise filtering hingga klasifikasi terhadap sinyal EEG sebagai masukan sistem. Pada masing-masing tahapan tersebut terdapat data yang menjadi keluaran suatu tahap atau proses, dan kemudian

menjadi masukan suatu tahap atau proses berikutnya. Data-data atau variabel penting yang digunakan pada proses ini dapat dilihat pada Tabel 3.1.



(a)



(b)

Gambar 3.1 (a) Contoh Data sampel kelas P300 sinyal EEG awal (b) Contoh Data sampel kelas non-P300 sinyal EEG awal

3.2.1.3 Data Keluaran

Data keluaran dari aplikasi ini adalah matriks *result*, yang berisi kelas dari setiap *testing* data. Kelas 0 merupakan representasi sinyal non-P300, sedangkan kelas 1 merupakan representasi sinyal P300.

Data keluaran yang lain berupa nilai akurasi, *spesificity* dan *sensitivity* terhadap hasil klasifikasi sinyal EEG. Hasil klasifikasi diperoleh berdasarkan pembandingan hasil identifikasi kelas pada matriks *result* dengan nilai kelas pada *truth label testing* data.

Tabel 3.1 Data proses dalam sistem

No.	Nama Data	Keterangan
1	*Flashing*	Stimulus dari intensifikasi
2	*StimulusCode*	Intensifikasi baris atau kolom kelas sinyal mentah
3	*Signal*	Sinyal mentah
4	*StimulusType*	Kelas dari sinyal mentah
5	*Input*	Sinyal masukan hasil pemrosesan
6	*Input_class*	Kelas dari sinyal masukan hasil pemrosesan

3.2.2 Desain Proses

Pada bagian ini dijelaskan mengenai algoritma aplikasi melalui diagram alir. Dengan menggunakan diagram alir, dapat diketahui data awal hingga akhir proses secara jelas.

Proses penelitian pada tugas akhir ini berfokus pada klasifikasi sinyal EEG dengan metode utama *Batch Normalisation Neural Network*. Secara garis besar, proses yang dilakukan melalui beberapa tahap utama, yaitu pemotongan sinyal, *noise filtering* untuk menghilangkan *noise* pada data sinyal mentah,normalisasi data, *signal averaging*, menghaluskan data yang dipakai dalam klasifikasi data, pembuatan model belajar dari data *training* dan dilanjutkan dengan klasifikasi terhadap setiap *testing* data menggunakan model *training*. Tahap *noise filtering* dilakukan menggunakan metode *bandpass filter* untuk *temporal filtering*. *Temporal filtering* bertujuan untuk menyaring *noise* pada data

mentah berupa frekuensi-frekuensi yang tidak mengandung nilai informasi yang dibutuhkan.Data hasil *noise filtering* kemudian dinormalisasi menggunakan metode *min-max normalization* dengan *range* -100 sampai 100. Pada tahap *signal averaging*, sinyal hasil normalisasi dijumlahkan kemudian dirata-rata, sehingga diharapkan perbedaan antara kelas P300 dan kelas non-P300 semakin terlihat.

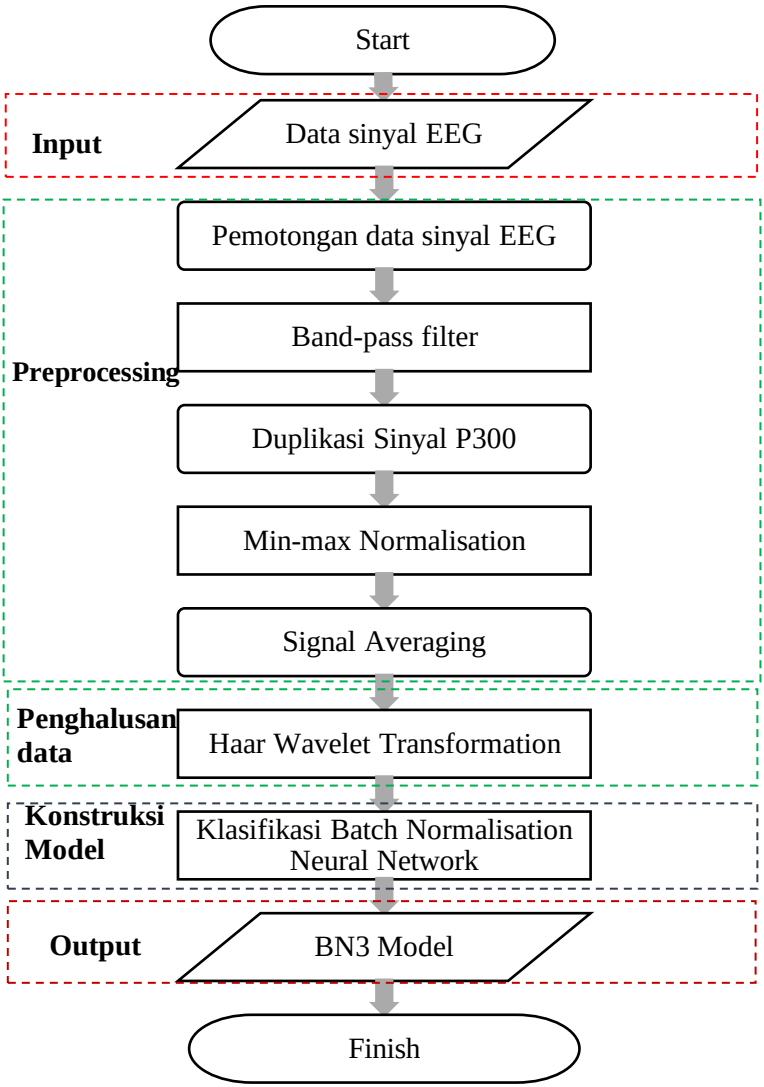
Pada tahap penghalusan data, diterapkan metode *Haar Wavelet Transformation* terhadap data sinyal hasil *averaging*. Proses ini dilakukan dengan tujuan memperhalus data, yang diklasifikasi. Tahap klasifikasi bertujuan untuk mengidentifikasi kelas suatu sinyal pada data *testing*. Kelas yang ada meliputi P300, dan non-P300. Untuk proses ini, metode yang digunakan adalah *Batch Normalisation Neural Network*(*BN3*).

Secara garis besar, proses-proses yang dikerjakan pada tugas ini digambarkan pada Gambar 3.2. dan Gambar 3.3. Gambar 3.2 menjelaskan garis besar pemrosesan *training* data, sedangkan Gambar 3.3 menjelaskan garis besar proses-proses terhadap *testing* data.

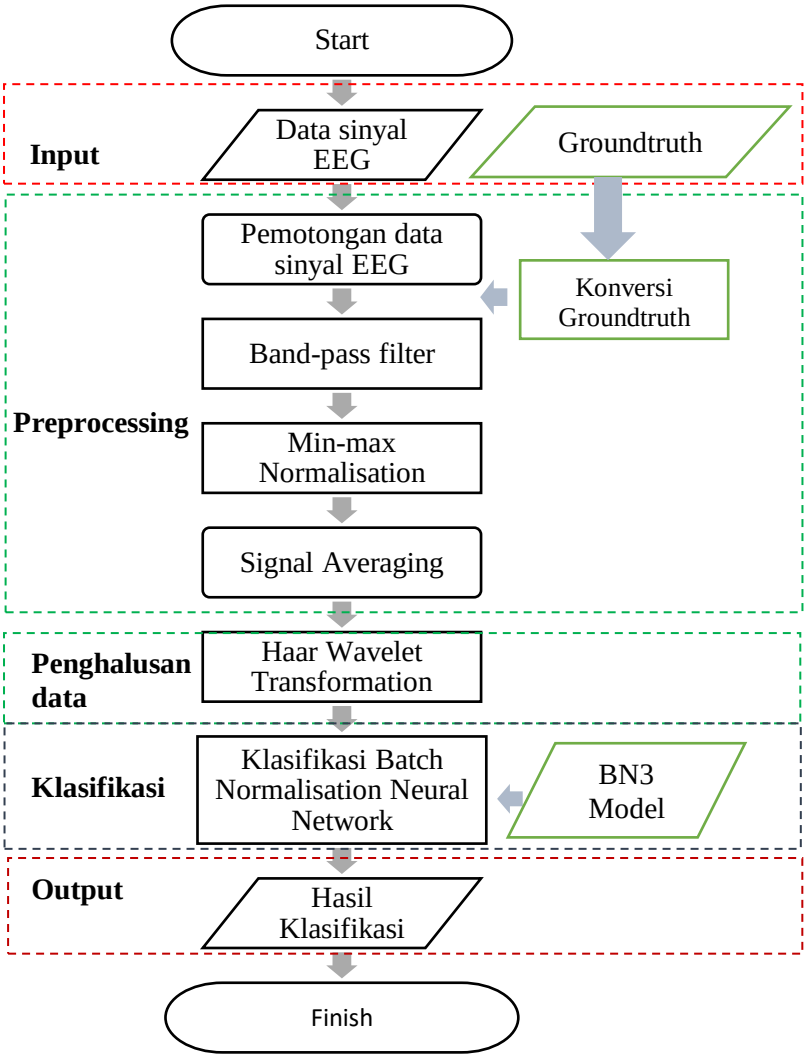
Pada pemrosesan *training* data, proses *BN3* pada *training* bukan bertujuan untuk klasifikasi, melainkan pembuatan model untuk klasifikasi data *testing*. Pada proses *testing*, model yang telah dibuat pada saat proses training digunakan untuk klasifikasi, berdasarkan *rule-rule* pada model yang terbentuk dari proses *training* sebelumnya.

3.2.3 Preprocessing

Pada tahap ini dilakukan pemrosesan pada sinyal EEG mentah. Tedapat empat sub proses utama dalam tahap ini yaitu pemotongan sinyal, Band-pass filter, min-max normalization, dan signal averaging. Terdapat perbedaan antara preprocessing data training dan data testing. Perbedaannya adalah pada data training groundtruth sudah merupakan sebuah matriks variabel sedangkan pada data testing groundtruth masih berupa huruf sehingga perlu diubah terlebih dahulu. Berikut penjelasan dari sub-sub proses tersebut.



Gambar 3.2 Diagram alir proses utama pada training data



Gambar 3.3 Diagram alir proses utama pada testing data

3.2.3.1 Pemotongan Sinyal

Pada tugas akhir ini dilakukan pemotongan sinyal 0-667ms setelah stimulus. Stimulus yang dimaksud dapat dilihat pada variabel *flashing*, dimana ketika flashing bernilai 1 maka terjadi stimulus, sedangkan 0 tidak terjadi stimulus. Stimulus memiliki Panjang 100ms atau 24 data, dimana stimulus dipakai sebagai acuan untuk memotong sinyal. Hasil dari pemotongan data berupa matriks berukuran *N x 160 x C*, dimana *N* merupakan jumlah sinyal yang akan diklasifikasi dan *C* adalah banyaknya channel. Untuk data *training* selain sinyal yang dipotong, dilakukan juga pemotongan variabel kelas sinyal tersebut, sedangkan pada data testing selain pemotongan sinyal dilakukan pemotongan variabel *StimulusCode* agar bisa dikonversi menjadi kelas *groundtruth*. Proses pemotongan pada masing-masing data adalah sebagai berikut.

- Data BCI Competition 2 dataset IIB (Dataset 2)

Dataset 2 terdiri dari 11 file data *training* dan 8 file data *testing*. Panjang sinyal mentah yang ada pada masing-masing file berbeda, namun memiliki format yang sama yaitu *S x C*, dimana *S* merupakan Panjang sinyal mentah dan *C* adalah banyaknya *channel*. Langkah pertama yang dilakukan adalah mencari pada data ke berapa saja variabel flashing bernilai 1 atau dalam kondisi terjadi stimulus. Setelah didapatkan 24 data indeks *flashing*, dilakukan pemotongan indeks menjadi *N x 24* data. Indeks yang sudah didapat dipakai sebagai acuan untuk memotong sinyal mentah. Kemudian sinyal mentah dipotong menjadi potongan sinyal sepanjang 160 data.

- Data BCI Competition 3 dataset II (Dataset 3)

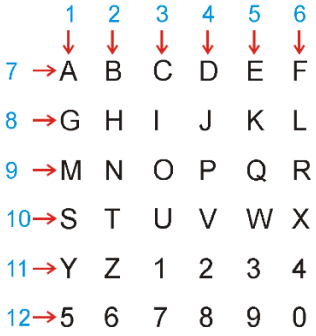
Data *BCI Competition* 3 dataset II hanya terdiri dari 2 file data *training* dan 2 file date *testing*. Baik Panjang sinyal mentah maupun format data masing-masing file sudah sama. Data pada data *BCI Competition* 3 dataset 2 memiliki format *T x S x C*, dimana *T* menandakan jumlah *trial*, S menandakan jumlah *samples*, dan *C* merupakan jumlah *channel.* Langkah pertama yang dilakukan sedikit berbeda dengan pemotongan sinyal di *BCI Competition* 2

dataset IIB, yaitu mencari pada setiap *trial* lokasi data variabel flashing bernilai 1 atau dalam kondisi terjadi stimulus. Dari hasil ini didapatkan *T x (24*D)* data. *D* merupakan jumlah sinyal pada masing-masing *trial*.

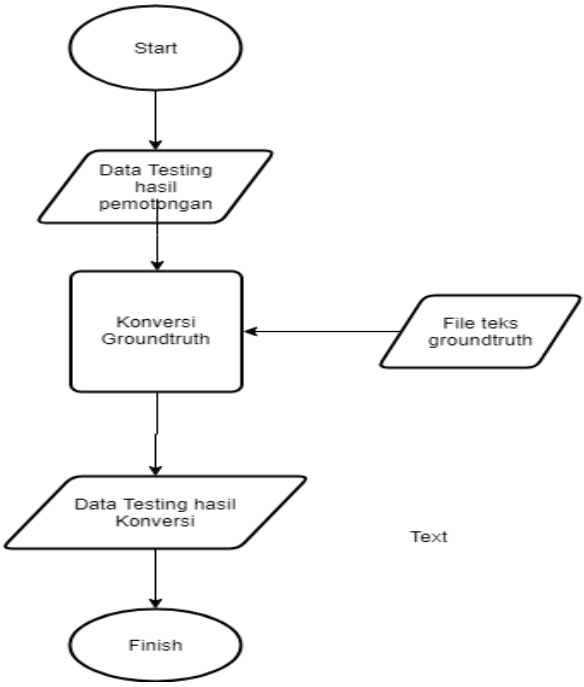
Kemudian data diubah menjadi *T x D x 24* data. Indeks yang sudah didapat dipakai sebagai acuan untuk memotong sinyal mentah. Kemudian sinyal mentah dipotong menjadi potongan sinyal sepanjang 160 data.

3.2.3.2 Konversi Groundtruth data testing

Pada data *testing*, *groundtruth* kelas dari data *testing* masih berupa huruf yang disimpan di file berekstensi .TXT, sehingga file tersebut perlu diubah menjadi kelas P300 atau non-P300. Langkah pertama yang dilakukan adalah membaca file *groundtruth* kemudian menyimpan huruf yang ada didalam file ke sebuah variabel, kemudian huruf yang ada di variabel tersebut dicari lokasi kolom dan barisnya. Penjelasan tentang kolom dan baris dari setiap huruf dapat dilihat pada gambar 3.4. Apabila variabel *StimulusCode* bernilai 6 maka sinyal tersebut merupakan huruf ‘F’. Apabila *groundtruth* yang terbaca merupakan huruf ‘F’ juga maka sinyal tersebut merupakan sinyal P300 dan nilai variabel kelas nya 1. Sebaliknya apabila *groundtruth* yang terbaca bukan huruf ‘F’ maka sinyal tersebut merupakan sinyal non-P300 dan nilai variabel kelas nya 0. Setelah itu kolom dan baris dari setiap huruf dicocokkan dengan variabel *StimulusCode* yang sudah ada pada data *testing*. Apabila variabel *StimulusCode* memiliki nilai yang sama dengan nilai kolom atau baris tersebut, maka variabel kelas diberi nilai 1, sebaliknya apabila nilainya tidak sama variabel kelas diberi nilai 0. Diagram Alir dari proses ini akan dijelaskan pada gambar 3.5



Gambar 3.4 Ilustrasi nilai dari variabel StimulusCode untuk intensifikasi baris / kolom yang berbeda.



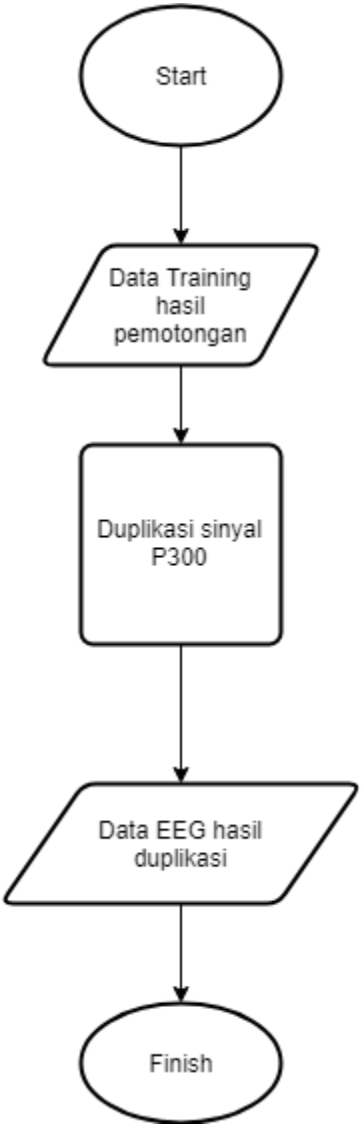
Gambar 3.5 Diagram Alir Konversi Groundtruth data testing

3.2.3.3 Band-pass filter

Pada tahap ini dilakukan pemrosesan pada sinyal EEG mentah yang didapat untuk mengurangi *noise* yang diakibatkan adanya inferensi antar sinyal yang ditangkap oleh masing-masing elektroda. Metode yang digunakan pada *noise filtering* adalah *bandpass filtering*. Data sinyal EEG yang didapat selalu tidak lepas dari temporal *noise*, yang bersifat kontinu terhadap waktu. *Temporal noise* yang dimaksud dalam bagian ini adalah adanya frekuensi-frekuensi dalam rentang tertentu yang tidak berhubungan dengan proses imajinasi gerakan motorik yang menjadi fokus penelitian ini, sehingga perlu dilakukan *filtering*. Untuk memaksimalkan pengambilan nilai informasi yang diperlukan, *bandpass filtering* diaplikasikan dalam pada *range* frekuensi antara 0.1-20 Hz. Data hasil pemotongan sinyal untuk setiap channelnya dilakukan *bandpass filtering*, dimana gelombang dengan frekuensi antara 0.1 Hz dan 20 Hz akan dilewatkan, sedangkan frekuensi diluar itu akan terfilter. Proses ini akan menghasilkan sinyal keluaran yang memiliki frekuensi lebih rendah dibanding sebelumnya, dikarenakan frekuensi sinyal awal cukup besar. Hasil keluaran yang diperoleh ini berupa matriks dengan sinyal yang telah terfilter dan lebih spesifik terhadap nilai informasi sesuai dengan kelas sinyal tersebut.

3.2.3.4 Duplikasi Sinyal P300

Data inputan pada tugas akhir ini merupakan data yang *imbalanced*, dan apabila dilakukan klasifikasi model yang dihasilkan akan cenderung mengklasifikasikan masukan data *test* sebagai kelas yang memiliki jumlah yang banyak. Oleh karena itu, dilakukan *oversampling* dengan cara duplikasi kelas yang sedikit. Karena perbandingan kelas yang ada 1:5 maka dilakukan duplikasi sebanyak 4 kali. Diagram alir dari proses duplikasi dapat dilihat pada gambar 3.6



Gambar 3.6 Diagram Alir Duplikasi Sinyal P300

3.2.3.5 Min-max normalisation

Data hasil bandpass filtering kemudian dinormalisasi, namun khusus untuk data *BCI Competition* 2 dataset IIB terlebih dahulu dilakukan penggabungan data sehingga hanya terdapat 1 file data *training* dan 1 file data *testing*, sehingga total terdapat 3 file data *training* dan 3 file data *testing* yang akan dinormalisasi.

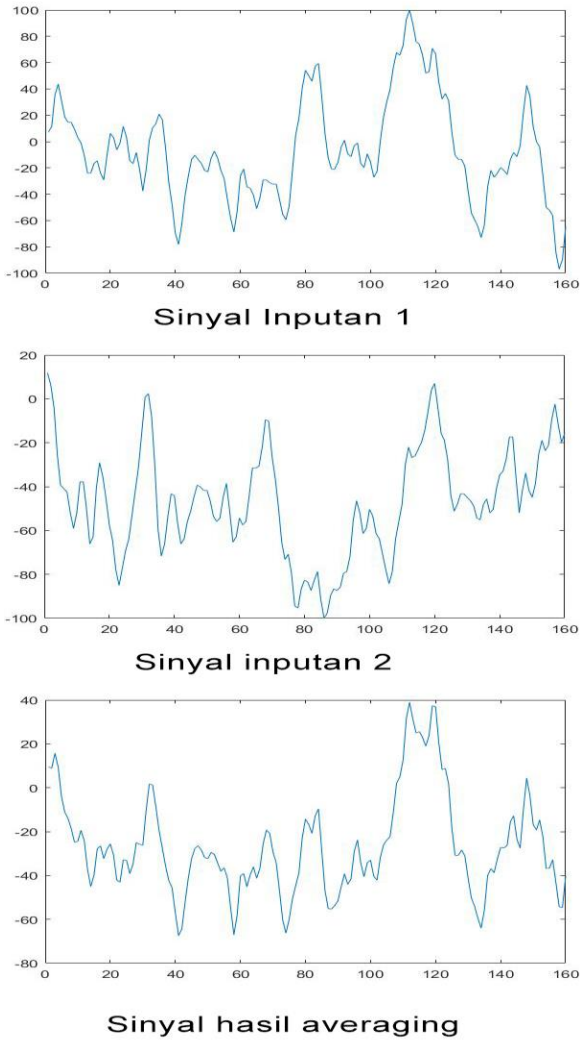
Proses normalisasi yang digunakan pada tugas akhir ini adalah min-max normalization. Dikarenakan data pada tugas akhir ini berupa data sinyal, maka hasil normalisasi berkisar antara -1 sampai 1. Kemudian hasil normalisasi dikali 100 agar rentang normalisasi menjadi -100 sampai 100.

3.2.3.6 Signal Averaging

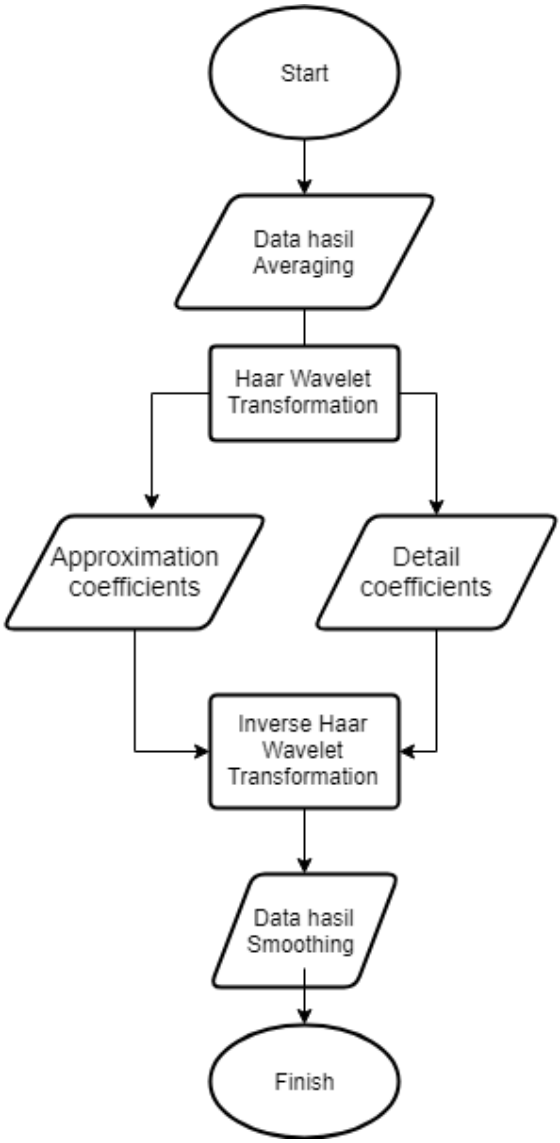
Setelah dilakukan normalisasi, dilakukan *signal averaging*. Pada tugas akhir ini dilakukan 3 jenis *signal averaging*, yaitu 1 signal, 2 signal dan 3 signal. Sehingga dihasilkan matriks sinyal yang berukuran 1 atau 1/2 atau 1/3 dari data hasil normalisasi. Contoh signal averaging dapat dilihat pada gambar 3.7

3.2.4 Penghalusan data dengan Haar Wavelet Transformation

Tahap penghalusan data pada tugas akhir ini dilakukan dengan metode *Haar Wavelet Transformation(HWT)*. HWT diimplementasikan untuk memperhalus inputan sinyal yang kemudian diklasifikasi pada tahap selanjutnya. Data inputan pada setiap channelnya ditransformasi menggunakan *haar 1-D wavelet transform* dengan parameter default, kemudian didapatkan koefisien aproksimasi dan detil koefisiennya. Kemudian koefisien aproksimasi dan detil koefisien digunakan sebagai inputan dari invers HWT. Level dari invers HWT yang digunakan pada tugas akhir ini ada 6 level scenario uji coba yaitu mulai dari level 0 sampai level 5. Keluaran dari invers HWT merupakan data masukan yang lebih halus. Gambar 3.8 menjelaskan proses penghalusan data dengan *Haar Wavelet Transformation.*



Gambar 3.7 Contoh proses averaging 2 sinyal



Gambar 3.8 Diagram Alir proses Penghalusan data

3.2.5 Klasifikasi dengan Batch Normalisation Neural Network

Setelah data dihaluskan menggunakan HWT dan *inverse* HWT, data siap untuk diklasifikasi. Pada tugas akhir ini digunakan metode klasifikasi Batch Normalisation Neural Network(BN3). BN3 yang digunakan terdiri dari 6 jaringan. Pada jaringan pertama atau L0 terdiri dari lapisan *input* dengan *Batch Normalisation*. Sedangkan pada lapisan kedua atau L1 merupakan lapisan konvolusi yang digunakan untuk ekstraksi fitur spasial. L2 adalah lapisan konvolusi dan subsampling 1-D untuk ekstraksi fitur. Pada tugas akhir ini digunakan Rectified Linear Unit (ReLU) sebagai fungsi aktivasi pada layer L2. Perhitungan efisien ReLU(x) = max (x, 0)[29] memungkinkan konvergensi lebih cepat selama tahap pelatihan. L3 dan L4 adalah lapisan yang terhubung sepenuhnya dengan menggunakan *dropout*. L 5 adalah lapisan keluaran dengan unit sigmoid tunggal, yang menghasilkan hasil klasifikasi biner. Topologi jaringan yang dipakai pada tugas akhir ini dapat dilihat pada gambar 3.9

Penjelasan masing-masing lapisan akan dijelaskan sebagai berikut.

- L0

  Batch Normalisation diterapkan untuk *batch input*.

- L1

  L1 berfungsi sebagai lapisan penyaringan spasial,dimana kernel konvolusi berukuran 64 × 1.Jumlah kernel konvolusi pada tugas akhir ini ada 3 skenario yaitu 1 kernel, 8 kernel, dan 16 kernel. Keluaran dari lapisan ini adalah vektor $T \; x \; S$, di mana T = 160 adalah panjang jendela waktu dan S adalah jumlah kernel konvolusinya.

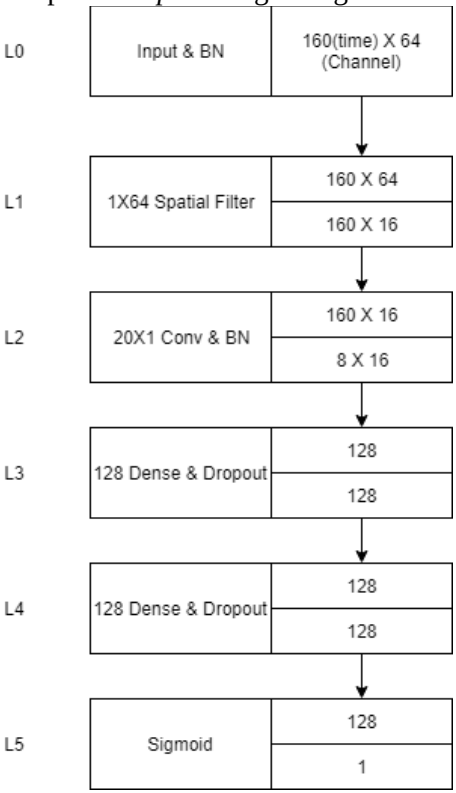
- L2

  Batch Normalisation dilakukan sebelum aktivasi untuk menghindari pergeseran distribusi, yang menyebabkan saturasi dan perlambatan pembelajaran. Kernel konvolusi pada layer ini memiliki ukuran 20 × 1 .Dengan demikian, Penulis menetapkan

tingkat *downsampling* menjadi 20 karena pertimbangan berikut. Jendela waktu asli setiap sampel adalah 160, mewakili sinyal 667 ms. Kemudian filter temporal diperkecil sebanyak 8 kali menjadi 667/8 = 83. 375 ms[29]. Pengaturan ini agar konvolusi temporal L2 tidak tumpang tindih

- L3

  Pada Layer L3 terdapat lapisan *fullyconnectedLayer* dan lapisan *dropout.* Ukuran lapisan *fullyconnectedLayer* dan probabilitas dari lapisan *dropout* bergantung dari skenario ujicoba.



Gambar 3.9 Topologi jaringan BN3

- L4

  L4 merupakan duplikasi dari L3.

- L5

Pada lapisan terakhir ini digunakan terdapat lapisan *fullyconnectedLayer* berukuran 2, *softmaxLayer* dan *classificationLayer*. Ketiga lapisan ini digunakan untuk memprediksi kelas sinyal inputan. Keluaran dari lapisan ini adalah nilai 1 untuk kelas P300 dan 0 untuk kelas non-P300.

3.2.6 Metode Evaluasi Kinerja

Pengukuran performa untuk evaluasi yang digunakan pada tugas akhir ini adalah akurasi, *sensitivity*, dan *specificity*. Pengujian evaluasi *classifier* pada tugas akhir ini menggunakan *simple split*, dimana data yang didapat sudah terbagi menjadi *training data* dan *testing data*, Hasil evaluasi menunjukkan tingkat kesesuaian prediksi sinyal P300, berdasarkan masukan dataset EEG yang ada.

BAB IV
IMPLEMENTASI

Bab ini membahas tentang implementasi dari sistem klasifikasi sinyal EEG dengan *Batch Normalisation Neural Network.*. Pembahasan implementasi ini meliputi deskripsi lingkungan tahap implementasi sistem ini dilakukan, proses-proses pada tahap implementasi yang dikerjakan, beserta penjelasan fungsi-fungsinya dalam bentuk kode sumber.

4.1 Lingkungan Implementasi

Subbab ini akan menjelaskan mengenai lingkungan implementasi perangkat lunak yang akan dibangun. Spesifikasi perangkat keras yang digunakan dalam desain dan implementasi perangkat lunak adalah prosesor berjenis Intel(R) Core(TM) i7-4720HQ CPU @ 2.60GHz (8 CPUs), ~2.6GHz, dengan kapasitas memori (RAM) sebesar 8.192 GB.

Sistem operasi yang digunakan adalah Microsoft Windows 10 Home Single Language 64-bit, sedangkan perangkat lunak yang digunakan dalam proses implementasi adalah MATLAB R2018a. Perangkat lunak MATLAB yang digunakan tidak boleh memiliki versi yang lebih rendah dari R2018a, karena parameter dari pembentukan model baru terdapat pada MATLAB R2018a.

4.2 Implementasi Program

Dalam subbab ini akan ditampilkan potongan kode program yang digunakan dalam klasifikasi sinyal EEG dengan *Batch Normalisation Neural Network.*. Potongan kode program yang akan ditampilkan dalam sub bab ini adalah potongan kode program untuk tahap *preprocessing*, penghalusan data, dan klasifikasi.

4.2.1 Implementasi Preprocessing

Preprocessing yang diimplementasikan pada tahap ini dibagi menjadi 5 bagian utama, penjelasan bagian-bagian pada tahap ini akan dijelaskan sebagai berikut

4.2.1.1 Implementasi Pemotongan Sinyal

Pada tugas akhir ini dilakukan pemotongan sinyal 0-667ms setelah stimulus. Stimulus yang dimaksud dapat dilihat pada variabel *flashing*, dimana ketika *flashing* bernilai 1 maka terjadi stimulus, sedangkan 0 tidak terjadi stimulus. Stimulus memiliki panjang 100ms atau 24 data. Hasil dari pemotongan data berupa matriks berukuran *N x 160 x C*, dimana *N* merupakan jumlah sinyal yang akan diklasifikasi dan *C* adalah banyaknya channel. Untuk data *training* selain sinyal yang dipotong, dilakukan juga pemotongan variabel kelas sinyal tersebut, sedangkan pada data *testing* selain pemotongan sinyal dilakukan pemotongan variabel *StimulusCode* agar bisa dikonversi menjadi kelas *groundtruth*.

Kode sumber 4.1 merupakan kode yang dipakai untuk memotong sinyal pada data *testing* dataset 2 pada scenario ini. Kode sumber pada baris ketiga di keempat kode sumber ini digunakan untuk mencari index dari variabel *flashing* yang memiliki nilai lebih dari 0. Keluaran dari kode ini adalah variabel *indexflashing* yang kemudian dikonversi menjadi *N x 24* menggunakan kode sumber pada baris 6 sampai 14. Kemudian baris 16 sampai 31 digunakan untuk memotong sinyal 0-667ms setelah stimulus dan stimulusCode berdasarkan indeks flashing yang sudah disimpan pada variabel index

Untuk data *training* dataset 2 terdapat perubahan pada baris 7 seperti ditunjukkan pada kode sumber 4.2 berikut. Perubahan disebabkan karena pada data testing variabel groundtruth tidak berada pada satu file seperti pada data training.

Kode sumber 4.3 merupakan kode yang dipakai untuk pemotongan sinyal pada data *testing* dataset 3. Seperti pada data *testing* data 2 baris ketiga digunakan untuk mencari indeks dari nilai flashing yang lebih besar dari 0. Baris 7 sampai 27 digunakan untuk merubah indeks nilai flashing menjadi tiga dimensi

mengikuti dimensi dari sinyal, sehingga dapat dilakukan pemotongan sinyal sesuai indexflashing tersebut. Baris 29 sampai 49 digunakan untuk memotong sinyal dan *stimulusCode* berdasarkan indeks pada *indexflashing*.

Pada data *training* terdapat perubahan kode di baris 8 seperti terlihat pada kode sumber 4.4. Perubahan ini dikarenakan *trial* pada data *testing* adalah 100 *trial* sedangkan pada data *training* hanya terdapat 85 *trial*. Selain itu, terdapat perubahan pada baris 39 yang dapat dilihat pada kode sumber 4.5. Seperti pada dataset 2, perubahan ini disebabkan karena pada data *testing* variabel *groundtruth* tidak berada pada satu file seperti pada data *training*.

```
1.  % mencari index flashing bernilai 1
2.  indexflashing = find(Flashing(:,:) >0); % Find left index
3.  x=0; y=0;
4.   for(i=1:size(indexflashing,1))
5.      if(mod(i,24)==1)
6.           y=y+1;  x=1;
7.      else   x=x+1; end
8.           index(y,x)=indexflashing(i); end
9.  % memotong signal flashing + respons
10. z=size(index,1);  sig=zeros(z,160,64);  class=zeros(z,1);
    x=1; y=1;
11.      for(j=1:size(index,1))
12.          a=1;          for(i=1:184)              if(i>24)
13.                  sig(j,a,:)=signal(index(j,1)+i-1,:);
14.                  class(j,1)=StimulusCode(index(j,1),1);
15.                  a=a+1; end end end
```

Kode Sumber 4.1 Pemotongan sinyal pada data testing data 2

```
1.  class(j,1)=StimulusType(inde(j,1),1);
```

Kode Sumber 4.2 Perubahan pemotongan sinyal pada data training data 2

1.	% mencari index flashing bernilai 1
2.	[indexflashing1,indexflashing2] = find(Flashing(:,:) >0); % Find left index
3.	x=0; y=0; z=1;
4.	for(i=1:size(indexflashing1,1))
5.	if(mod(i,100)==1)
6.	y=y+1; x=1; else x=x+1; end
7.	ind(x,y)=indexflashing2(i); end
8.	x=1; for(x=1:size(ind,1)) y=0;
9.	for(i=1:size(ind,2)) if(mod(i,24)==1)
10.	z=1; y=y+1; end
11.	index(x,y,z)=ind(x,i); z=z+1; end end
12.	% memotong signal flashing + respons
13.	sig=zeros(size(index,1),size(index,2),160,64);
14.	Clas=zeros(size(index,1),size(index,2));
15.	for(i=1:size(Flashing,1))
16.	x=1;y=1;
17.	for(j=1:size(Flashing,2))
18.	if(j==index(i,x,1)&& x<=size(index,2))
19.	a=1; for(y=1:184) if(y>24)
20.	sig(i,x,a,:)=Signal(i,j+y-1,:);
21.	clas(i,x)=StimulusCode(i,j); a=a+1; end end
22.	x=x+1; if(x>180) x=1; end end end end

Kode Sumber 4.3 Pemotongan sinyal pada data testing data 3

1.	if(mod(i,85)==1)

Kode Sumber 4.4 Perubahan pemotongan sinyal pada data training data 3 baris 8

1.	clas(i,x)=StimulusType(i,j);

Kode Sumber 4.5 Perubahan pemotongan sinyal pada data training data 3 baris 39

4.2.1.2 Implementasi Konversi Groundtruth

Konversi *groundtruth* hanya terjadi pada data *testing*. Untuk melakukan konversi pertama-tama file yang berisi *groundtruth* huruf dibuka. Kode sumber 4.6 baris 1 membuka file tersebut kemudian menyimpannya ke sebuah variabel menggunakan kode baris 2 sampai 5, kemudian masing masing huruf diubah menjadi variabel *B* yang merupakan lokasi baris dan kolom dari huruf tersebut menggunakan kode sumber baris 7 sampai 19. Letak baris dan kolom dari masing-masing variabel dapat dilihat pada gambar 3.4. Pembacaan *groundtruth testing* memiliki kode yang sama baik dataset 3 maupun dataset 2.

```
1.  fileID = fopen(‘grountrouth test B.txt','r');
2.  formatSpec = '%s';
3.  sizeA = [1000 1];
4.  A = fscanf(fileID,formatSpec,sizeA);
5.  fclose(fileID);
6.  B=zeros(size(A,1),2);
7.  for(i=1:size(A,1))
8.    if(strcmp(A(i,1),'A')==1)
9.      B(i,1)=1;
10.     B(i,2)=7;
11.   elseif(strcmp(A(i,1),'B')==1)
12.     B(i,1)=2;
13.     B(i,2)=7;
14.   …
15.   elseif(strcmp(A(i,1),'_')==1)
16.     B(i,1)=6;
17.     B(i,2)=12;
18.   end
19. end
```

Kode Sumber 4.6 Pembacaan groundtruth testing

Kemudian pada dataset 2 dilakukan konversi dengan mencocokkan variabel *B* dengan variabel *StimulusCode*, apabila memiliki nilai yang sama, maka sinyal tersebut merupakan sinyal P300 atau kelas 1, sedangkan apabila nilainya tidak sama, maka sinyal tersebut merupakan sinyal non-P300 atau kelas 0. Kode sumber 4.7 merupakan kode sumber yang dipakai untuk mengkonversi groundtruth pada dataset 2, sedangkan kode sumber 4.8 digunakan untuk mengkonversi groundtruth pada dataset 3.

```
1.  %   membuat groundtruth testing
2.  for(i=1:size(class,1))
3.          if(class(i,1)==B(j,1) || class(i,1)==B(j,2))
4.             class(i,1)=1;
5.          else
6.              class(i,1)=0;
7.          end
8.  End
```

Kode Sumber 4.7 Konversi groundtruth testing dataset 2

```
1.   %   membuat groundtruth testing
2.   for(i=1:size(clas,1))
3.      for(j=1:size(clas,2))
4.         if(clas(i,j)==B(i,1) || clas(i,j)==B(i,2))
5.            clas(i,j)=1;
6.         Else
7.             clas(i,j)=0;
8.         End
9.      End
10.  End
```

Kode Sumber 4.8 Konversi groundtruth testing dataset 3

4.2.1.3 Implementasi Bandpass Filter

Bandpass filter yang dipakai pada tugas akhir ini adalah Bandpass Butterworth Filter, dengan *Filter order* 8 dan *cut-off frequency* 0.1-20 Hz. Kode sumber 4.9 menjelaskan penggunaan filter ini. Pada baris 2 dibuat Bandpass Filter menggunakan *designfilt*, dengan parameter *bandpassiir* untuk menyatakan bahwa filter yang akan dipakai adalah bandpass, parameter *filterorder* menyatakan *level filter order* yang akan dipakai, sedangkan *HalfPowerFrequency1* dan *HalfPowerFrequency2* menyatakan batas bawah dan batas atas dari *cut off* frekuensinya.

```
1.  %   Bandpass Butterworth filter
2.  d = designfilt('bandpassiir','FilterOrder',8, ...
    'HalfPowerFrequency1',0.1,'HalfPowerFrequency2',20, ...
    'SampleRate',240);
3.  sign=filter(d,sig);
```

Kode Sumber 4.9 Bandpass Filter

4.2.1.4 Implementasi Duplikasi Sinyal P300

Pada data *training*, dilakukan duplikasi sinyal dikarenakan data yang ada tidak seimbang. Namun sebelum itu ada beberapa proses yang harus dilakukan terlebih dahulu sebelumnya. Setelah duplikasi juga masih perlu dilakukan proses penggabungan data dan penggabungan file untuk dataset 2. Penjelasan masing-masing proses akan dijelaskan sebagai berikut.

- Mengubah hasil pemotongan sinyal pada dataset 3
  Perubahan hasil pemotongan diperlukan karena keluaran dari pemotongan merupakan matriks 4 dimensi. Oleh sebab itu diperlukan perubahan dimensi matriks dari 4 dimensi ke 3 dimensi. Kode sumber yang diperlukan untuk mengubah dimensi dapat dilihat pada kode sumber 4.10
- Membagi hasil pemotongan berdasarkan kelas
  Untuk menduplikasi data yang tidak seimbang maka data perlu dikelompokkan sesuai kelas masing-masing. Kode untuk pengelompokan masing-masing data berserta kelasnya dapat dilihat pada kode sumber 4.11

- Duplikasi sinyal
  Setelah data dikelompokkan berdasarkan kelas masing-masing, maka duplikasi dapat dilakukan. Proses penduplikasian menggunakan *repmat* dimana kelas P300 yang jumlahnya lebih sedikit diduplikasi sehingga total datanya sama dengan kelas non-P300. Kode untuk duplikasi dapat dilihat pada kode sumber 4.12
- Penggabungan sinyal
  Setelah data seimbang, maka data per kelas perlu digabung kembali. Kode untuk penggabungan dapat dilihat pada kode sumber 4.13
- Penggabungan file data 2
  Khusus untuk data 2 baik data *training* maupun data *testing* perlu dilakukan penggabungan file menjadi 1 file data *training* dan 1 file data *testing*. Karena data *training* maupun data *testing* awalnya terpisah menjadi 11 file data *training* dan 8 file data *testing*. Kode untuk penggabungan dapat dilihat pada kode sumber 4.14

```
1.  % mengubah hasil pemotongan 4 dimensi menjadi 3 dimensi dengan
2.  % menggabungkan pemotongan samples tiap character epoch
3.   x=1;
4.   signal=zeros((size(sign,1)*size(sign,2)),160,64);
5.   class=zeros((size(sign,1)*size(sign,2)),1);
6.   for(i=1:size(sig,1))
7.      for(j=1:size(sig,2))
8.       signal(x,:,:)=sig(i,j,:,:);
9.       class(x,1)=clas(i,j);
10.       x=x+1;
11.     End
12.   End
```

Kode Sumber 4.10 Perubahan dimensi dataset 3

```
1.  % membagi hasil pemotongan berdasarkan groundtruth P300 atau bukan
2.  truth=zeros(size(sign,1)/6,160,64);
3.  truth_class=zeros(size(sign,1)/6,1);
4.  false=zeros((size(sign,1)/6)*5,160,64);
5.  false_class=zeros((size(sign,1)/6)*5,1);
6.  y=1;z=1;
7.  for(x=1:size(class,1))
8.      if(class(x,1)==1)
9.          truth(y,:,:)=sign(x,:,:);
10.         truth_class(y,1)=class(x,:);
11.         y=y+1;
12.     else
13.         false(z,:,:)=sign(x,:,:);
14.         false_class(z,:)=class(x,:);
15.         z=z+1;
16.     end
17. end
```

Kode Sumber 4.11 Pengelompokan data berdasarkan kelas

```
1.  % menggandakan sinyal P300 sebanyak 4 kali
2.  truth_rep=repmat(truth,5,1);
3.  truth_rep_class=repmat(truth_class,5,1);
```

Kode Sumber 4.12 Duplikasi sinyal P300

```
1.  % menggabungkan hasil penggandaan P300 dan sinyal non P300
2.  input=cat(1,truth_rep,false);
3.  input_class=cat(1,truth_rep_class,false_class);
```

Kode Sumber 4.13 Penggabungan data

```
1.  % menggabungkan hasil pemrosesan dataset2
2.  % penggabungan menjadi dataset training dan dataset testing
3.  load('db221208T.mat');
4.  in=input;
5.  in_class=input_class;
6.   load('db221207T.mat');
7.   inp=cat(1,input,in);
8.   inp_class =cat(1,input_class, in_class);
9.  …
10.  load('db221201T.mat');
11.  inp=cat(1,input,in);
12.  inp_class=cat(1,input_class,in_class);
13.  input=inp;
14.  Input_class=inp_class;
```

Kode Sumber 4.14 Penggabungan file dataset 2

4.2.1.5 Implementasi Min-max normalisation

Proses selanjutnya adalah melakukan min-max normalisation. Kode sumber 4.15 merupakan kode yang dipakai untuk mengimplementasikan normalisasinya. Pada baris 7 sampai 14 dilakukan pencarian nilai terbesar atau nilai terkecil, sedangkan pada baris 15 sampai 17 dilakukan normalisasi dengan skala -100 sampai 100.

```
1.  varSize1 =size(input,1);
2.  varSize2 =size(input,2);
3.  varSize3 =size(input,3);
4.  for(a=1:varSize1)
5.     for(b=1:varSize3)
6.        x=max(input(a,:,b)); y=abs(min(input(a,:,b)));
7.        if(y>x)   p=y;  else p=x; end
8.         for(c=1:varSize2)
9.           input(a,c,b)=(100*input(a,c,b))/p;  end end end
```

Kode Sumber 4.15 Min-Max Normalisation

4.2.1.6 Implementasi Signal Averaging

Setelah dinormalisasi, data sinyal kemudian dilakukan sebuah proses yang disebut dengan *Signal Averaging*. Kode sumber 4.16 merupakan kode yang dipakai untuk melakukan *averaging* 3 sinyal. Skenario lain yang digunakan pada tugas akhir ini yaitu *averaging* 2 sinyal, dan averaging 1 sinyal. Kode sumber 4.16 perlu diubah pada baris 8 sampai 23 seperti terlihat pada kode sumber 4.17

```
1.  varSize1 =size(input,1);
2.  varSize2 =size(input,2);
3.  varSize3 =size(input,3);
4.
5.  a=varSize1/3;
6.  input1=zeros(a,varSize2,varSize3);
7.  input_class1=zeros(a,1);
8.  x=1;
9.
10. for(s=1:varSize1)
11.     input1(x,:,:)=input1(x,:,:)+input(s,:,:);
12.     input_class1(x,1)=input_class1(x,1)+input_class(s,1);
13.     if(mod(s,3)==0)
14.     x=x+1;
15.     end
16. end
17.
18. for(s=1:a)
19.     input1(s,:,:)=input1(s,:,:)/3;
20.     input_class1(s,1)=input_class1(s,1)/3;
21. end
22.
23. input=input1;
24. input_class=input_class1;
```

Kode Sumber 4.16 Signal Averaging 3 sinyal

```
1.  a=varSize1/2;
2.  input1=zeros(a,varSize2,varSize3);
3.  input_class1=zeros(a,1);
4.  x=1;
5.  for(s=1:varSize1)
6.      input1(x,:,:)=input1(x,:,:)+input(s,:,:);
7.      input_class1(x,1)=input_class1(x,1)+input_class(s,1);
8.      if(mod(s,2)==0)
9.      x=x+1;
10.     end
11. end
12.
13. for(s=1:a)
14.     input1(s,:,:)=input1(s,:,:)/2;
15.     input_class1(s,1)=input_class1(s,1)/2;
16. end
```

Kode Sumber 4.17 Perubahan untuk signal averaging 2 sinyal

4.2.2 Implementasi Penghalusan data dengan Haar Wavelet Transformation

Proses selanjutnya adalah proses penghalusan data menggunakan HWT dan *inverse* HWT. Kode sumber 4.18 merupakan kode yang dipakai untuk menghaluskan data. Terdapat 6 skenario penghalusan data yang digunakan pada tugas akhir ini, yaitu *inverse* HWT level 0 sampai level 5. Perubahan yang dilakukan pada masing-masing skenario ada pada baris ke 9, dimana parameter *ihaart* diganti sesuai level pada scenario uji coba.

```
1.  varSize1 =size(input,1);
2.  varSize3 =size(input,3);
3.  inpd=zeros(varSize1,160,64);
4.  for(a=1:varSize1)   for(b=1:varSize3)
5.          [s,d]=haart(input(a,:,b));
6.          inpd(a,:,b)=ihaart(s,d,1); end end input=inpd;
```

Kode Sumber 4.18 Penghalusan data Haar Wavelet Transformation level 1

4.2.3 Implementasi Klasifikasi dengan Batch Normalisation Neural Network

Setelah dilakukan penghalusan, data siap untuk diklasifikasi. Langkah pertama yang dilakukan adalah membangun model berdasarkan data *training*. Model yang dibangun akan digunakan untuk mengklasifikasikan data *testing*. Pada tugas akhir ini terdapat 3 jenis model data *training*, yaitu model untuk data *training* dataset 2, model untuk data *training* dataset 3 subjek A (dataset 3A), dan model untuk data *training* dataset 3 subjek B (dataset 3B). Alasan pembuatan 3 jenis model adalah keterbatasan *software* dan *hardware* yang dipakai pada tugas akhir ini. Kode sumber 4.19 adalah kode yang dipakai untuk membangun model BN3 dari data training, sedangkan kode sumber 4.20 adalah kode yang dipakai untuk mengklasifikasikan data testing berdasarkan model yang dibangun. Terdapat beberapa skenario uji coba yang akan dijelaskan sebagai berikut.

- Jumlah Epoch
  Pada tugas akhir ini terdapat 4 scenario uji coba untuk parameter jumlah *epoch*, yaitu 100 *epoch*, 400 *epoch*, 700 *epoch*, dan 1000 *epoch*. Di kode sumber 4.19 parameter MaxEpoch pada baris 41 dapat diubah sesuai dengan scenario uji coba.
- Dropout Probability
  Terdapat 5 scenario uji coba untuk nilai *Dropout Probability* yaitu 0, 0.2, 0.4, 0.6, dan 0.8. Di kode sumber 4.19 nilai *Dropout Probability* pada baris 32 dan 34 dapat diubah sesuai dengan scenario uji coba.
- Fully Connected Layer
  Terdapat 4 scenario uji coba untuk jumlah *Fully Connected Layer* pada L3 dan L4 yaitu 64, 128, 192, dan 256. Di kode sumber 4.19 jumlah Fully Connected Layer pada baris 31 dan 33 dapat diubah sesuai dengan scenario uji coba.

- Convolution Filter
  Tedapat 3 scenario uji coba untuk jumlah *Convolution layer* pada L1 yaitu 1 *filter*, 8 *filter*, dan 16 *filter*. Di kode sumber 4.19 jumlah *Convolution filter* pada baris 21 dapat diubah sesuai dengan scenario uji coba.

```
1.  varSize1 =size(inp,1); varSize2 =size(inp,2);
2.  varSize3 =size(inp,3); x=0;
3.  t=zeros (varSize2,varSize3,1,varSize1);
4.  for(a=1:size(t,1))
5.     for(b=1:size(t,2))
6.        for(c=1:size(t,3))
7.           for(d=1:size(t,4))
8.              t(a,b,c,d)=inp(d,a,b); end end end end
9.   inpst1=reshape(inpst,[1,varSize1]);
10.  B = categorical(inpst1);
11. conv1 = convolution2dLayer([1 64],16);
12. conv2 = convolution2dLayer([20 1],1,'Stride',[20 1]);
13.  layers = [
14.     imageInputLayer([varSize2 varSize3 1]);
15.     batchNormalizationLayer;
16.     conv1;     conv2;
17.     batchNormalizationLayer;     reluLayer();
18.     fullyConnectedLayer(128);     dropoutLayer(0.8);
19.     fullyConnectedLayer(128);     dropoutLayer(0.8);
20.     fullyConnectedLayer(2);     softmaxLayer();
21.     classificationLayer()];
22. opts = trainingOptions('adam', ...
23.    'Plots','training-progress', ...
24.    'InitialLearnRate', 0.00001, ...
25.    'MaxEpochs',1000, ...
26.    'GradientDecayFactor',0.9,...
27.    'MiniBatchSize',64);
28. [net, info] = trainNetwork(t,B, layers, opts);
```

Kode Sumber 4.19 Pembuatan model klasifikasi

```
1.  varSize1 =size(input,1);
2.  varSize2 =size(input,2);
3.  varSize3 =size(input,3);
4.   inp=zeros (varSize2,varSize3,1,varSize1);
5.
6.  for(a=1:size(te,1))
7.     for(b=1:size(te,2))
8.        for(c=1:size(te,3))
9.           for(d=1:size(te,4))
10.             inp(a,b,c,d)=input(d,a,b);
11.          end
12.       end
13.    end
14. end
15. inp_class=reshape(input_class,[1,varSize1]);
16. B = categorical(inp_class);
17. [YPred,scores] = classify(net,inp);
18. Y=reshape(YPred,[1,varSize1]);
```

Kode Sumber 4.20 Pengklasifikasian data testing

4.2.4 Implementasi Perhitungan Evaluasi Kerja

Untuk mengevaluasi kinerja dari model yang dibuat, hasil klasifikasi perlu diubah menjadi *Confusion matrix* seperti pada kode sumber 4.21 baris 1. Dari *Confusion matrix* tersebut dapat dihitung akurasi, sensitivity, dan spesificitynya.

```
1.  [c, o]=confusionmat(B,Y);
2.  spesificity=c(1,1)/(c(1,1)+c(1,2));
3.  sensitivity=c(2,2)/(c(2,2)+c(2,1));
4.  acc=(c(1,1)+c(2,2))/(c(1,1)+c(1,2)+c(2,1)+c(2,2));
```

Kode Sumber 4.21 Perhitungan evaluasi kerja

[Halaman ini sengaja dikosongkan]

BAB V
UJI COBA DAN EVALUASI

Bab ini membahas tentang scenario uji coba dan evaluasi yang dilakukan pada klasifikasi sinyal EEG dengan metode utama *Batch Normalisation Neural Network* Hasil uji coba pada tahap ini akan dievaluasi dengan tujian memperoleh jawaban dari rumusan masalah yang telah dirumuskan di awal.

5.1 Lingkungan Uji Coba

Lingkungan uji coba dan evaluasi merupakan komputer tempat uji coba perangkat lunak. Spesifikasi perangkat keras yang digunakan untuk melakukan uji coba perangkat lunak terdiri dari prosesor berjenis Intel(R) Core(TM) i7-4720HQ CPU @ 2.60GHz (8 CPUs), ~2.6GHz, dengan kapasitas memori (RAM) sebesar 8.192 GB.

Sistem operasi yang digunakan adalah Microsoft Windows 10 Home Single Language 64-bit, sedangkan perangkat lunak yang digunakan adalah MATLAB R2018a. Perangkat lunak MATLAB yang digunakan tidak boleh memiliki versi yang lebih rendah dari R2018a, karena parameter dari pembentukan model baru terdapat pada MATLAB R2018a.

5.2 Data Uji Coba

Data yang digunakan untuk uji coba klasifikasi sinyal EEG dengan metode utama *Batch Normalisation Neural Network* ini diperoleh dari *BCI Competition* II dataset 2B dan *BCI Competition* III dataset 2. Sinyal yang tersedia *BCI Competition* II dataset 2B diambil dari hasil perekaman 64 *channel* elektroda. Matriks dataset mentah sinyal yang diperoleh berdimensi $S \times C$, dimana S menandakan jumlah *samples,* dan C merupakan jumlah *channel.* Terdapat total 12930 sinyal pada dataset, dengan 10950 sinyal merupakan kelas 0 dan 1980 sinyal kelas 1. Kelas 0 merepresentasikan sinyal non-P300, sedangkan kelas 1 merepresentasikan sinyal P300. Sedangkan data dari *BCI Competition* III dataset 2 terdiri memiliki sinyal yang diambil dari

hasil perekaman 64 *channel* elektroda. Matriks dataset mentah sinyal yang diperoleh berdimensi *T x S x C*, dimana *T* menandakan jumlah *trial*, S menandakan jumlah *samples*, dan *C* merupakan jumlah *channel*. Terdapat total 65750 sinyal pada dataset, dengan 55500 sinyal *merupakan* kelas 0 dan 10250 sinyal kelas 1. Kelas 0 merepresentasikan sinyal non-P300, sedangkan kelas 1 merepresentasikan sinyal P300. Dataset *BCI Competition* II dataset 2b selanjutnya akan disebut dataset 2, sedangkan Dataset *BCI Competition* III dataset 2 akan dibagi sesuai subjeknya dan akan disebut dataset 3A dan dataset 3B.

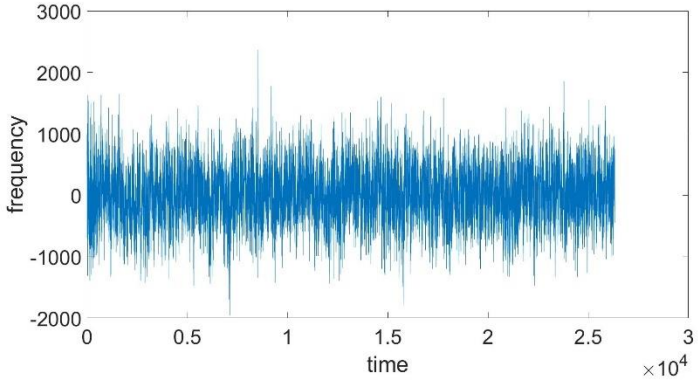
Pengujian evaluasi *classifier* pada tugas akhir ini menggunakan *simple split*, dimana data yang didapat sudah terbagi menjadi *training data* dan *testing data*. Pembentukan model dan pengujian klasifikasi menggunakan kode sumber dari arsitektur BN3 yang ada pada kode sumber 4.19 dan 4.20. Sedangkan parameter evaluasi pada tugas akhir ini menggunakan akurasi, *sensitivity*, dan *specificity*. Parameter dari scenario uji coba sebelumnya dipakai sebagai acuan untuk melakukan uji coba selanjutnya. Pemilihan parameter uji coba terbaik didasarkan pada nilai akurasi terbaik, dan nilai *sensitivity*, dan atau *specificity* sudah melebihi 60%.

5.3 Hasil uji coba setiap tahapan

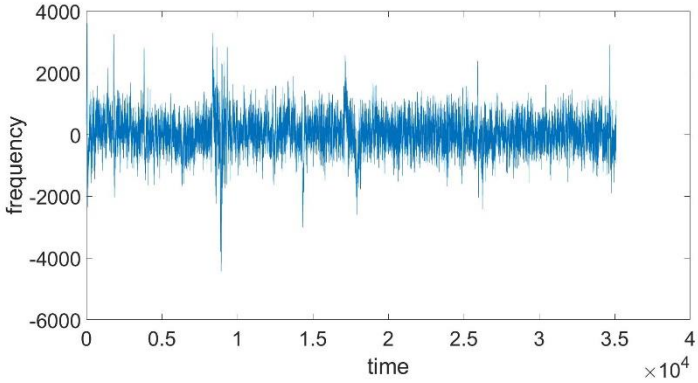
Gambar pada sub bab ini merupakan gambar sinyal hasil uji coba dengan sumbu x adalah waktu dan sumbu y adalah frekuensi dari sinyal tersebut.

5.3.1 Uji coba Pemotongan Sinyal

Pada sub bab ini akan dibahas mengenai hasil pemotongan sinyal pada data mentah. Gambar 5.1 sampai 5.3 merupakan gambar sampel data mentah pada *channel* 11. Data tersebut dipotong dan menghasilkan data sinyal seperti pada Gambar 5.4 sampai 5.6

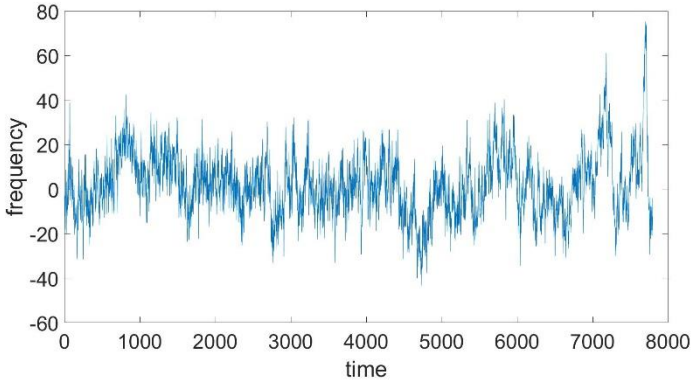


(a)

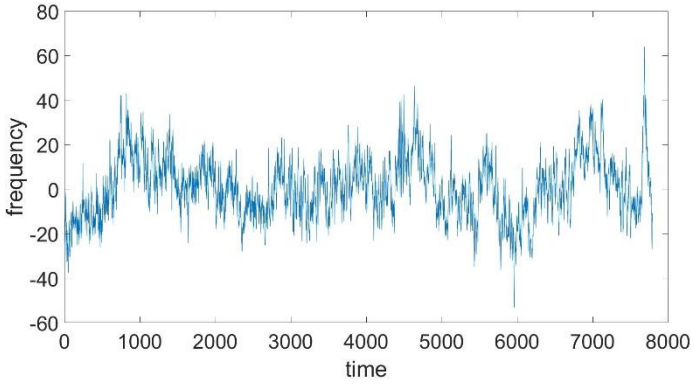


(b)

Gambar 5.1 (a)Contoh sinyal mentah dataset 2 training (b) Contoh sinyal mentah dataset 2 test

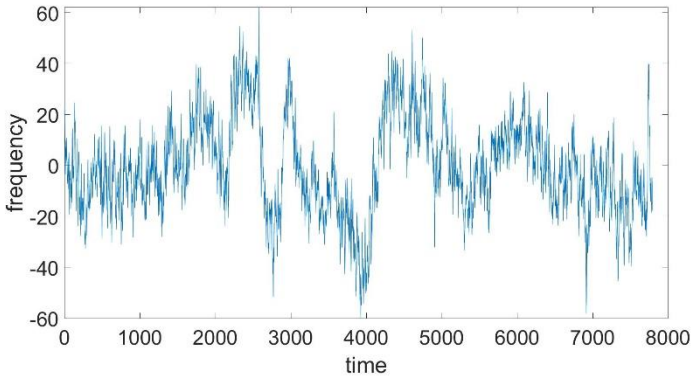


(a)

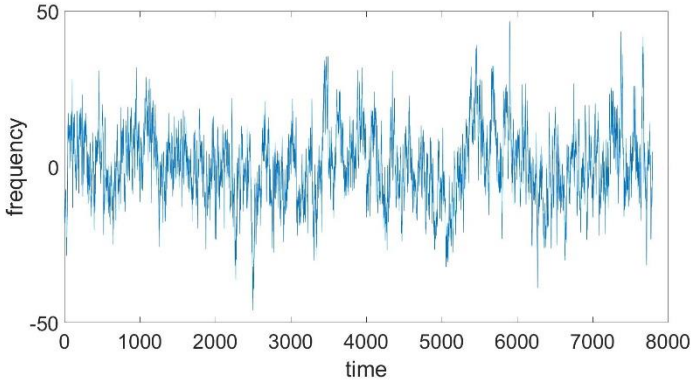


(b)

Gambar 5.2 (a)Contoh sinyal mentah dataset 3A training (b) Contoh sinyal mentah dataset 3A test

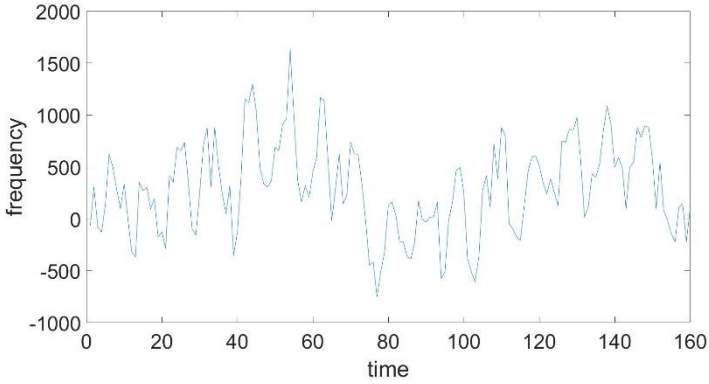


(a)

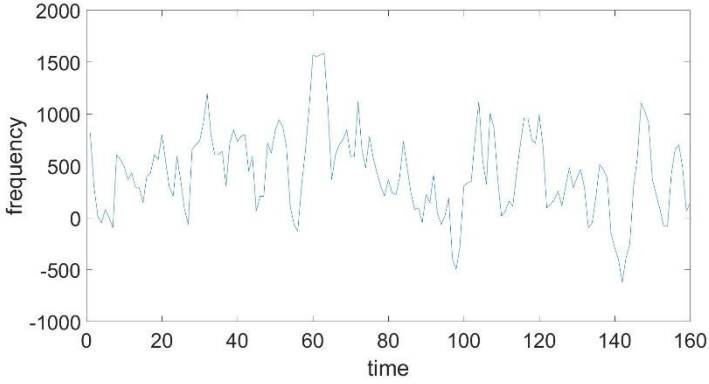


(b)

Gambar 5.3 (a)Contoh sinyal mentah dataset 3B training (b) Contoh sinyal mentah dataset 3B test

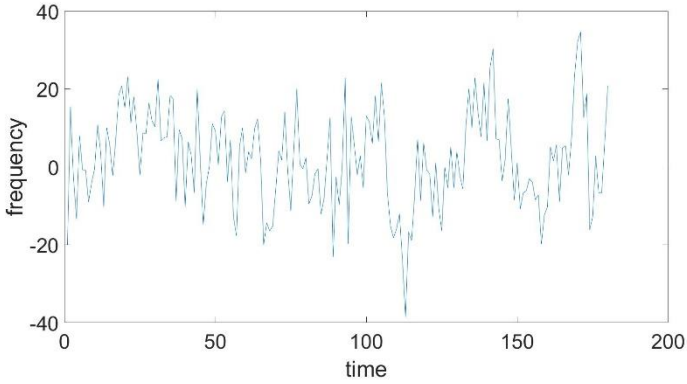


(a)

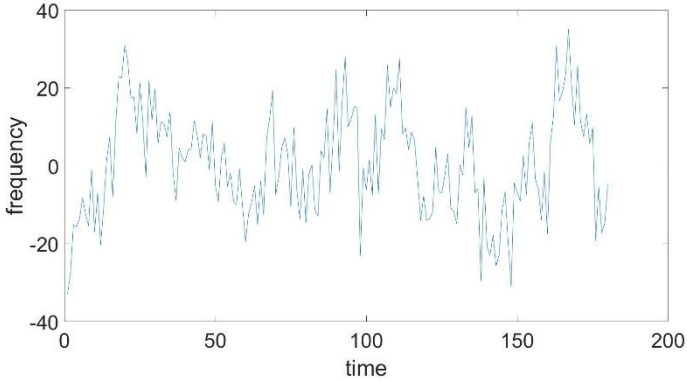


(b)

Gambar 5.4 (a) Contoh sinyal hasil pemotongan dataset 2 training (b) Contoh sinyal hasil pemotongan dataset 2 test

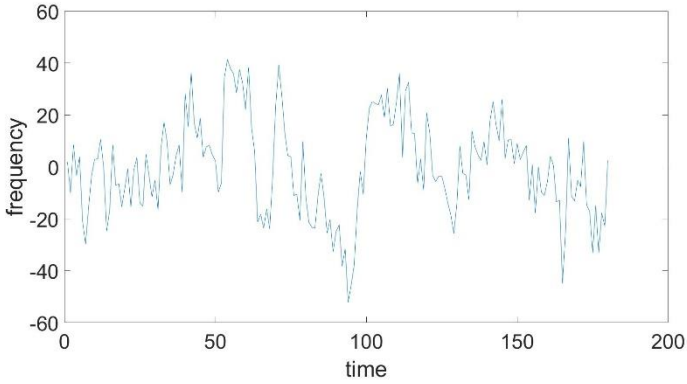


(a)

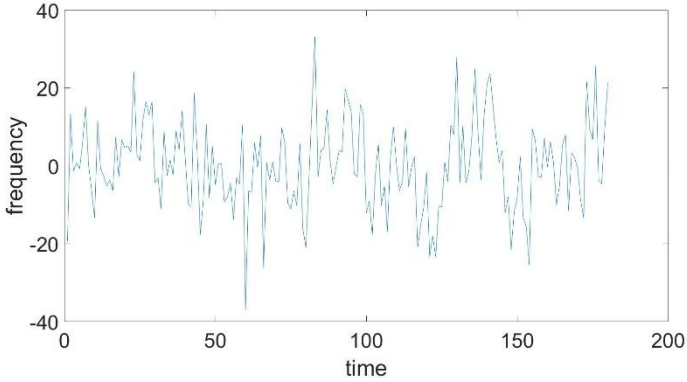


(b)

Gambar 5.5 (a) Contoh sinyal hasil pemotongan dataset 3A training (b) Contoh sinyal hasil pemotongan dataset 3A test



(a)



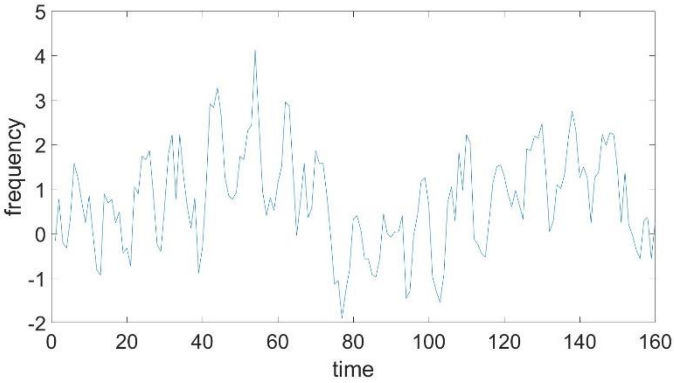
(b)

Gambar 5.6 (a) Contoh sinyal hasil pemotongan dataset 3B training (b) Contoh sinyal hasil pemotongan dataset 3B test

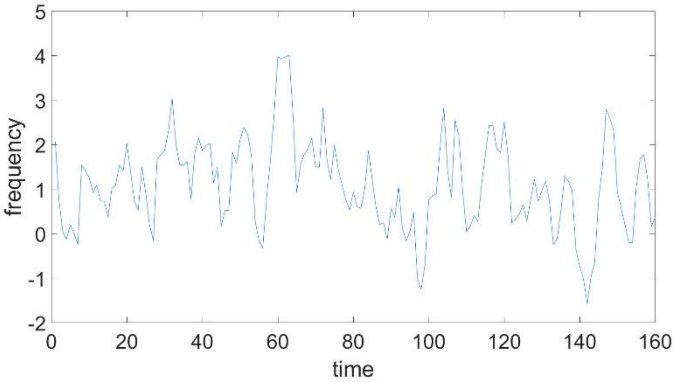
5.3.2 Uji coba Bandpass Filter

Pada sub bab ini akan dibahas mengenai hasil *Bandpass filter* pada matriks data sinyal yang telah dipotong. *Bandpass filter* akan diaplikasikan terlebih dahulu pada *training* data dan *testing*

data. Gambar 5.7 sampai 5.9 adalah contoh sinyal hasil *Bandpass* filter

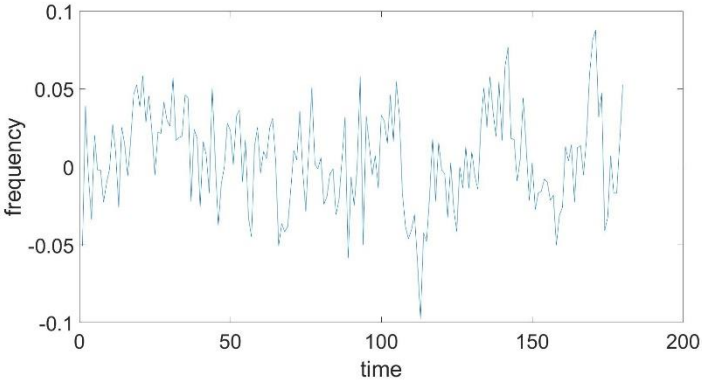


(a)

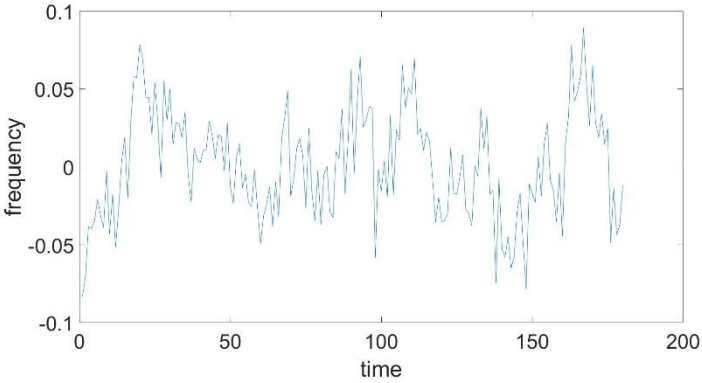


(b)

Gambar 5.7 (a) Contoh sinyal hasil Bandpass filter dataset 2 training (b) Contoh sinyal hasil Bandpass filter dataset 2 test

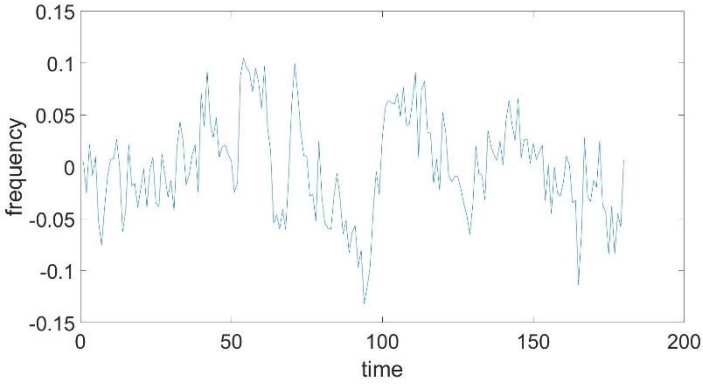


(a)

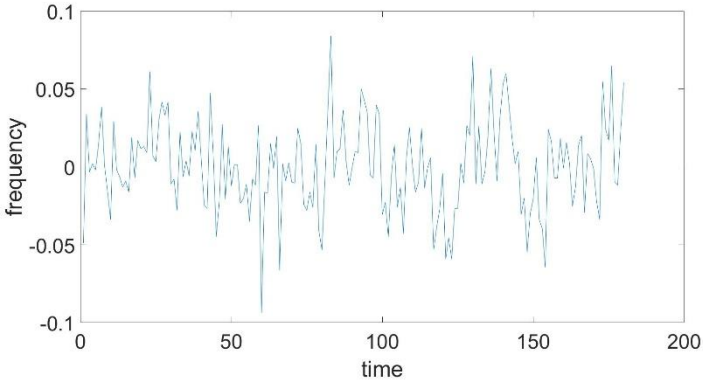


(b)

Gambar 5.8 (a) Contoh sinyal hasil Bandpass filter dataset 3A training (b) Contoh sinyal hasil Bandpass filter dataset 3A test



(a)

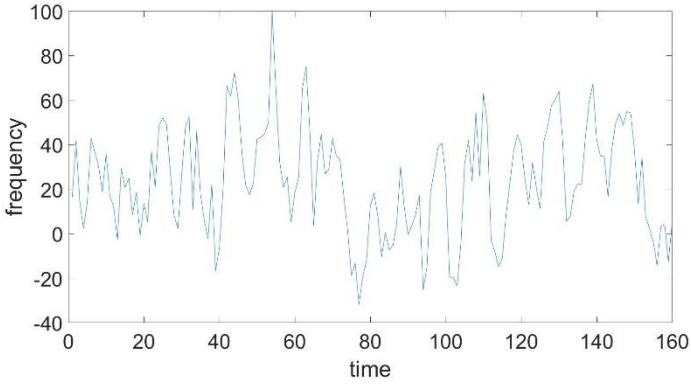


(b)

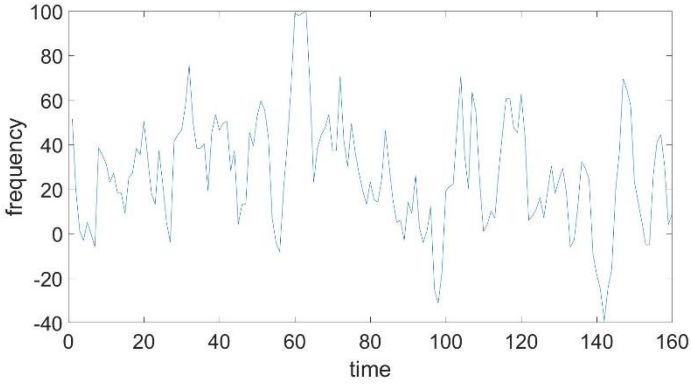
Gambar 5.9 (a) Contoh sinyal hasil Bandpass filter dataset 3B training (b) Contoh sinyal hasil Bandpass filter dataset 3B test

5.3.3 Uji Coba Min-max normalisation

Pada sub bab ini akan dibahas mengenai hasil *Min-max normalisation* pada matriks data sinyal. *Min-max normalisation* akan diaplikasikan terlebih dahulu pada *training* data dan *testing* data. Contoh sinyal hasil normalisasi dapat dilihat pada gambar 5.10 sampai 5.12

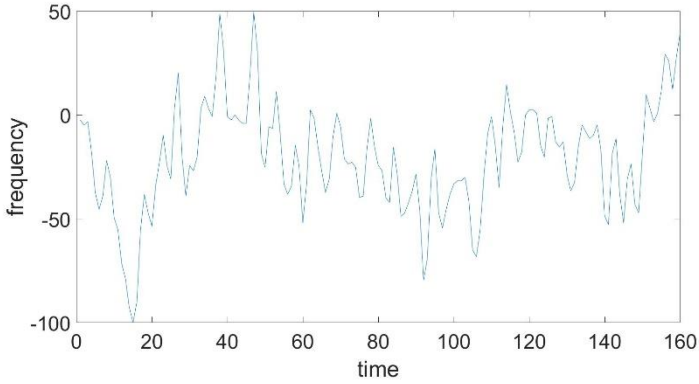


(a)

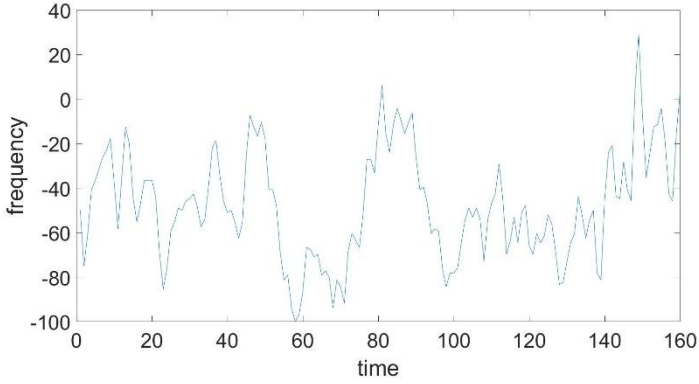


(b)

Gambar 5.10 (a) Contoh sinyal hasil normalisasi dataset 2 training (b) Contoh sinyal hasil normalisasi dataset 2 test

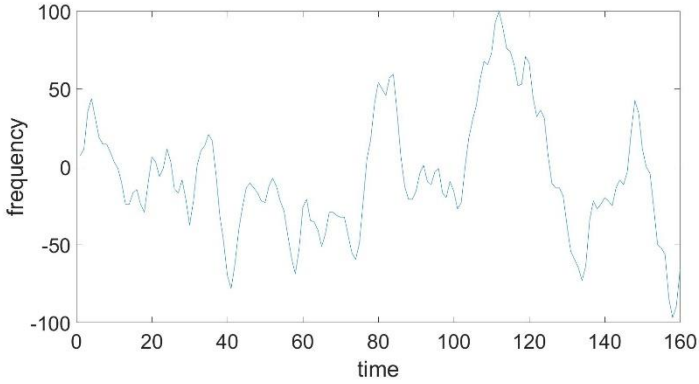


(a)

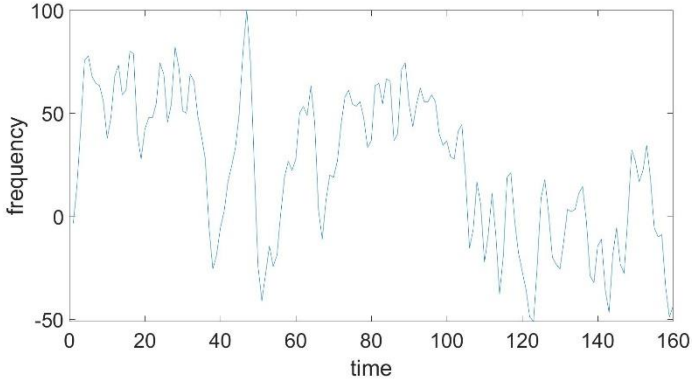


(b)

Gambar 5.11 (a) Contoh sinyal hasil normalisasi dataset 3A training (b) Contoh sinyal hasil normalisasi dataset 3A test



(a)



(b)

Gambar 5.12 (a) Contoh sinyal hasil normalisasi dataset 3B training (b) Contoh sinyal hasil normalisasi dataset 3B test

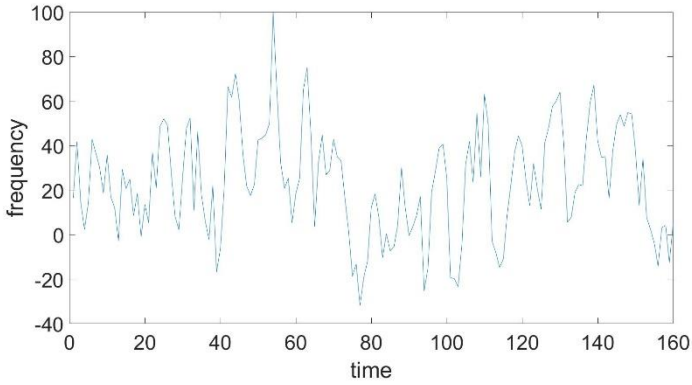
5.4 Skenario Uji Coba

Parameter / nilai yang dipilih dilihat berdasarkan 3 parameter evaluasi kerja yaitu akurasi, *sensitivity*, dan *specificity*. Parameter / nilai dipilih apabila memiliki nilai akurasi tertinggi, dan nilai *sensitivity* dan *specificity* diatas 60%. Parameter yang

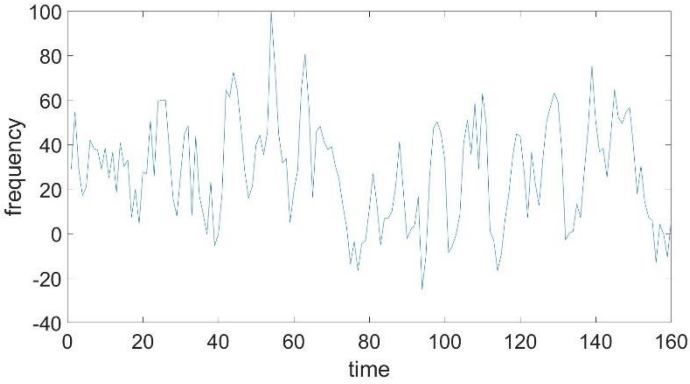
dipilih pada ujicoba pertama akan digunakan sebagai parameter tetap di uji coba berikutnya. Parameter yang dipilih akan diwarnai kuning pada tabel hasil uji cobanya, sedangkan parameter evaluasi terbaik akan diwarnai krem pada tabel hasil uji cobanya.

5.4.1 Skenario Uji Coba Averaging

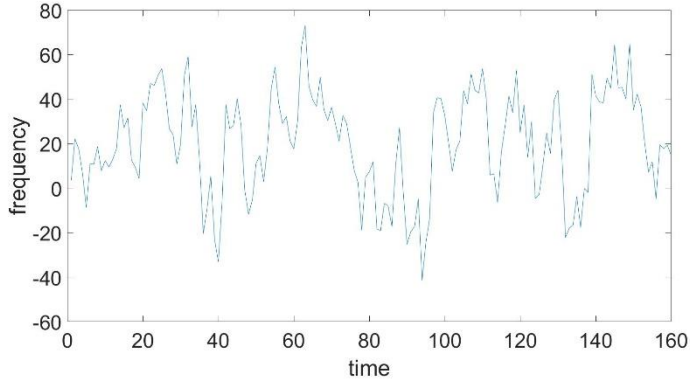
Sinyal yang telah dinormalisasi kemudian di *averaging* sebanyak 1, 2 dan 3 sinyal. Kemudian dilakukan evaluasi untuk mencari jumlah *signal averaging* terbaik dari ketiga skenario tersebut. Gambar 5.13 hingga 5.15 merupakan contoh perbandingan sinyal yang diaveraging sebanyak 1, 2, dan 3 sinyal. Hasil proses *training* dapat dilihat pada lampiran A.1. Hasil uji coba dapat dilihat pada tabel 5.1. *Confusion matrix* hasil klasifikasi dapat dilihat pada lampiran B.1. Dari hasil uji coba diketahui untuk ketiga dataset hasil terbaik didapat ketika sinyal diaveraging sebanyak 3 sinyal. Hal ini disebabkan karena averaging 3 sinyal mengurangi lebih banyak noise daripada 2 atau 1 sinyal, selain itu juga lebih menonjolkan ciri-ciri sinyal P300, sehingga lebih mudah untuk diklasifikasi.



Gambar 5.13 Contoh sinyal averaging 1 sinyal



Gambar 5.14 Contoh sinyal averaging 2 sinyal



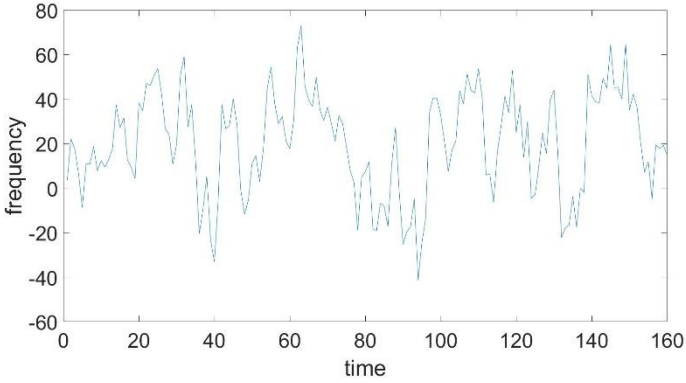
Gambar 5.15 Contoh sinyal averaging 3 sinyal

Tabel 5.1 Perbandingan hasil uji coba averaging

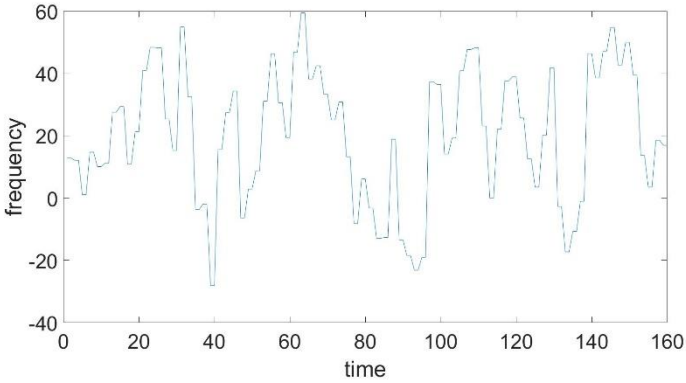
Dataset	Averaging	Akurasi	Sensitivity	Spesificity
2	1 sinyal	52.07	49.03	52.68
	2 sinyal	55.73	59.13	55.05
	3 sinyal	67.84	62.58	68.90
3A	1 sinyal	65.03	58.06	66.43
	2 sinyal	69.65	58.93	71.80
	3 sinyal	80.63	52.70	86.22
3B	1 sinyal	73.84	60.23	76.56
	2 sinyal	77.07	63.00	79.89
	3 sinyal	82.51	60.50	86.92

5.4.2 Skenario Uji Coba Penghalusan data

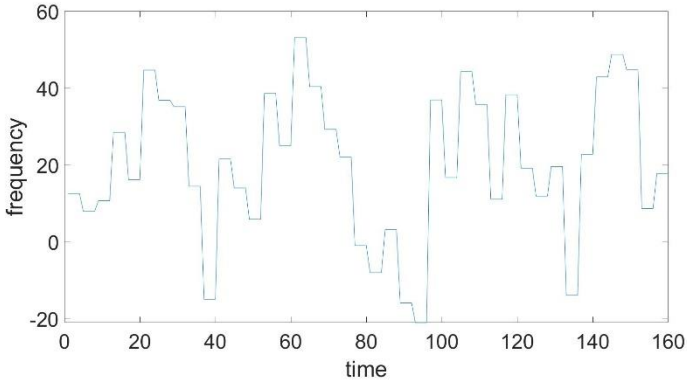
Pada sub bab ini akan dilakukan eksperimen dengan mencari level *inverse* HWT terbaik. Uji coba dilakukan dengan menghaluskan data sinyal hasil *averaging* dengan 6 level *inverse* HWT yang berbeda yaitu level 0, level 1, level 2, level 3, level 4, level 5, level 6. Gambar 5.16 hingga 5.21 merupakan contoh perbandingan sinyal yang dihaluskan menggunakan *inverse* HWT pada masing-masing scenario. Hasil proses *training* dapat dilihat pada lampiran A.2. Hasil uji coba dapat dilihat pada tabel 5.2. *Confusion matrix* hasil klasifikasi dapat dilihat pada lampiran B.2. Dapat dilihat bahwa pada dataset 2 level *inverse* HWT terbaik adalah level 0, sedangkan pada dataset 3A level *inverse* HWT terbaik adalah level 5 dan untuk dataset 3B level *inverse* HWT terbaik pada level 4. Perbedaan hasil terjadi karena perbedaan karakteristik sinyal masing-masing dataset.



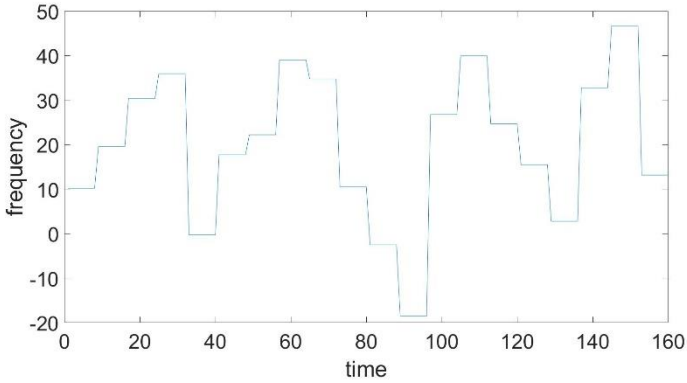
Gambar 5.16 Contoh sinyal inverse HWT level 0



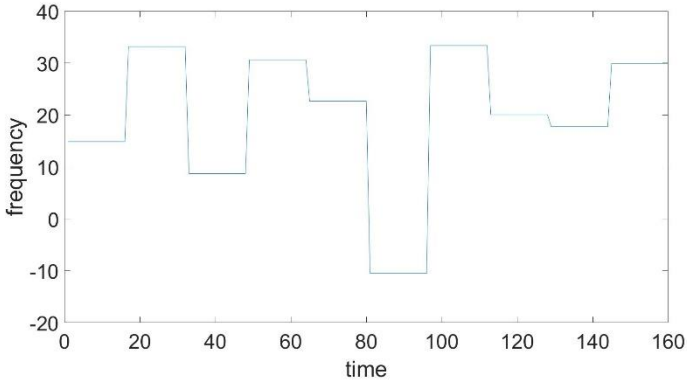
Gambar 5.17 Contoh sinyal inverse HWT level 1



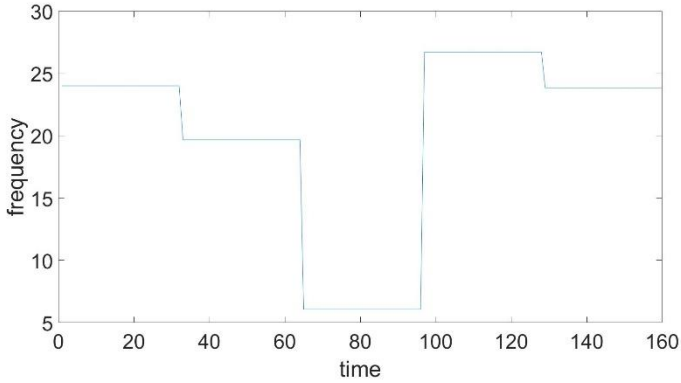
Gambar 5.18 Contoh sinyal inverse HWT level 2



Gambar 5.19 Contoh sinyal inverse HWT level 3



Gambar 5.20 Contoh sinyal inverse HWT level 4



Gambar 5.21 Contoh sinyal inverse HWT level 5

Tabel 5.2 Perbandingan hasil uji coba penghalusan data

Dataset	Inverse HWT	Akurasi	Sensitivity	Spesificity
2	level 0	67.85	62.58	68.90
	level 1	66.51	63.55	67.10
	level 2	63.55	67.10	62.84
	level 3	56.61	67.74	54.39
	level 4	64.09	61.61	64.58
	level 5	55.91	74.52	52.19
3A	level 0	80.63	52.70	86.22
	level 1	76.13	55.50	80.26
	level 2	76.77	65.10	79.10
	level 3	81.78	65.70	85.00
	level 4	78.48	71.00	79.98
	level 5	84.03	68.60	87.12
3B	level 0	82.52	60.50	86.92
	level 1	81.32	62.80	85.02
	level 2	85.18	67.70	88.68
	level 3	81.45	72.60	83.22
	level 4	86.63	71.10	89.74
	level 5	80.43	74.30	81.66

5.4.3 Skenario Uji Coba Batch Normalisation Neural Network

Pada sub bab ini akan dilakukan eksperimen dengan mengubah beberapa parameter pada arsitektur *Batch Normalisation Neural Network.* Penjelasan scenario uji coba masing-masing parameter adalah sebagai berikut

5.4.3.1 Jumlah Epoch

Pada uji coba ini dilakukan 4 skenario uji coba untuk menentukan jumlah *epoch* yang dapat menghasilkan hasil kinerja terbaik. Keempat scenario tersebut adalah 100 *epoch*, 400 *epoch*, 700 *epoch*, dan 1000 *epoch*. Hasil proses *training* dapat dilihat

pada lampiran A.3. Hasil uji coba dapat dilihat pada tabel 5.3. *Confusion matrix* hasil klasifikasi dapat dilihat pada lampiran B.3. Dari hasil uji coba diketahui untuk ketiga dataset, jumlah *epoch* yang memberikan kinerja terbaik adalah 1000 *epoch*. Hal ini disebabkan karena semakin banyak *epoch* dapat membuat model dapat belajar lebih tepat untuk mengklasifikasikan sinyal.

Tabel 5.3 Perbandingan hasil uji coba jumlah epoch

Dataset	Jumlah epoch	Akurasi	Sensitivity	Spesificity
2	100 epoch	60.54	55.16	61.61
	400 epoch	60.86	63.22	60.38
	700 epoch	67.42	55.80	69.74
	1000 epoch	67.85	62.58	68.90
3A	100 epoch	75.13	78.20	74.52
	400 epoch	77.65	75.00	78.18
	700 epoch	77.43	73.30	78.26
	1000 epoch	84.03	68.60	87.12
3B	100 epoch	76.90	77.70	76.74
	400 epoch	82.66	76.00	84.00
	700 epoch	82.43	75.60	83.80
	1000 epoch	86.63	71.10	89.74

5.4.3.2 Dropout Probability

Setelah mengetahui parameter jumlah *epoch* terbaik, ujicoba selanjutnya mencari parameter *dropout probability* terbaik. Terdapat 5 skenario uji coba untuk mencari nilai *dropout probability* terbaik, yaitu *dropout probability* dengan nilai parameter 0, 0.2, 0.4, 0.6, 0.8. . Hasil proses *training* dapat dilihat pada lampiran A.4. Hasil uji coba dapat dilihat pada tabel 5.4. *Confusion matrix* hasil klasifikasi dapat dilihat pada lampiran B.4. Dapat dilihat bahwa pada dataset 2 nilai *dropout probability* terbaik adalah 0.2 , sedangkan pada dataset 3A nilai *dropout*

probability terbaik adalah 0.8, dan untuk dataset 3B nilai *dropout probability* terbaik adalah 0.

Tabel 5.4 Perbandingan hasil uji coba dropout probability

Dataset	Dropout probability	Akurasi	Sensitivity	Spesificity
2	0	69.67	63.87	70.83
	0.2	70.80	54.83	74.00
	0.4	68.11	61.29	69.48
	0.6	70.75	55.80	73.74
	0.8	67.84	62.58	68.90
3A	0	83.03	69.20	85.80
	0.2	77.76	73.20	78.68
	0.4	78.31	72.10	79.56
	0.6	77.63	72.80	78.60
	0.8	84.03	68.60	87.12
3B	0	86.85	71.10	90.00
	0.2	85.78	73.10	88.32
	0.4	82.35	75.20	83.78
	0.6	82.58	75.70	83.96
	0.8	86.63	71.10	89.74

5.4.3.3 Fully Connected Layer

Uji coba selanjutnya mencari jumlah kernel pada *Fully Connected Layer* terbaik. Terdapat 4 skenario uji coba untuk mencari jumlah kernel pada *Fully Connected Layer* terbaik, yaitu kernel *Fully Connected Layer* sejumlah 64, 128, 192, dan 256. Hasil proses *training* dapat dilihat pada lampiran A.5. Hasil uji coba dapat dilihat pada tabel 5.5. *Confusion matrix* hasil klasifikasi dapat dilihat pada lampiran B.5. Dapat dilihat bahwa untuk ketiga dataset jumlah kernel terbaik adalah 128 .

Tabel 5.5 Perbandingan hasil uji coba Fully Connected Layer

Dataset	Jumlah Kernel	Akurasi	Sensitivity	Spesificity
2	64 kernel	68.27	57.74	70.38
	128 kernel	70.80	54.83	74.00
	192 kernel	67.41	66.77	67.54
	256 kernel	69.24	57.41	71.61
3A	64 kernel	77.88	78.30	77.8
	128 kernel	84.03	68.60	87.12
	192 kernel	76.73	73.30	77.42
	256 kernel	77.63	75.20	78.12
3B	64 kernel	82.25	75.00	83.70
	128 kernel	86.85	71.10	90.00
	192 kernel	85.83	72.90	88.42
	256 kernel	85.05	71.50	87.76

5.4.3.4 Convolutional Layer

Selain jumlah kernel pada *Fully Connected Layer*, ujicoba juga dilakukan untuk mencari jumlah kernel pada layer konvolusi di L1. Terdapat 3 skenario uji coba untuk mencari jumlah kernel pada *Convolutional Layer* terbaik, yaitu kernel *Convolutional Layer* sejumlah 1, 8, dan 16. . Hasil proses training dapat dilihat pada lampiran A.6. Hasil uji coba dapat dilihat pada tabel 5.6. *Confusion matrix* hasil klasifikasi dapat dilihat pada lampiran B.6. Dapat dilihat bahwa untuk ketiga dataset jumlah kernel terbaik adalah 16 kernel .

Tabel 5.6 Perbandingan hasil uji coba Convolutional Layer

Dataset	Jumlah kernel	Akurasi	Sensitivity	Spesificity
2	1 kernel	48.22	90.32	39.80
	8 kernel	64.40	70.00	63.29
	16 kernel	70.80	54.83	74.00
3A	1 kernel	79.36	75.90	80.06
	8 kernel	78.38	78.40	78.38
	16 kernel	84.03	68.60	87.12
3B	1 kernel	84.21	79.90	85.08
	8 kernel	84.85	70.70	87.68
	16 kernel	86.85	71.10	90.00

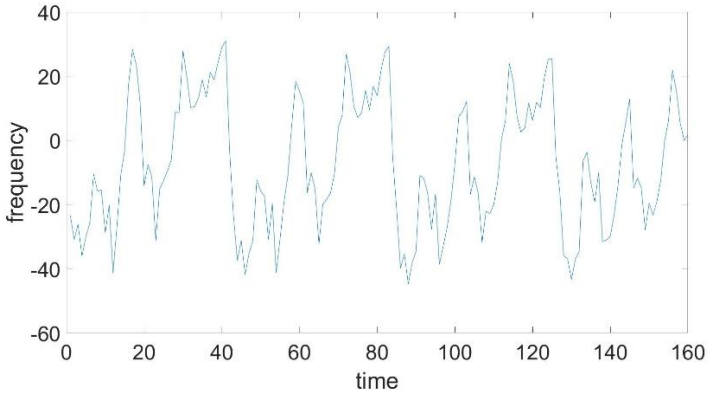
5.5 Evaluasi

Dari Hasil scenario uji coba *averaging* didapatkan hasil akurasi terbaik untuk ketiga dataset adalah *averaging* 3 sinyal. Hal ini disebabkan karena averaging 3 sinyal mengurangi lebih banyak noise daripada 2 atau 1 sinyal, selain itu juga lebih menonjolkan ciri-ciri sinyal P300, sehingga lebih mudah untuk diklasifikasi. Sedangkan pada uji coba penghalusan data pada masing-masing dataset terdapat level inverse HWT yang berbeda-beda. Hal ini disebabkan oleh karakteristik sinyal masing-masing individu yang berbeda.

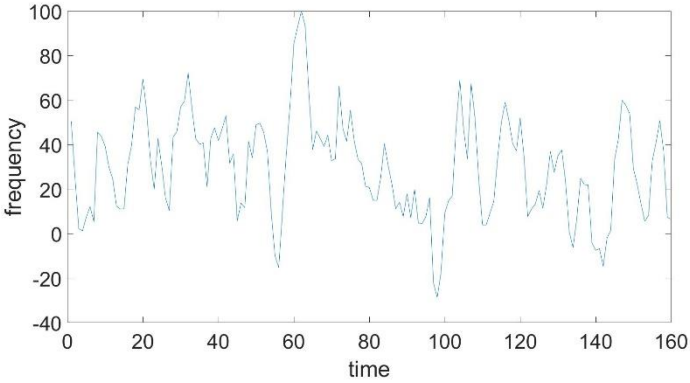
Untuk uji coba jumlah *epoch* diketahui untuk ketiga data, jumlah *epoch* yang memberikan kinerja terbaik adalah 1000 *epoch*. Hal ini disebabkan karena semakin banyak epoch dapat membuat model dapat belajar lebih tepat untuk mengklasifikasikan sinyal. Pada uji coba *Dropout probability* hasil terbaik masing-masing dataset berbeda-beda, disebabkan oleh karakteristik inputan sinyal hasil penghalusan yang berbeda-beda. Untuk uji coba *Fully*

Connected Layer didapat hasil terbaik untuk ketiga dataset pada saat jumlah kernelnya 128. Hal ini disebabkan karena pada saat jumlah kernelnya 128 tidak terdapat informasi yang hilang akibat perbedaan input dan ukuran kernel. Untuk ujicoba *Convolutional Layer*, pada saat jumlah kernelnya 16 informasi yang didapat lebih banyak jika dibandingkan dengan 1 kernel dan 8 kernel, sehingga untuk ketiga dataset, 16 kernel memberikan hasil terbaik.

Dari hasil uji coba didapatkan untuk parameter *sensitivity* memiliki nilai yang rendah, disebabkan oleh sinyalnya yang dinamis. Sinyal yang dinamis terjadi karena pengaruh kualitas alat yang digunakan untuk merekam sinyal dan focus atau tidaknya subjek. Pada Gambar 5.22 merupakan contoh sinyal yang salah diklasifikasi, sedangkan Gambar 5.23 merupakan contoh sinyal yang diklasifikasi dengan benar. Sinyal pada Gambar 5.22 diklasifikasikan menjadi sinyal non-P300, seharusnya diklasifikasikan menjadi sinyal P300. Hal ini terjadi akibat bentuk sinyal yang mirip antara sinyal P300 dan sinyal non-P300 yang dapat dilihat pada Gambar 5.23 tersebut. Alasan lain terjadinya salah klasifikasi karena fitur dari kedua sinyal yang mirip. Di BN3 diambil beberapa fitur dari sinyal tersebut, yang mungkin saja mirip satu sama lainnya.



Gambar 5.22 Contoh sinyal salah klasifikasi



Gambar 5.23 Contoh sinyal yang benar diklasifikasi

[Halaman ini sengaja dikosongkan]

BAB VI
PENUTUP

Pada bab ini dijelaskan mengenai kesimpulan yang didapatkan setelah melakukan serangkaian uji coba. Selain itu, dijelaskan juga saran pengembangan klasifikasi sinyal EEG dengan *Batch Normalisation Neural Network* selanjutnya.

6.1. Kesimpulan

Berdasarkan kajian, uji coba, dan evaluasi yang telah dilakukan, dapat disimpulkan beberapa hal penting sebagai berikut:

1. Averaging 3 sinyal memberikan hasil terbaik dibandingkan dengan averaging 1 atau 2 sinyal, dengan hasil akurasi terbaik 82.51%
2. HWT terbaik untuk dataset 2 adalah level 0 dengan hasil akurasi 67.85%, untuk dataset 3A adalah level 5 dengan hasil akurasi 84.03%, dan untuk dataset 3B adalah level 4 dengan hasil akurasi 86.63%
3. Jumlah Epoch terbaik untuk dataset 2, 3A, dan 3B adalah 1000 dengan hasil akurasi terbaik 86.63%
4. Nilai Dropout Probability terbaik untuk dataset 2 adalah 0 dengan hasil akurasi 70.80%, untuk dataset 3A adalah 0.8 dengan hasil akurasi 84.03% dan untuk dataset 3B adalah 0 dengan hasil akurasi 86.63%
5. Jumlah Fully Connected Layer terbaik untuk dataset 2, 3A, dan 3B adalah 128 dengan hasil akurasi terbaik 86.85%
6. Jumlah Filter Convolution pada filter spasial terbaik untuk dataset 2, 3A, dan 3B adalah 16 dengan hasil akurasi terbaik 86.85%
7. Secara keseluruhan, setelah melalui serangkaian uji coba dengan metode *simple split* diperoleh hasil akurasi tertinggi 86.85%, *sensitivity* terbaik 71.10%, dan *specificity* terbaik 90.00%.

6.2. Saran

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

1. Panjang pemotongan sinyal data mentah dapat diperpanjang untuk mengantisipasi respon stimulus subjek yang berbeda-beda
2. Proses normalisasi sinyal yang telah dipotong kedepannya perlu diuji pengaruh yang diberikan terhadap proses klasifikasi
3. Tugas akhir ini dapat dilanjutkan ke penelitian selanjutnya untuk mendeteksi kata-kata pada saat sinyal P300 terjadi

DAFTAR PUSTAKA

[1] D. Zhang , H. Song , H. Xu , W. Wu , S. Gao , B. Hong , An N200 speller integrating the spatial profile for the detection of the non-control state, J. Neural Eng. 9 (2) (2012) 026016 .
[2] J. Jin , B.Z. Allison , E.W. Sellers , C. Brunner , P. Horki , X. Wang , C. Neuper , An adaptive P300-based control system, J. Neural Eng. 8 (3) (2011) 036006 .
[3] Z. Gu , Z. Yu , Z. Shen , Y. Li , An online semi-supervised brain–computer interface, IEEE Trans. Biomed. Eng. 60 (9) (2013) 2614–2623 .
[4] A. Rakotomamonjy , V. Guigue , BCI competition III: dataset II-ensemble of SVMs for BCI P300 speller, IEEE Trans. Biomed. Eng. 55 (3) (2008) 1147–1154 .
[5] D. Krusienski , G. Schalk ,in: BCI Competition III Challenge 2004.
[6] Y. Li , Z. Ma , W. Lu , Y. Li , Automatic removal of the eye blink artifact from EEG using an ICA-based template matching approach, Physiol. Meas. 27 (4) (2006) 425 .
[7] V. Bostanov, BCI competition 2003-data sets Ib and IIb: feature extraction from event-related brain potentials with the continuous wavelet transform and the t-value scalogram, IEEE Trans. Biomed. Eng. 51 (6) (2004) 1057–1061, doi: 10. 1109/TBME.2004.826702 .
[8] B. Blankertz , Documentation second wadsworth BCI dataset (P300 evoked po- tentials) data acquired using BCI20 0 0 P30 0 speller paradigm, Proceedings of the BCI Classification Contest, November 2002 .
[9] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv: 1409.1556 , 2014.
[10] K. He , X. Zhang , S. Ren , J. Sun , Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recogni- tion, 2016, pp. 770–778 .

[11] W. Wu , S. Nagarajan , Z. Chen ,Bayesian machine learning: EEG/MEG signal pro- cessing measurements, IEEE Signal Process. Mag. 33 (1) (2016) 14–36 .
[12] H. Cecotti , A. Graser , Convolutional neural networks for p300 detection with application to brain–computer interfaces, IEEE Trans. Pattern Anal. Mach. Intell. 33 (3) (2011) 433–445 .
[13] M. Längkvist , L. Karlsson , A. Loutfi, Sleep stage classification using unsuper- vised feature learning, Adv. Arti. Neural Syst. 2012 (2012) 5 .
[14] D. Wulsin , J. Gupta , R. Mani , J. Blanco , B. Litt , Modeling electroencephalogra- phy waveforms with semi-supervised deep belief nets: fast classification and anomaly measurement, J. Neural Eng. 8 (3) (2011) 036015 .
[15] S. Stober , D.J. Cameron , J.A. Grahn , Using convolutional neural networks to recognize rhythm stimuli from electroencephalography recordings, in: Pro- ceedings of the Advances in Neural Information Processing Systems, 2014, pp. 1449–1457 .
[16] Larry D. Paarmann, Design and Analysis of Analog Filters, A Signal Processing Perspective with MATLAB® Examples, Kluwer Academic Publishers, 2001.
[17] Shalabi, L.A., Z. Shaaban and B. Kasasbeh, *Data Mining: A Preprocessing Engine*, J. Comput. Sci., 2: 735-739, 2006
[18] *S.Gopal Krishna Patro, Pragyan Parimita Sahoo, Ipsita Panda, Kishore Kumar Sahu, "Technical Analysis on Financial Forecasting", International Journal of Computer Sciences and Engineering, Volume-03, Issue-01, Page No (1-6), E-ISSN: 2347-2693, Jan -2015*
[19] Tompkins, W. J. and Webster, J. G. (eds.) 1981. Design of Microcomputer-based Medical Instrumentation. Englewood Cliffs, NJ: Prentice Hall.
[20] P. Tagare, "Signal Averaging" in Biomedical Digital Signal Processing (J. Tomkins, Ed,), Prentice Hall, 1993
[21] Walker, James. (2008). A Primer on Wavelets and Their Scientific Applications.

[22] HORGAN., G. W. Wavelets for data smoothing: a review and some simulation results
[23] Matlab. (2016, September 15). *ihaart*. Retrieved from Mathworks: https://www.mathworks.com/help/wavelet/ref/ihaart.html#bveev6r-5
[24] D. J. Brownlee., "Machine Learning Mastery," 16 August 2016. [Online]. Available: https://machinelearningmastery.com/what-is-deep-learning/. [Diakses 26 December 2017].
[25] A. Ng, "Deep Learning," [Online]. Available: http://ufldl.stanford.edu/tutorial/supervised/ConvolutionalNeuralNetwork/. [Diakses 26 December 2017].
[26] Matlab. (2016, March 3). *2-D Convolutional Layer*. Retrieved from Mathworks: https://www.mathworks.com/help/nnet/ref/nnet.cnn.layer.convolution2dlayer.html
[27] Matlab. (2016, March 3). *Fully Connected Layer*. Retrieved from Matchworks: https://www.mathworks.com/help/nnet/ref/nnet.cnn.layer.fullyconnectedlayer.html
[28] J. Collis, "Medium," 28 June 2017. [Online]. Available: https://medium.com/deeper-learning/glossary-of-deep-learning-batch-normalisation-8266dcd2fa82. [Diakses 26 December 2017].
[29] M. Liu et al., Deep learning based on Batch Normalization for P300 signal detection, Neurocomputing (2017), http://dx.doi.org/10.1016/j.neucom.2017.08.039
[30]
DLI, A. (2018, July 8). *Synchronous Averaging*. Retrieved July 8, 2018, from Azima DLI: http://azimadli.com/vibman/synchronousaveraging.htm
[31] Kashyap, R. (2012, September 10). *Haar Wavelet Transform*. Retrieved from Mathworks: https://www.mathworks.com/matlabcentral/fileexchange/38061-haar-wavelet-transform?focused=5246584&tab=function

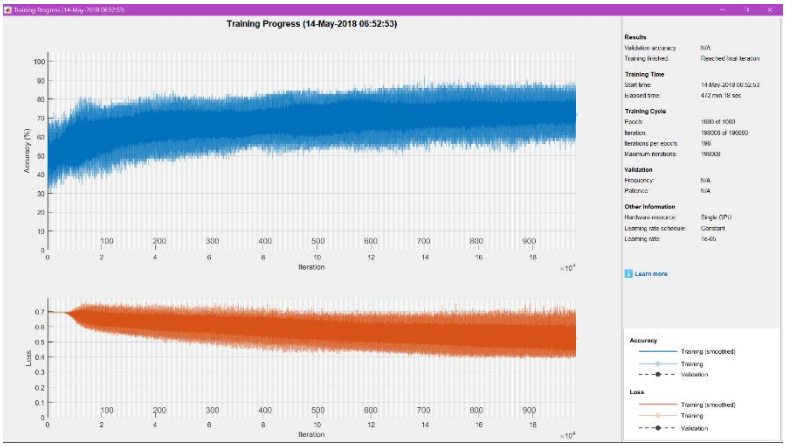
[32] Kashyap, R. (2012, September 10). *Haar Wavelet Transform*. Retrieved from Mathworks: https://www.mathworks.com/matlabcentral/fileexchange/38061-haar-wavelet-transform?focused=5246584&tab=function
[33] Kashyap, R. (2012, September 10). *Haar Wavelet Transform*. Retrieved from Mathworks: https://www.mathworks.com/matlabcentral/fileexchange/38061-haar-wavelet-transform?focused=5246584&tab=function
[34] Ando, Kota & Takamaeda-Yamazaki, Shinya & Ikebe, Masayuki & Asai, Tetsuya & Motomura, Masato. (2017). A Multithreaded CGRA for Convolutional Neural Network Processing. Circuits and Systems. 08. 149-170. 10.4236/cs.2017.86010.
[35] Santos, L. A. (n.d.). *Dropout Layer*. Retrieved from Gitbook: https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/dropout_layer.html

LAMPIRAN

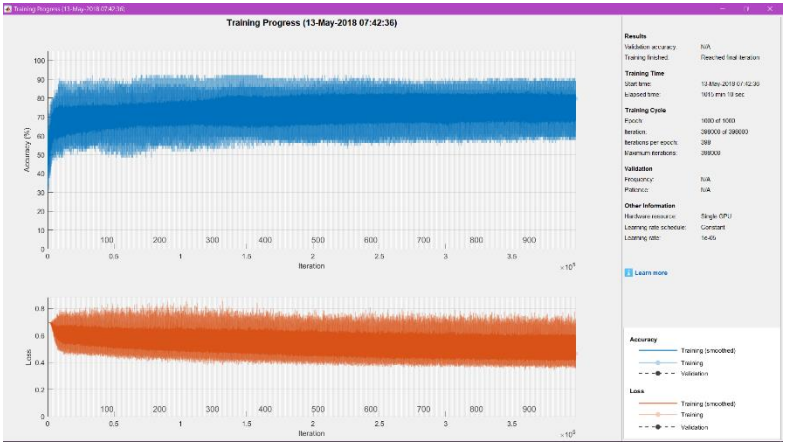
A. Hasil proses training

A.1. Hasil Training scenario uji coba averaging

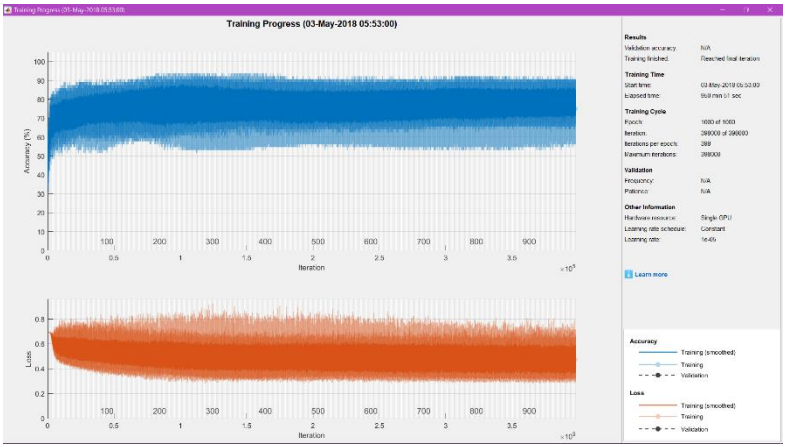
A.1.1. Uji coba 1 Sinyal



Hasil training dataset 2

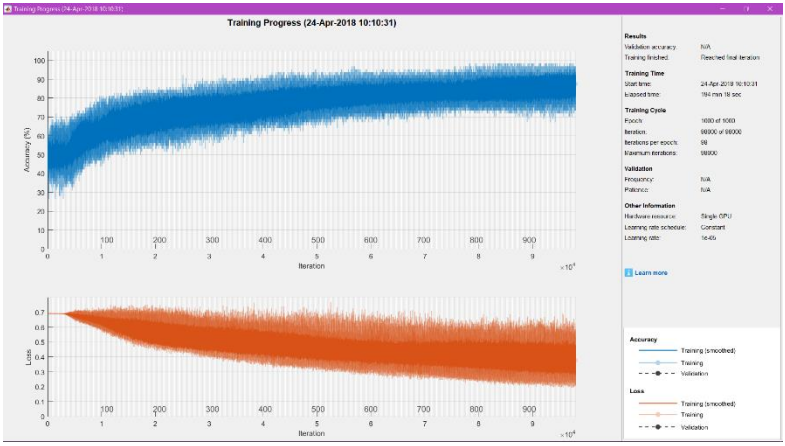


Hasil training dataset 3A

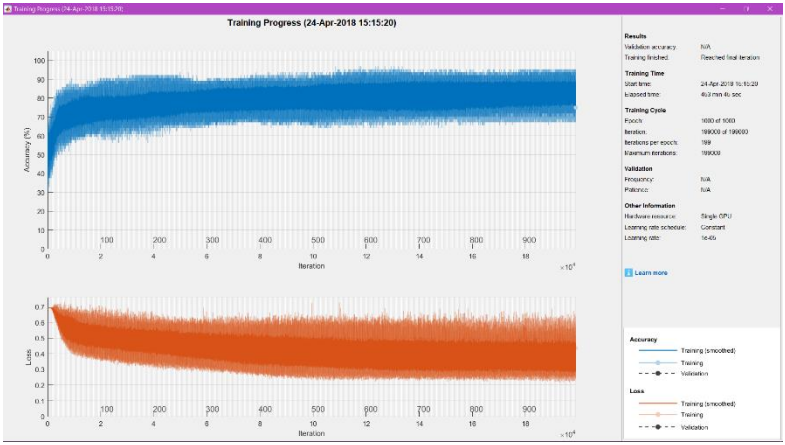


Hasil training dataset 3B

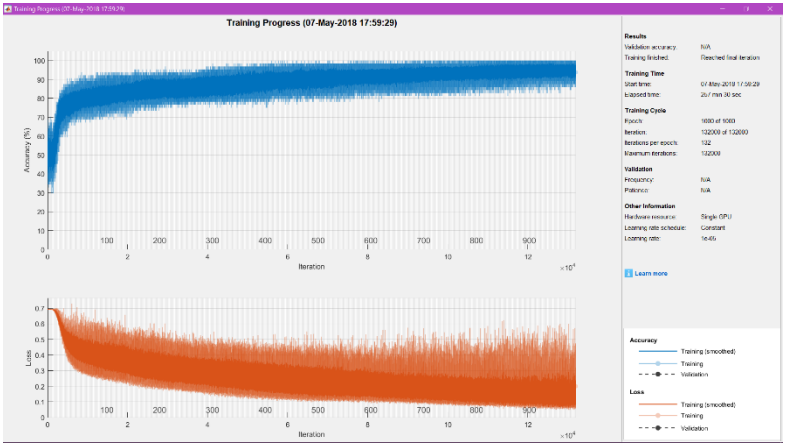
A.1.2. Uji coba 2 Sinyal



Hasil training dataset 2

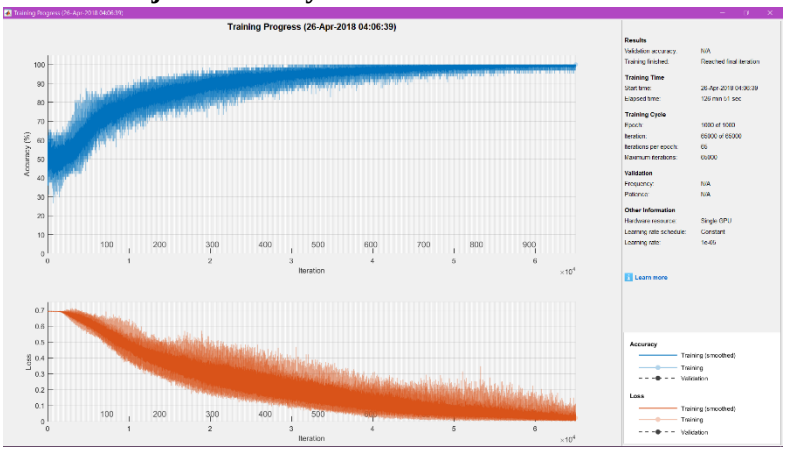


Hasil training dataset 3A

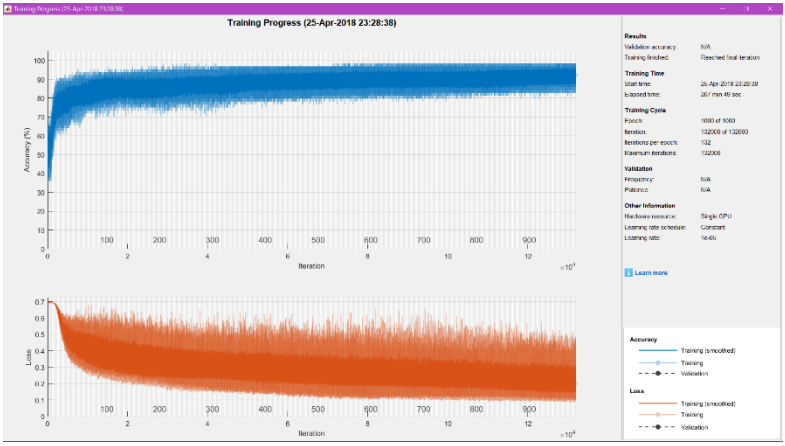


Hasil training dataset 3B

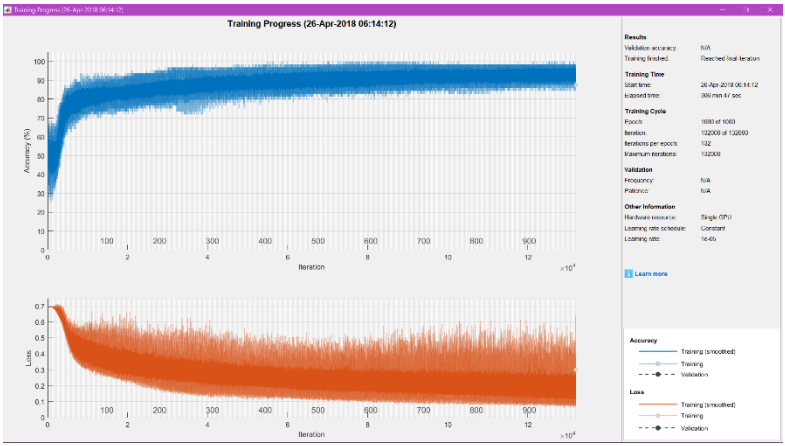
A.1.3. Uji coba 3 Sinyal



Hasil training dataset 2



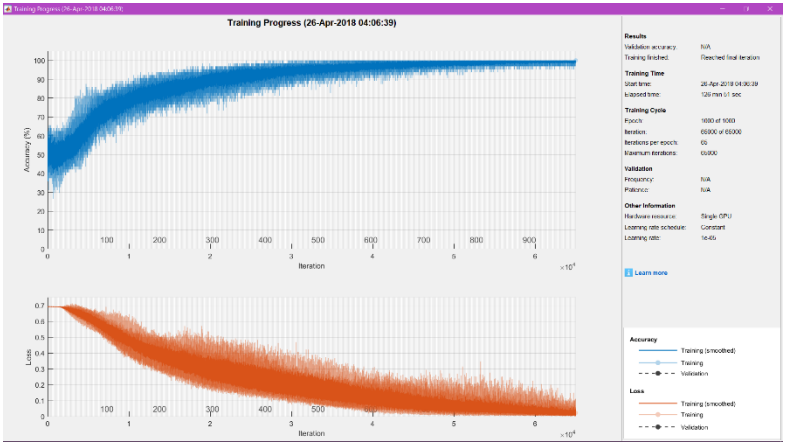
Hasil training dataset 3A



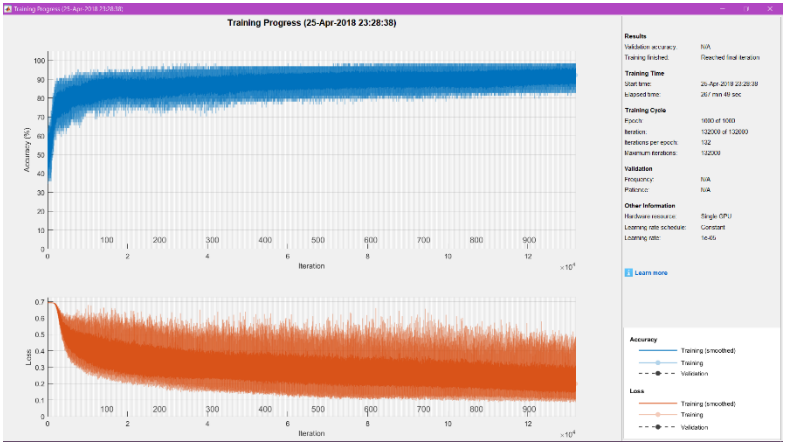
Hasil training dataset 3B

A.2. Hasil Training scenario uji coba penghalusan data

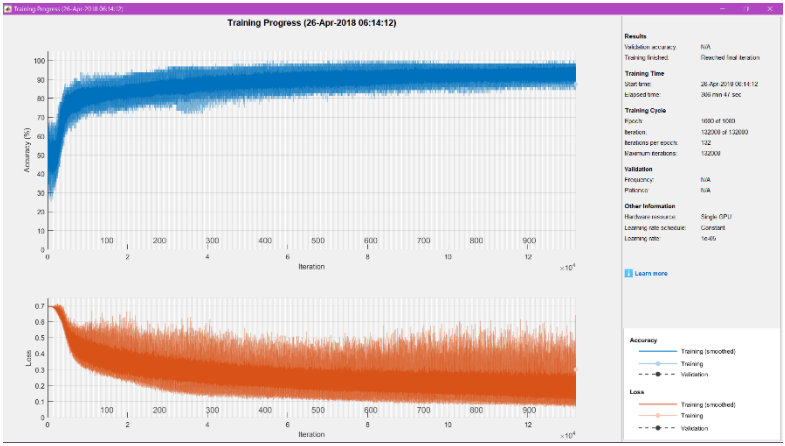
A.2.1. Level 0



Hasil training dataset 2

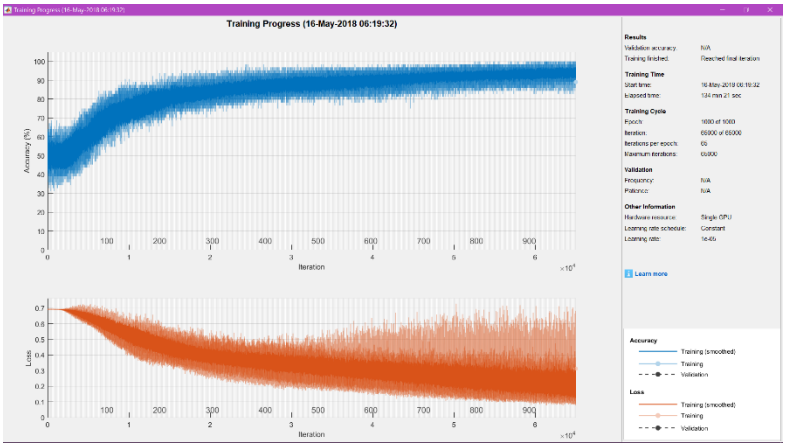


Hasil training dataset 3A

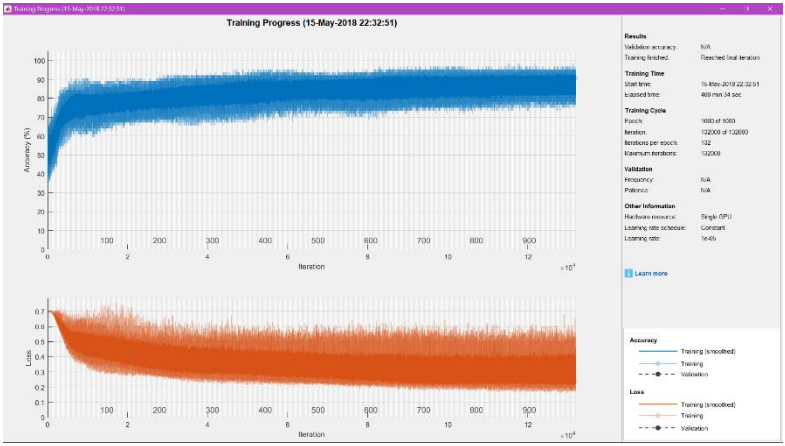


Hasil training dataset 3B

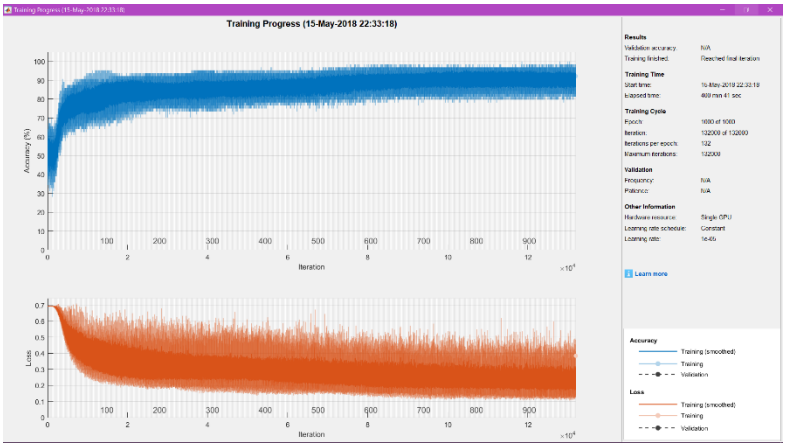
A.2.2. Level 1



Hasil training dataset 2

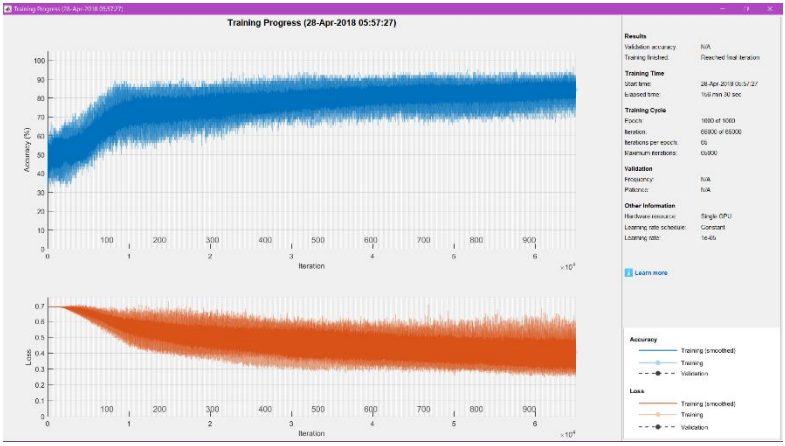


Hasil training dataset 3A

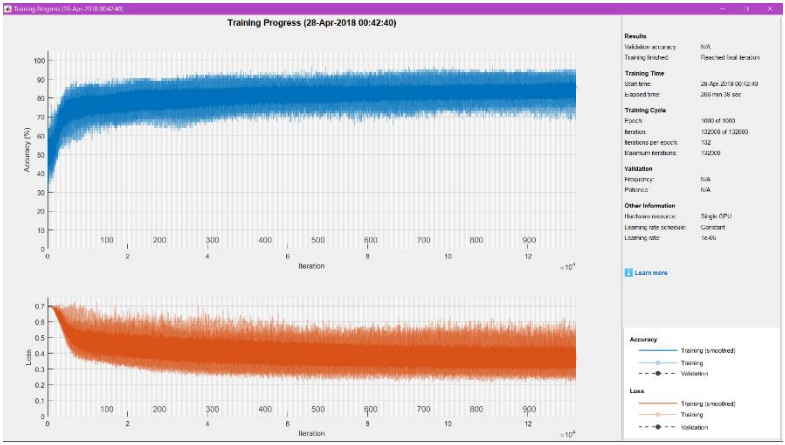


Hasil training dataset 3B

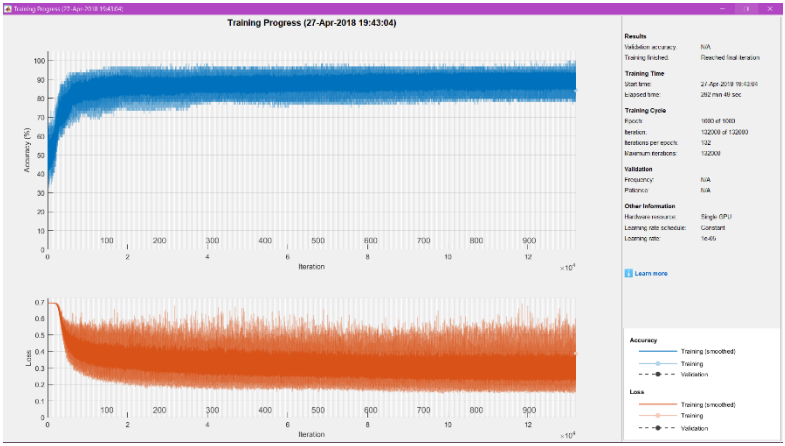
A.2.3. Level 2



Hasil training dataset 2

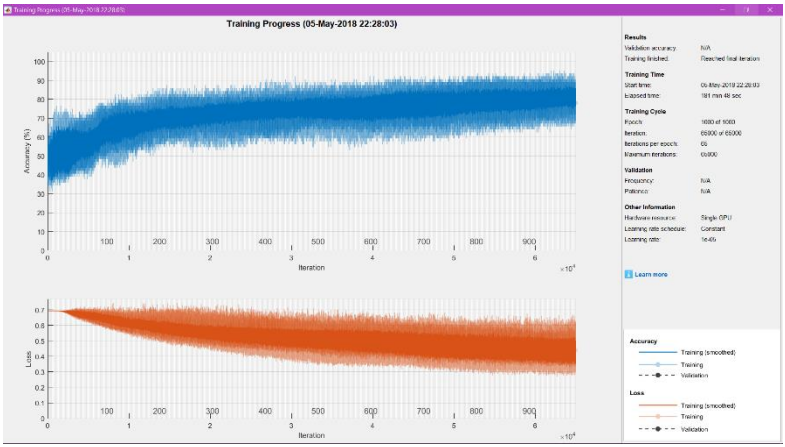


Hasil training dataset 3A

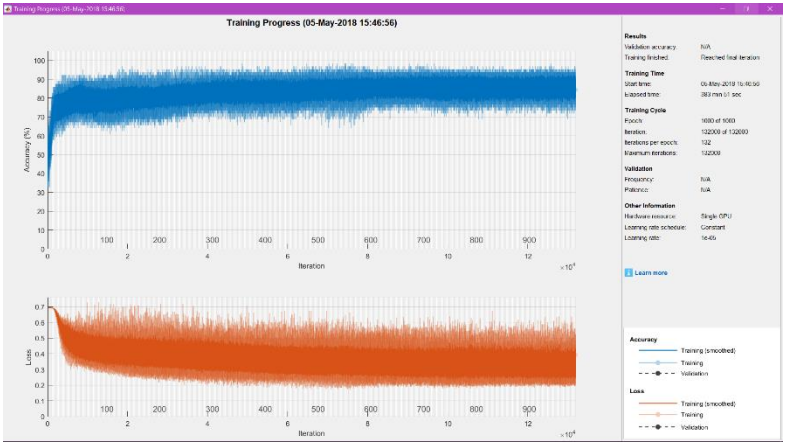


Hasil training dataset 3B

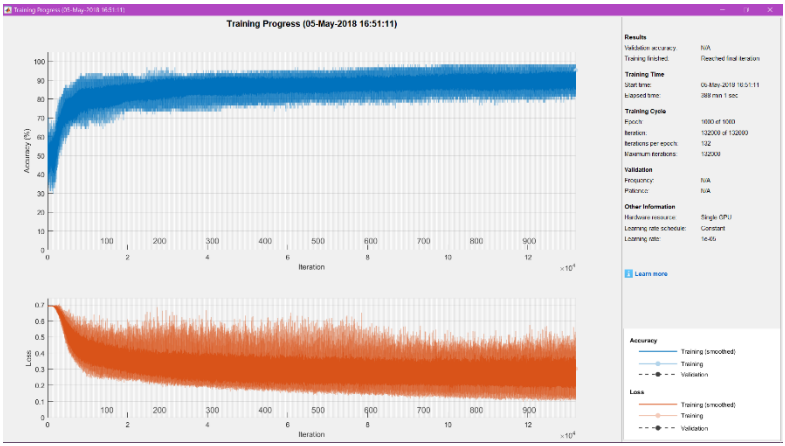
A.2.4. Level 3



Hasil training dataset 2

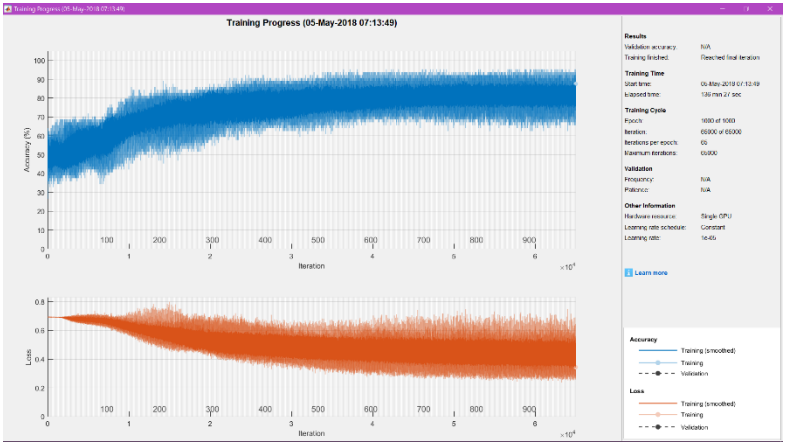


Hasil training dataset 3A

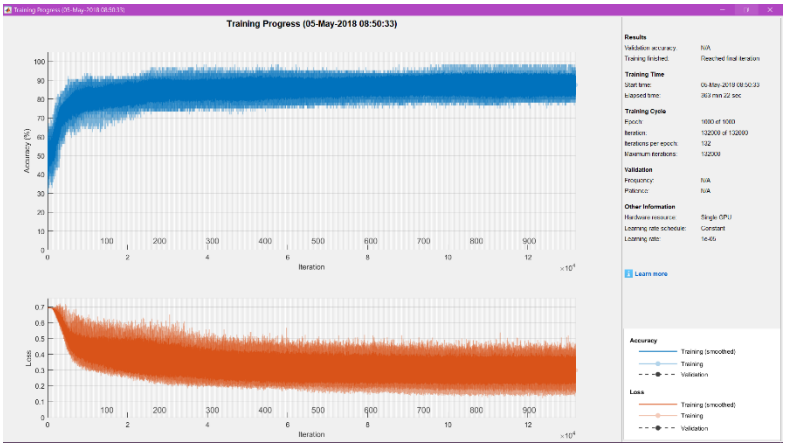


Hasil training dataset 3B

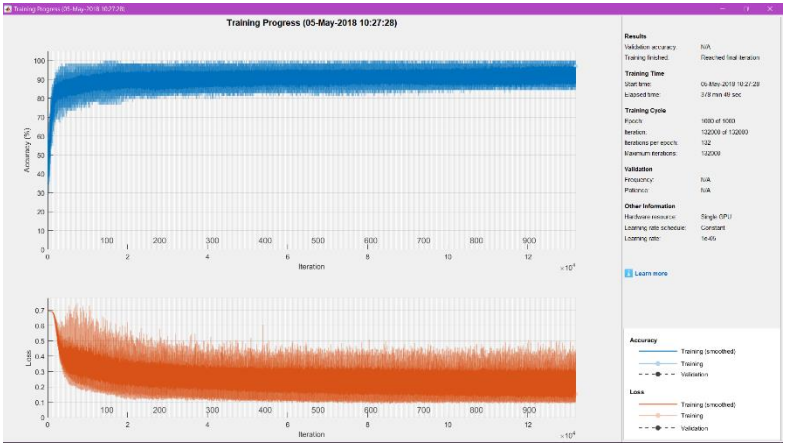
A.2.5. Level 4



Hasil training dataset 2

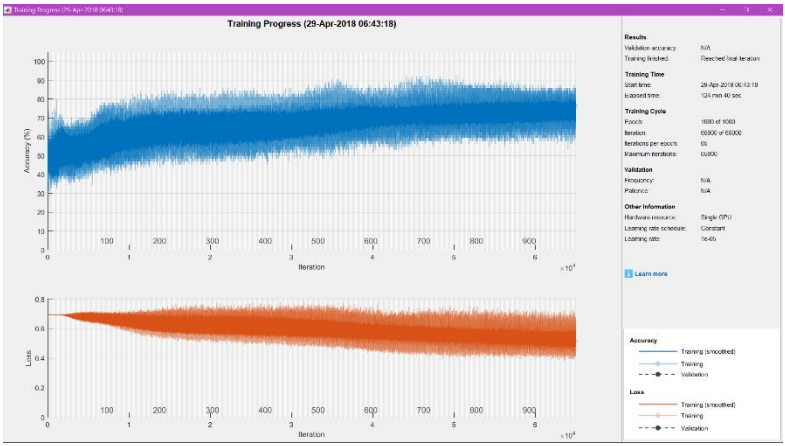


Hasil training dataset 3A

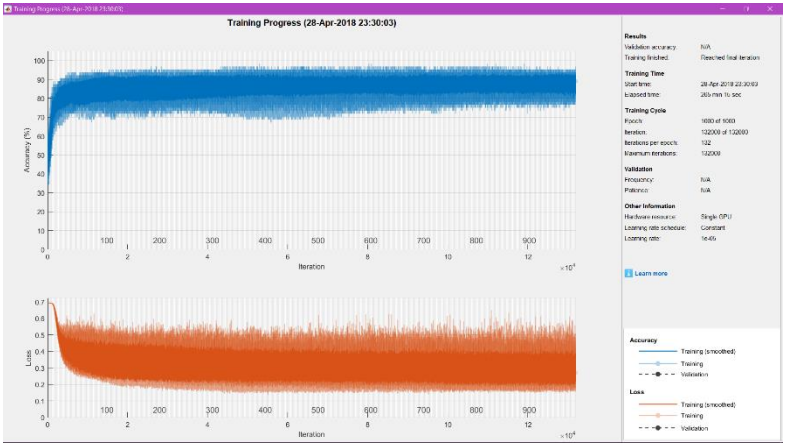


Hasil training dataset 3B

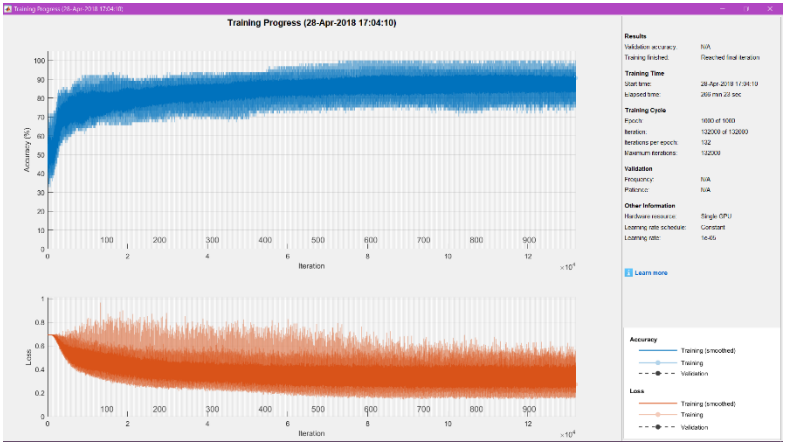
A.2.6. Level 5



Hasil training dataset 2



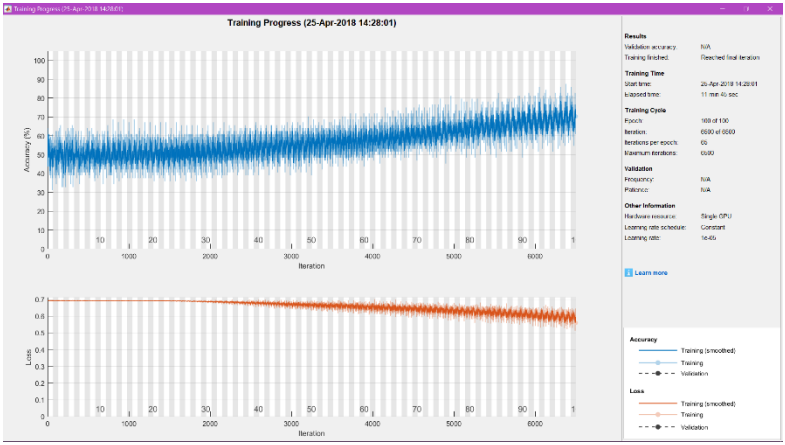
Hasil training dataset 3A



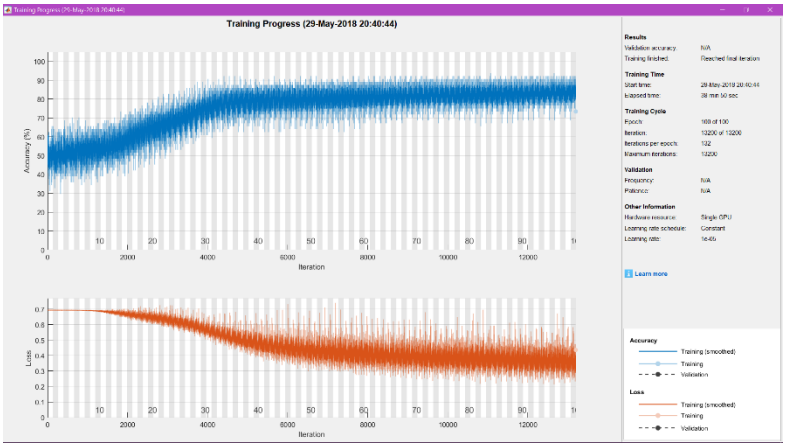
Hasil training dataset 3B

A.3. Hasil Training scenario uji coba jumlah epoch

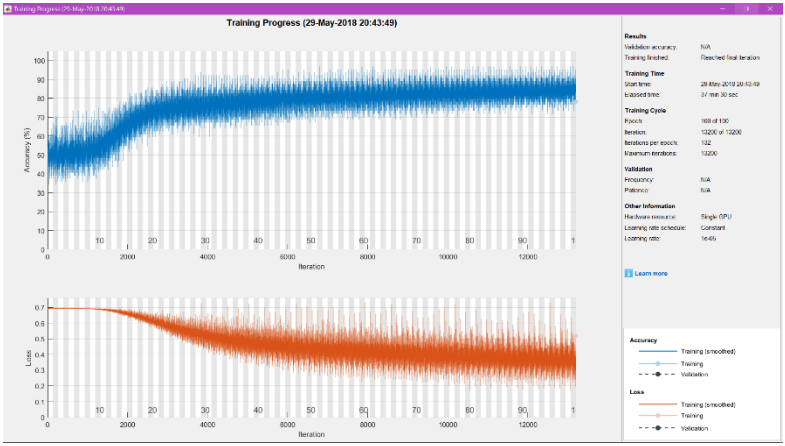
A.3.1. 100 epoch



Hasil training dataset 2

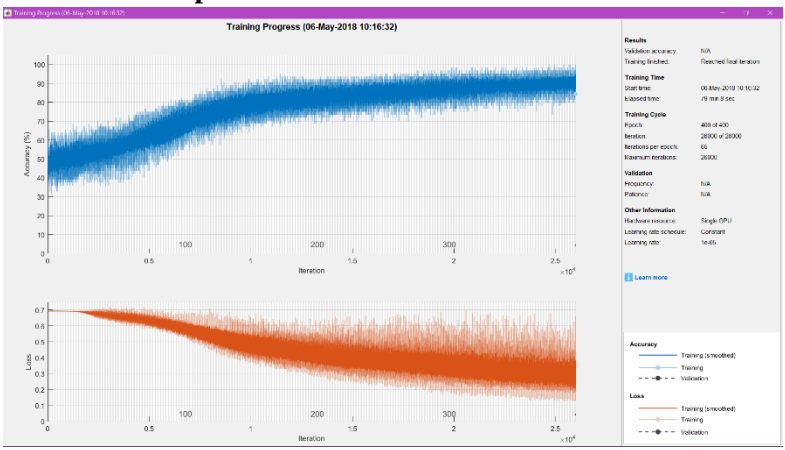


Hasil training dataset 3A

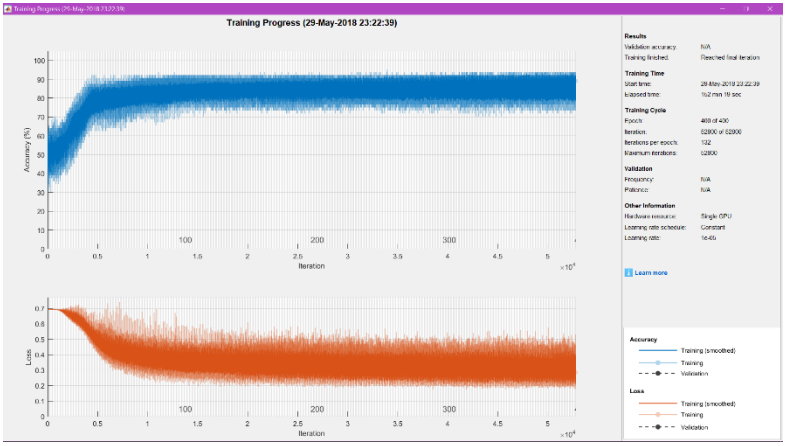


Hasil training dataset 3B

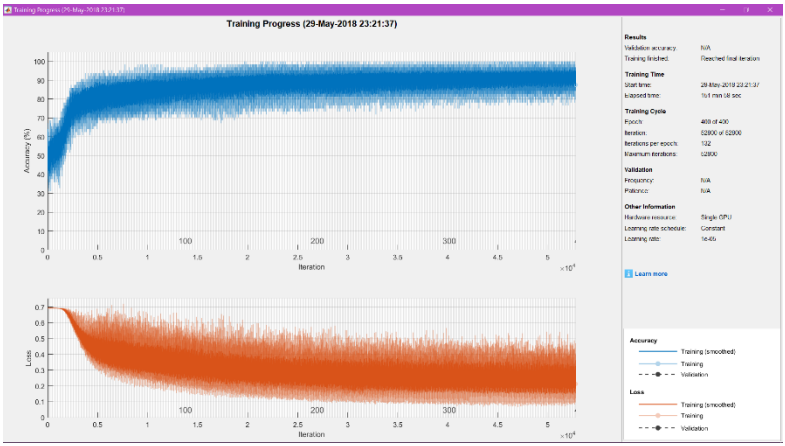
A.3.2. 400 epoch



Hasil training dataset 2

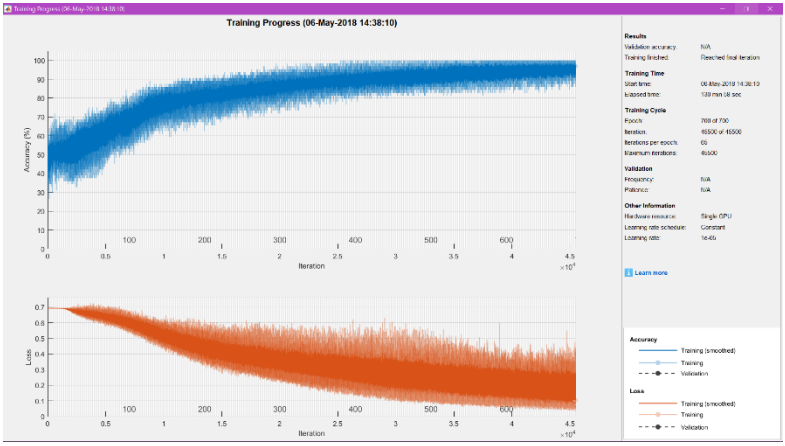


Hasil training dataset 3A

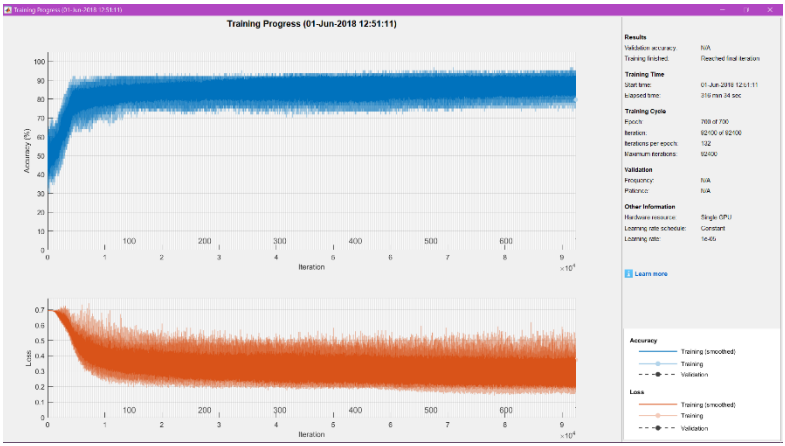


Hasil training dataset 3B

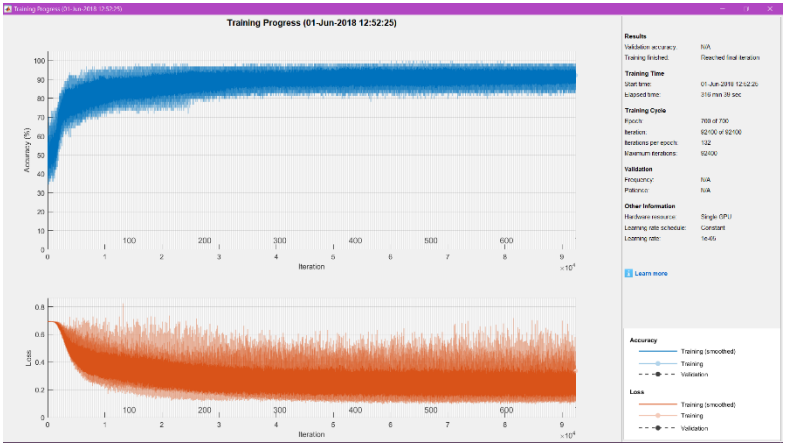
A.3.3. 700 epoch



Hasil training dataset 2

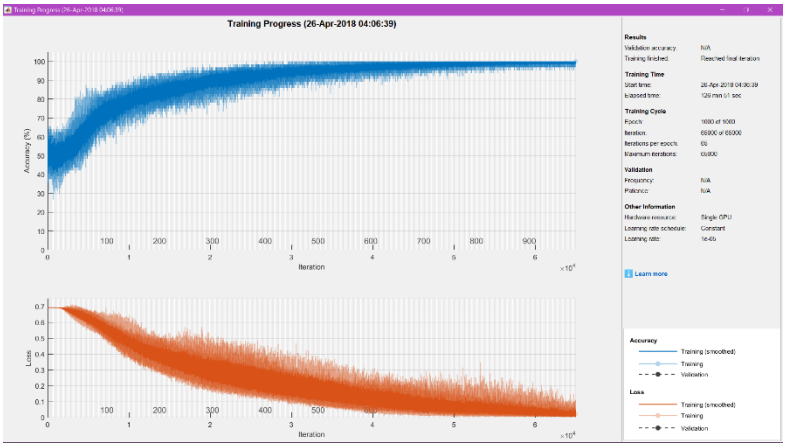


Hasil training dataset 3A

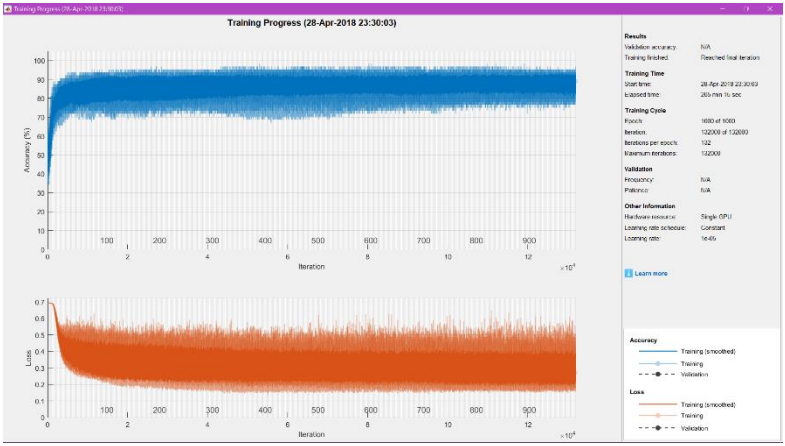


Hasil training dataset 3B

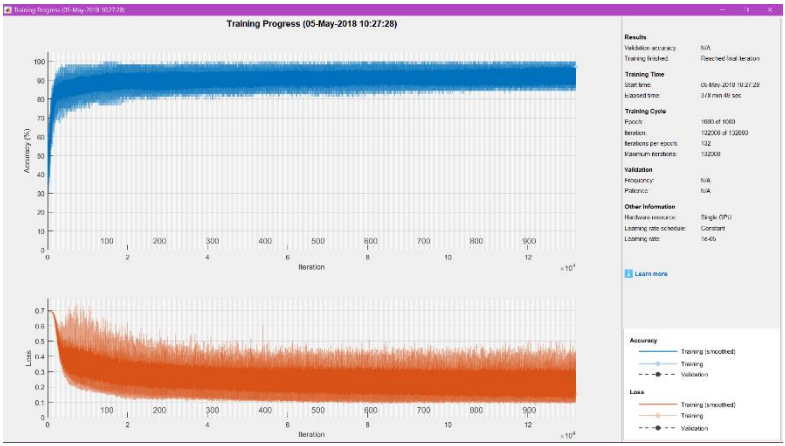
A.3.4. 1000 epoch



Hasil training dataset 2



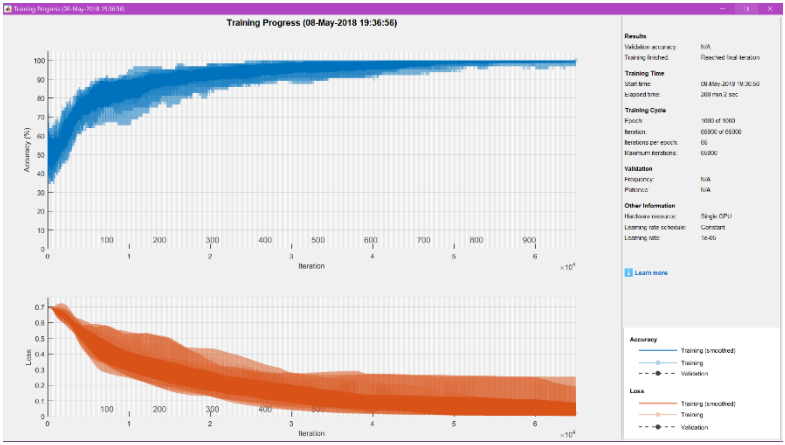
Hasil training dataset 3A



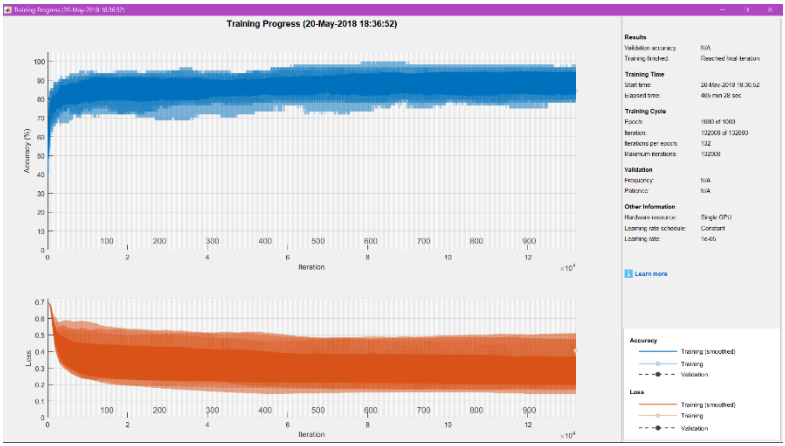
Hasil training dataset 3B

A.4. Hasil Training scenario uji coba Dropout probability

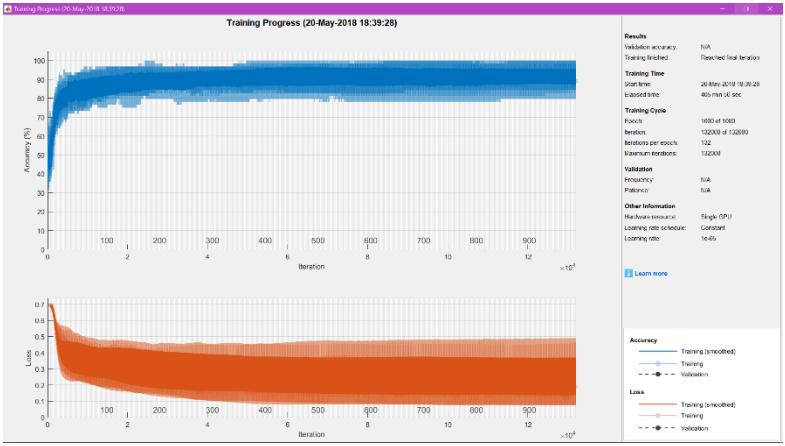
A.4.1. 0



Hasil training dataset 2

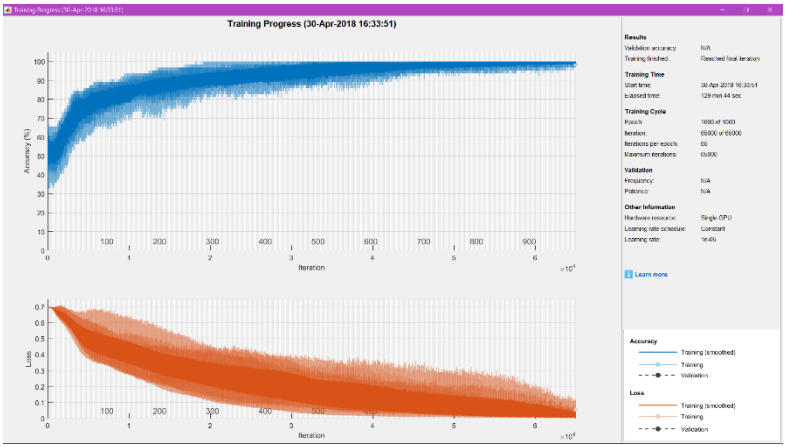


Hasil training dataset 3A

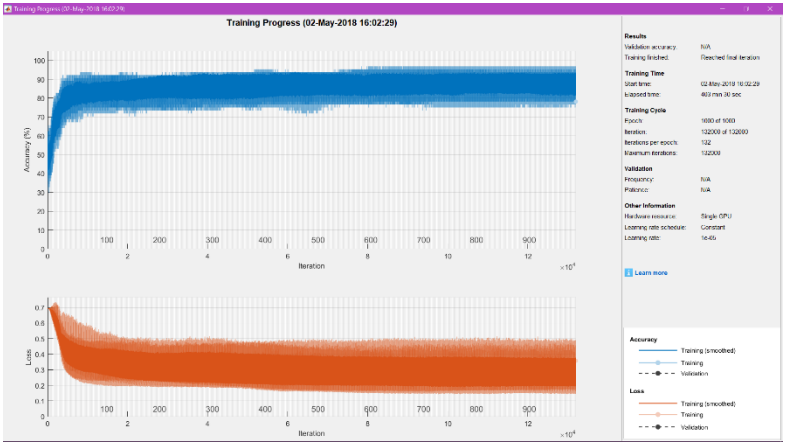


Hasil training dataset 3B

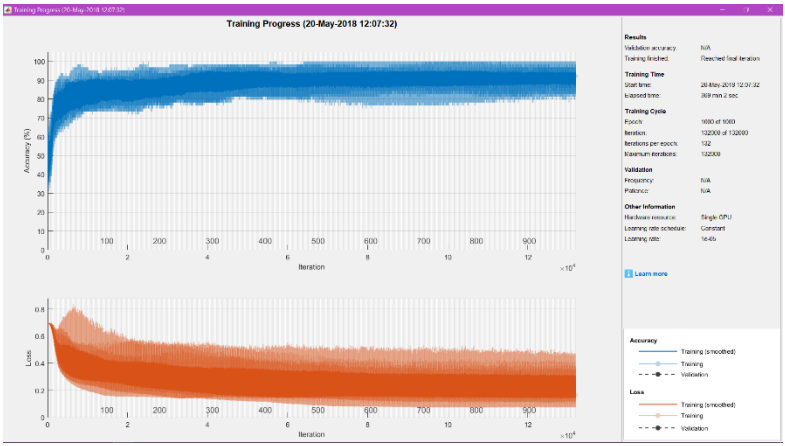
A.4.2. 0.2



Hasil training dataset 2

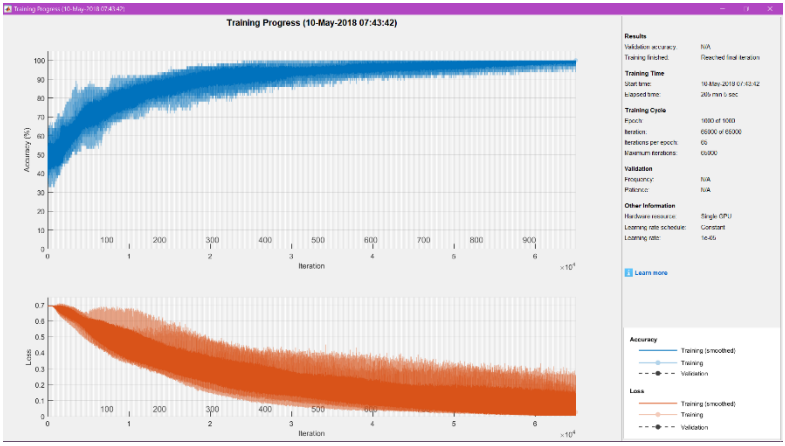


Hasil training dataset 3A

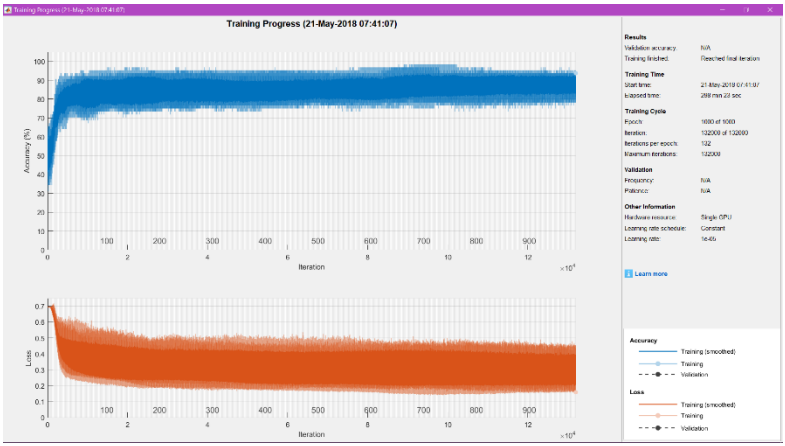


Hasil training dataset 3B

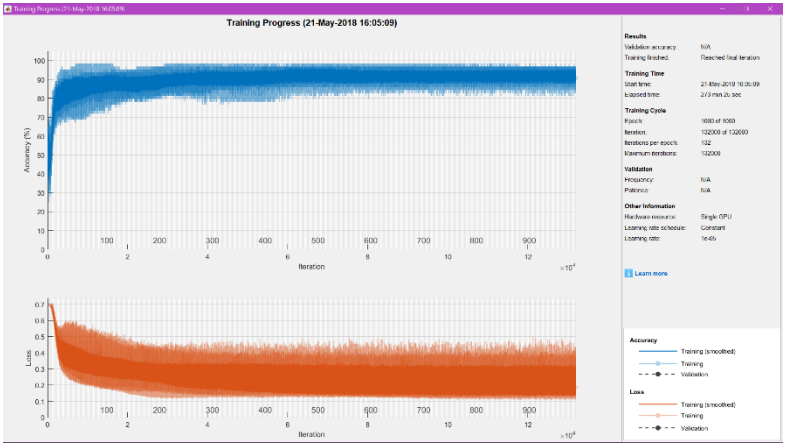
A.4.3. 0.4



Hasil training dataset 2

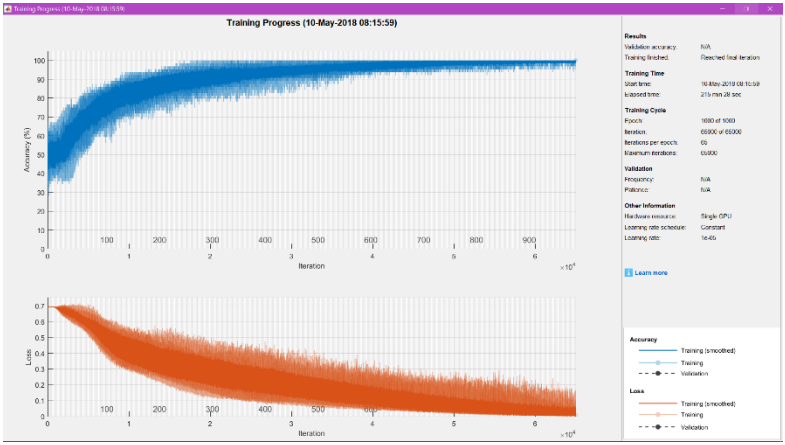


Hasil training dataset 3A

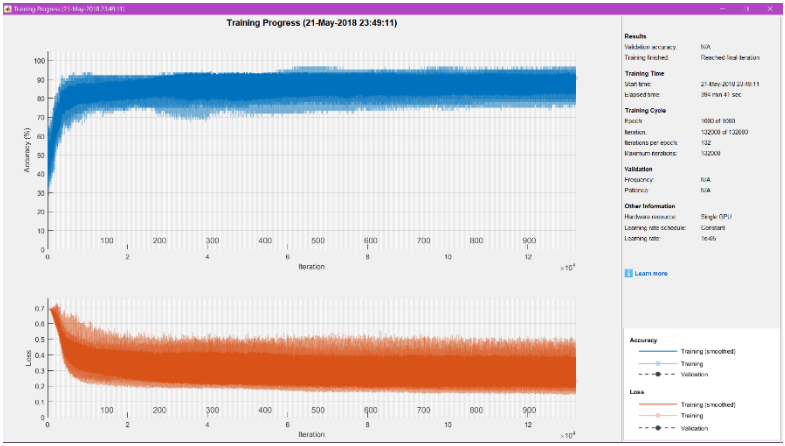


Hasil training dataset 3B

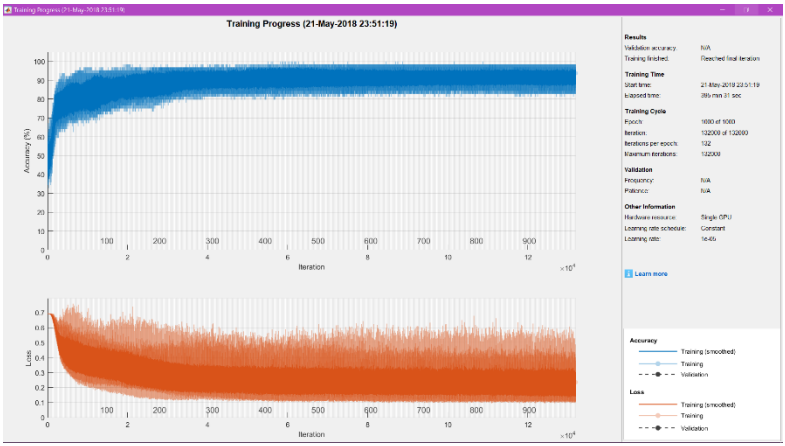
A.4.4. 0.6



Hasil training dataset 2

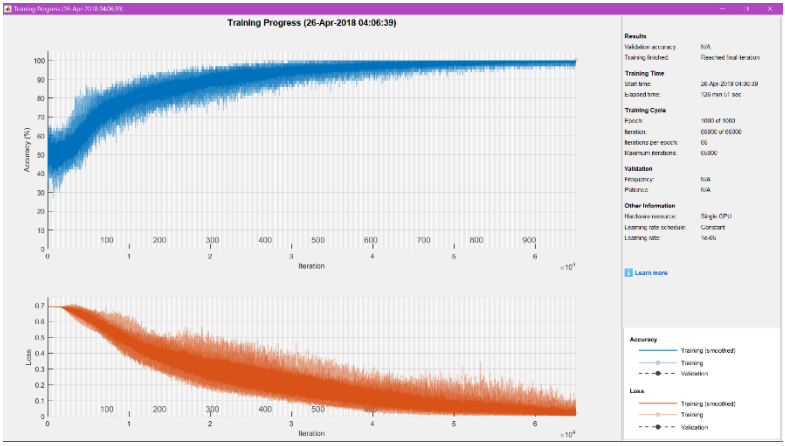


Hasil training dataset 3A

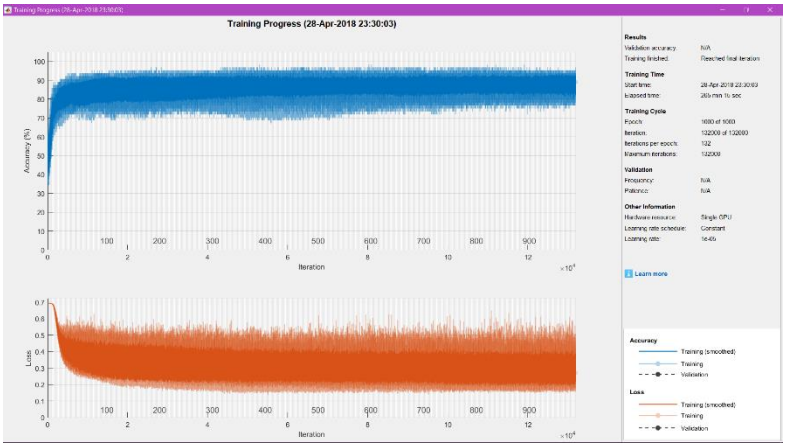


Hasil training dataset 3B

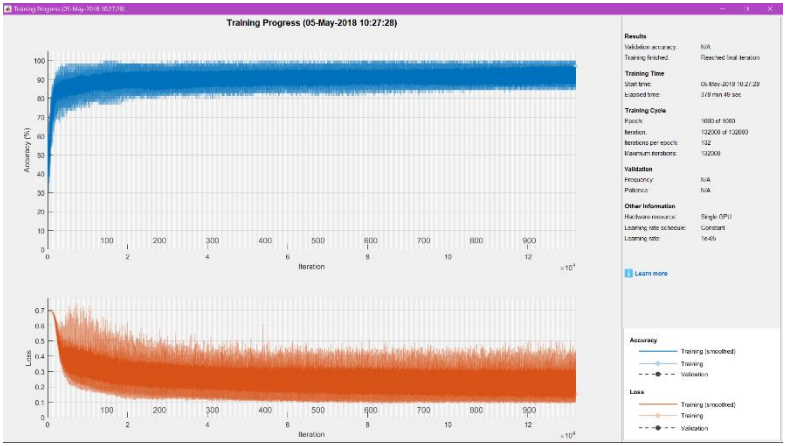
A.4.5. 0.8



Hasil training dataset 2



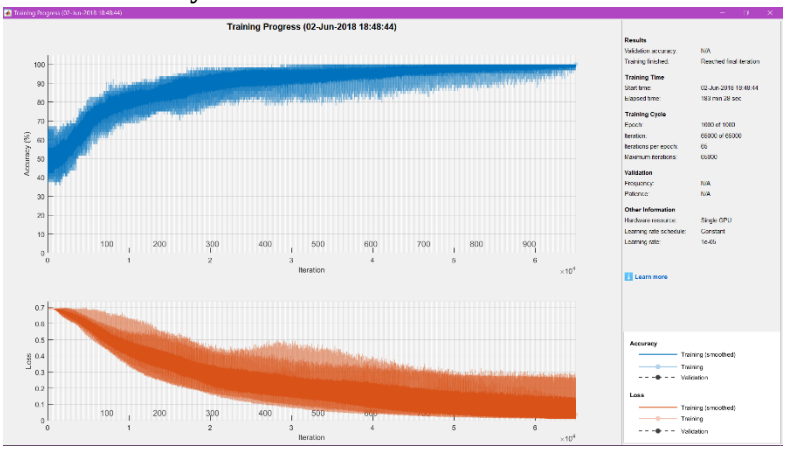
Hasil training dataset 3A



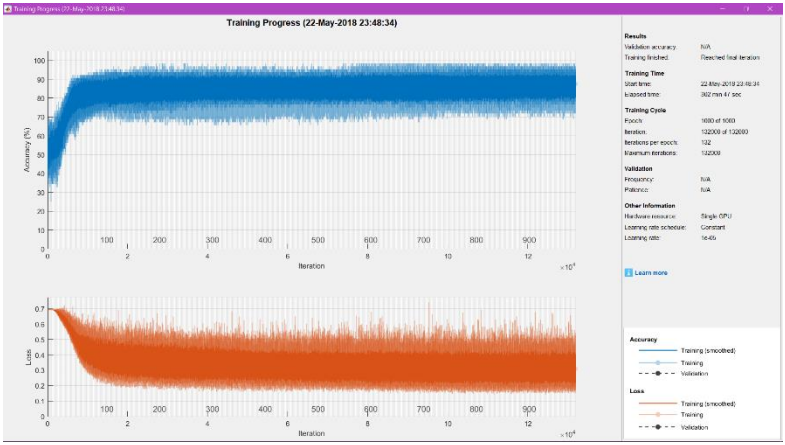
Hasil training dataset 3B

A.5. Hasil Training scenario uji coba Fully Connected Layer

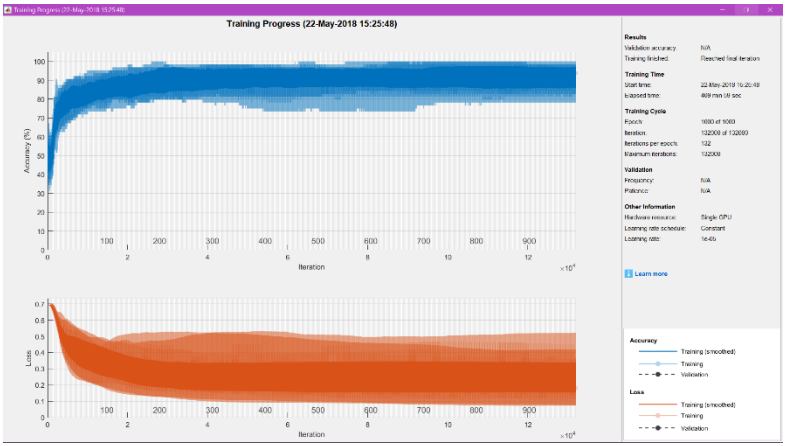
A.5.1. 64 layer



Hasil training dataset 2

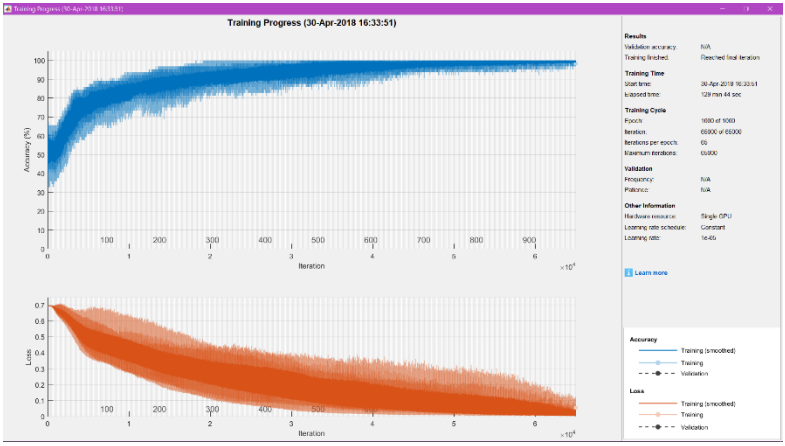


Hasil training dataset 3A

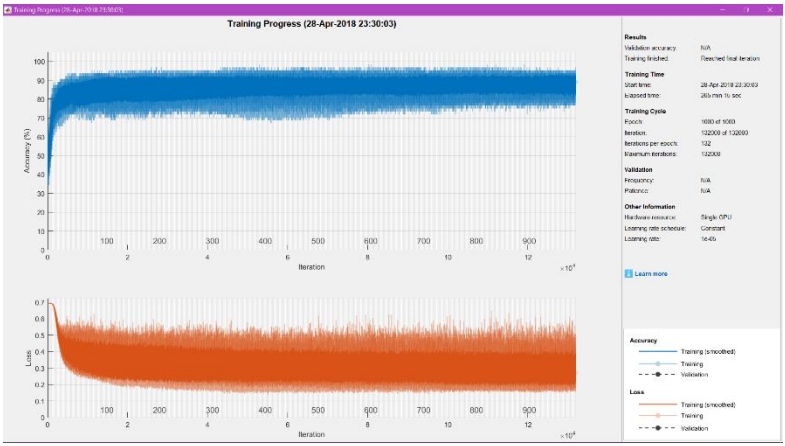


Hasil training dataset 3B

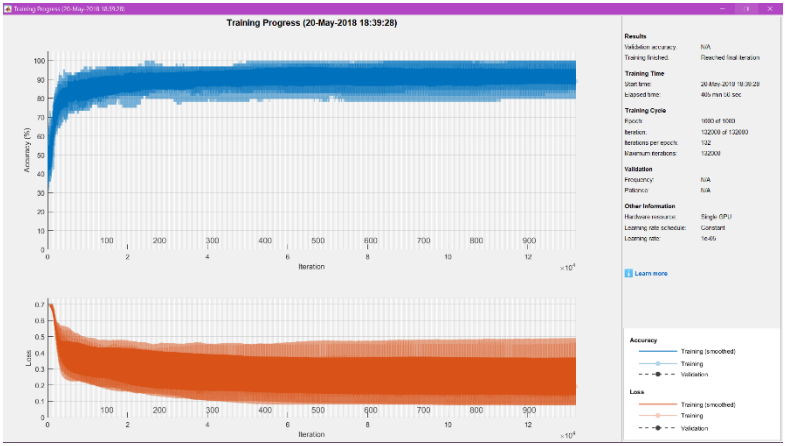
A.5.2. 128 layer



Hasil training dataset 2

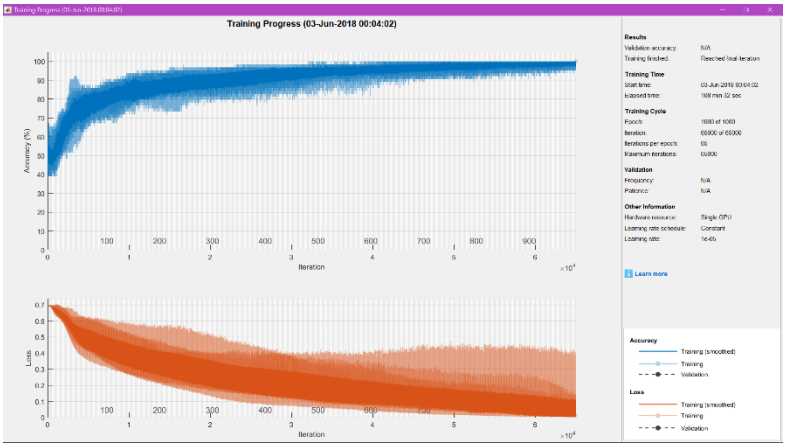


Hasil training dataset 3A

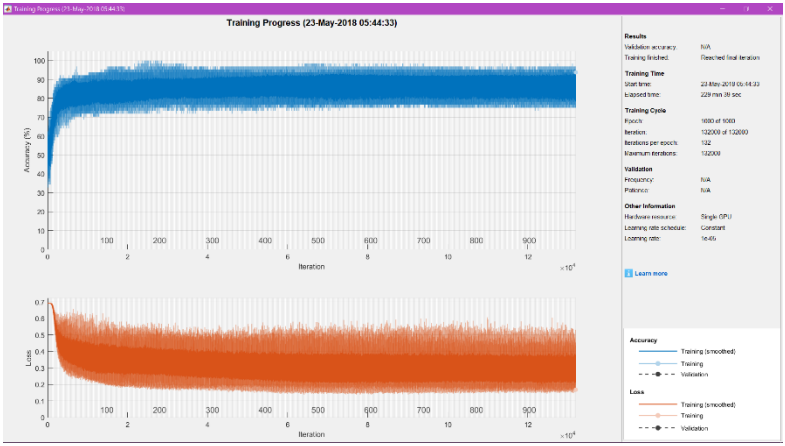


Hasil training dataset 3B

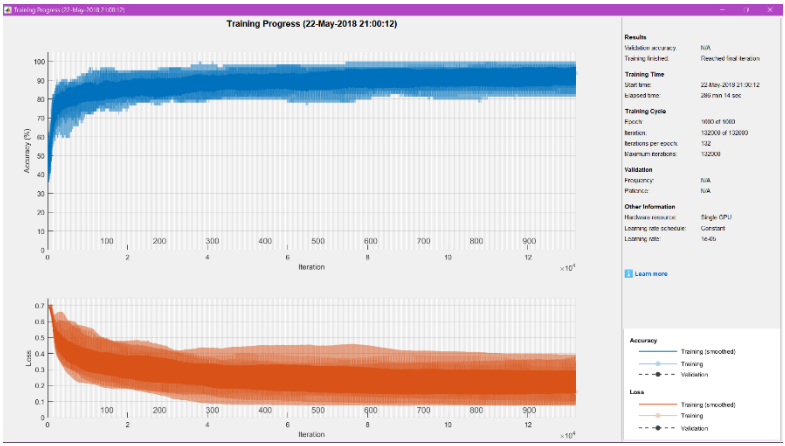
A.5.3. 192 layer



Hasil training dataset 2

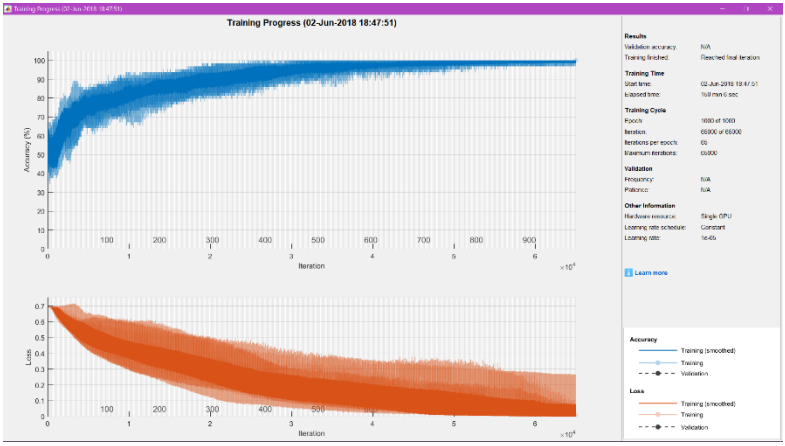


Hasil training dataset 3A

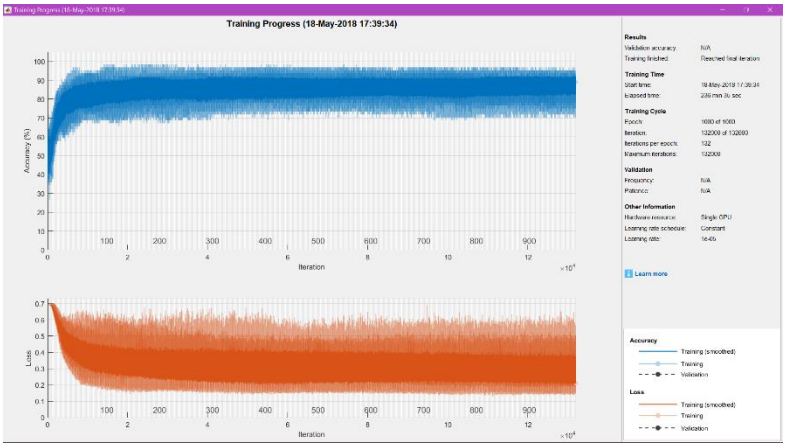


Hasil training dataset 3B

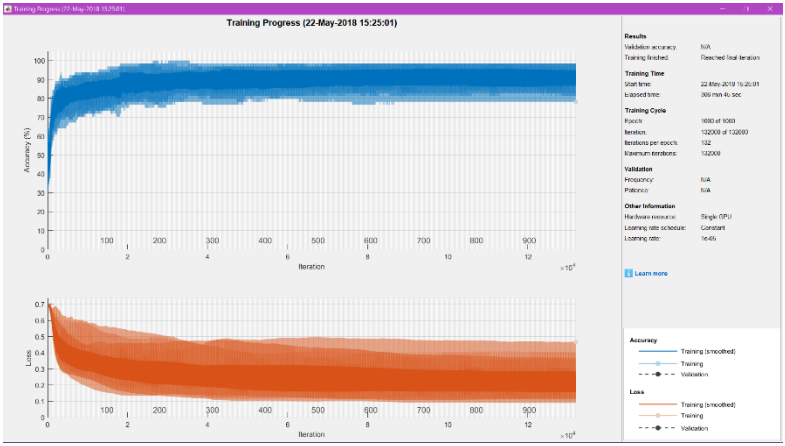
A.5.4. 256 layer



Hasil training dataset 2



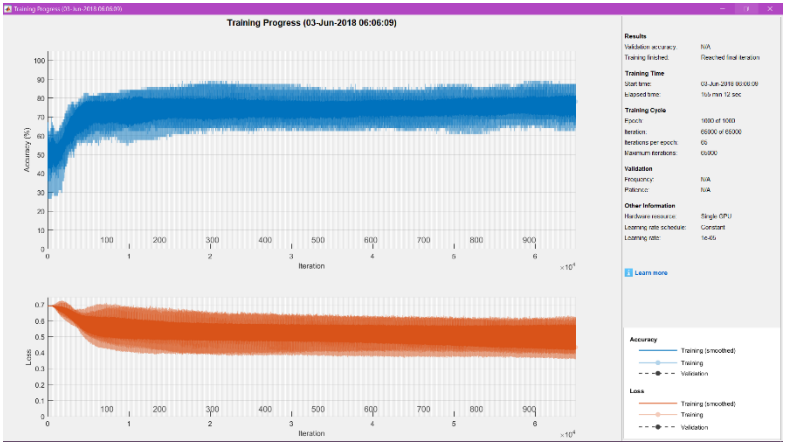
Hasil training dataset 3A



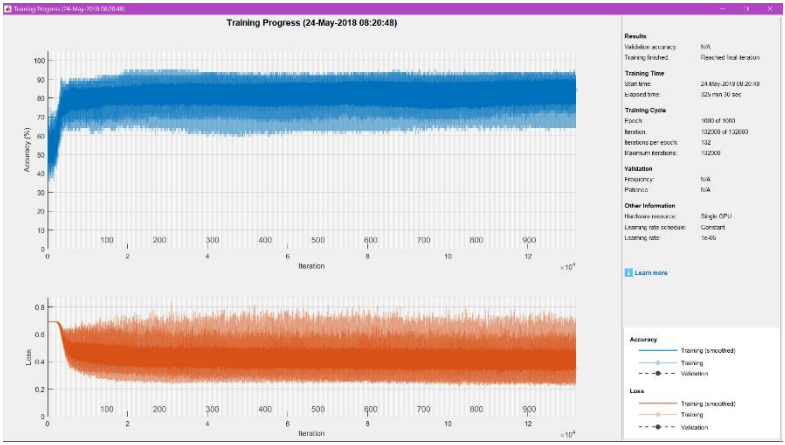
Hasil training dataset 3B

A.6. Hasil Training scenario uji coba Convolutional Layer

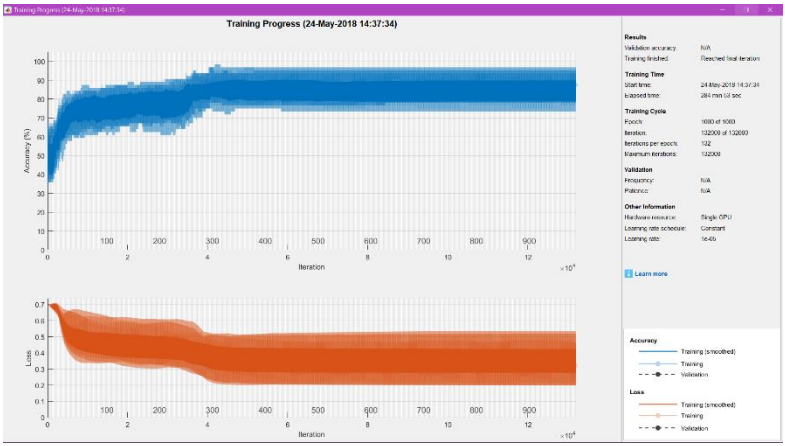
A.6.1. 1 layer



Hasil training dataset 2

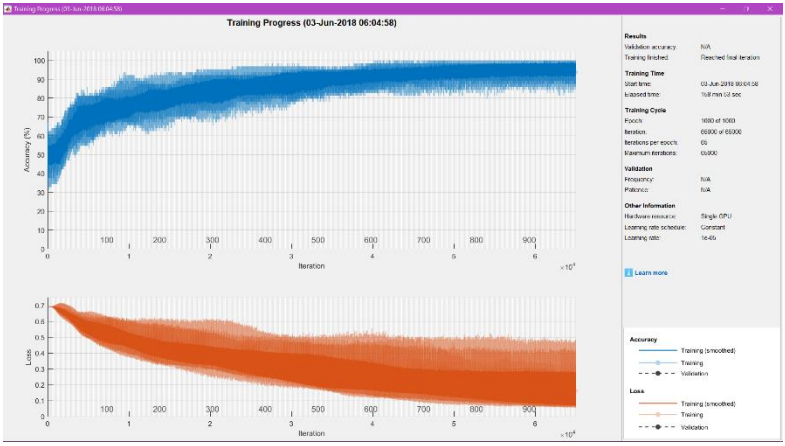


Hasil training dataset 3A

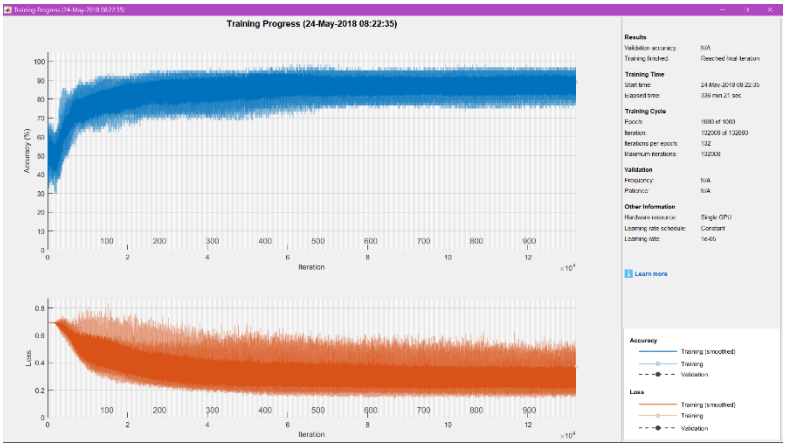


Hasil training dataset 3B

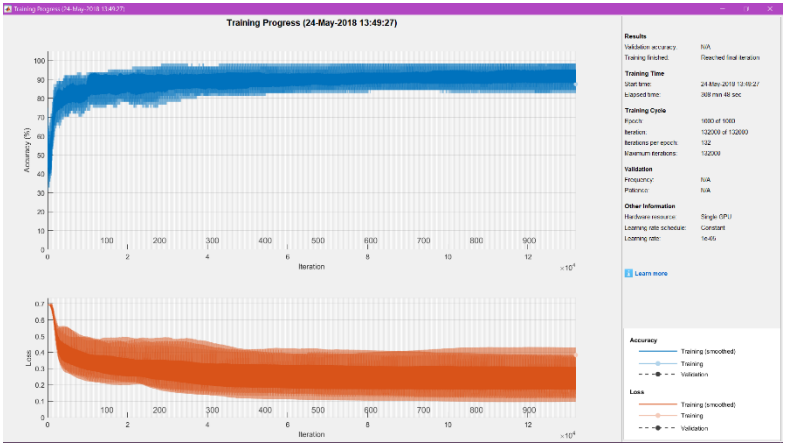
A.6.2. 8 layer



Hasil training dataset 2

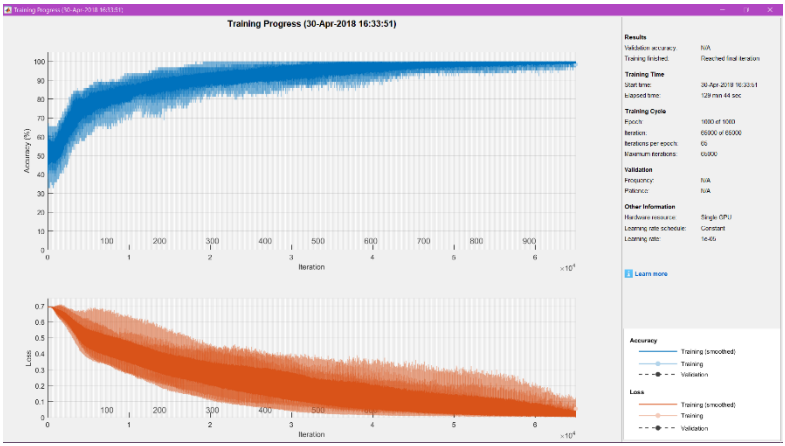


Hasil training dataset 3A

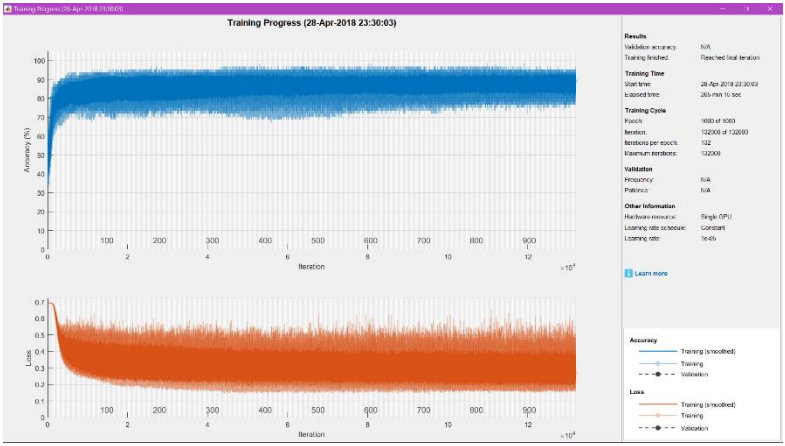


Hasil training dataset 3B

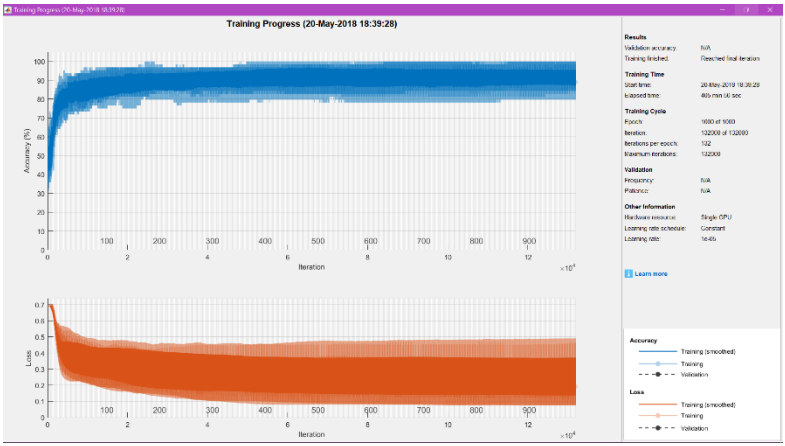
A.6.3. 16 layer



Hasil training dataset 2



Hasil training dataset 3A



Hasil training dataset 3B

B. Confusion matrix

B.1. Confusion matrix scenario uji coba averaging

Dataset	Averaging	TP	FN	FP	TN
2	1 sinyal	456	474	2200	2450
	2 sinyal	275	190	1045	1280
	3 sinyal	194	116	482	1068
3A	1 sinyal	1742	1258	5035	9965
	2 sinyal	884	616	2115	5385
	3 sinyal	527	473	689	4311
3B	1 sinyal	1807	1193	3515	11485
	2 sinyal	945	555	1508	5992
	3 sinyal	605	395	654	4346

B.2. Confusion matrix scenario uji coba penghalusan data

Dataset	Inverse HWT	TP	FN	FP	TN
2	level 0	194	116	482	1068
	level 1	197	113	510	1040
	level 2	208	102	576	974
	level 3	210	100	707	843
	level 4	191	119	549	1001
	level 5	231	79	741	809
3A	level 0	527	473	689	4311
	level 1	555	445	987	4013
	level 2	651	349	1045	3955
	level 3	657	343	750	4250
	level 4	710	290	1001	3999
	level 5	686	314	644	4356
3B	level 0	605	395	654	4346
	level 1	628	372	749	4251
	level 2	677	323	566	4434
	level 3	726	274	839	4161
	level 4	711	289	513	4487
	level 5	743	257	917	4083

B.3. Confusion matrix scenario uji coba jumlah epoch

Dataset	Jumlah Epoch	TP	FN	FP	TN
2	100 epoch	171	139	595	955
	400 epoch	196	114	614	936
	700 epoch	173	137	469	1081
	1000 epoch	194	116	482	1068
3A	100 epoch	782	218	1274	3726
	400 epoch	750	250	1091	3909
	700 epoch	733	267	1087	3913
	1000 epoch	686	314	644	4356
3B	100 epoch	777	223	1163	3837
	400 epoch	760	240	800	4200
	700 epoch	756	244	810	4190
	1000 epoch	711	289	513	4487

B.4. Confusion matrix scenario uji coba Dropout probability

Dataset	Dropout Probability	TP	FN	FP	TN
2	0	198	112	452	1098
	0.2	170	140	403	1147
	0.4	190	120	473	1077
	0.6	173	137	407	1143
	0.8	194	116	482	1068
3A	0	692	308	710	4290
	0.2	732	268	1066	3934
	0.4	721	279	1022	3978
	0.6	728	272	1070	3930
	0.8	686	314	644	4356
3B	0	711	289	500	4500
	0.2	731	269	584	4416
	0.4	752	248	811	4189
	0.6	757	243	802	4198
	0.8	711	289	513	4487

B.5. Confusion matrix scenario uji coba Fully Connected Layer

Dataset	Jumlah Kernel	TP	FN	FP	TN
2	64 kernel	178	132	440	1110
	128 kernel	170	140	403	1147
	192 kernel	207	103	503	1047
	256 kernel	179	131	459	1091
3A	64 kernel	783	217	1110	3890
	128 kernel	686	314	644	4356
	192 kernel	733	267	1129	3871
	256 kernel	752	248	1094	3906
3B	64 kernel	750	250	815	4185
	128 kernel	711	289	500	4500
	192 kernel	729	271	579	4421
	256 kernel	715	285	612	4388

B.6. Confusion matrix scenario uji coba Convolutional Layer

Dataset	Jumlah kernel	TP	FN	FP	TN
2	1 kernel	280	30	933	617
	8 kernel	217	93	569	981
	16 kernel	170	140	403	1147
3A	1 kernel	759	241	997	4003
	8 kernel	784	216	1081	3919
	16 kernel	686	314	644	4356
3B	1 kernel	799	201	746	4254
	8 kernel	707	293	616	4384
	16 kernel	799	201	746	4254

BIODATA PENULIS

Adenuar Purnomo, lahir pada tanggal 14 Januari 1997 di Pasuruan. Penulis merupakan seorang mahasiswa yang sedang menempuh studi di Departemen Teknik Informatika Institut Teknologi Sepuluh Nopember. Memiliki beberapa hobi antara lain membaca komik dan tidur. Selama menempuh pendidikan di kampus, penulis juga aktif dalam organisasi kemahasiswaan, antara lain Koordinator Departemen Kerohanian Tim Pembina Kerohanian Buddha ITS pada tahun ke-2, serta Ketua Harian Tim Pembina Kerohanian Buddha ITS pada tahun ke-3.

Email: adenuarp@gmail.com