PENGENALAN POLA GETARAN REL KERETA API DENGAN METODE MFCC DAN JARINGAN SARAF TIRUAN

SKRIPSI

Disusun Untuk Melengkapi Syarat-Syarat Guna Memperoleh Gelar Sarjana Sains

BUDIMAN SIMBOLON

3225122045

PROGRAM STUDI FISIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS NEGERI JAKARTA

2017

PERSETUJUAN PANITIA UJIAN SKRIPSI

Pengenalan Pola Getaran Rel Kereta Api dengan Metode MFCC dan Jaringan Saraf Tiruan

Nama : Budiman Simbolon

No. Reg. : 3225122045

	Nama	Tanda tangan	Tanggal
Penanggung Jawab			
Dekan	: Prof. Dr. Suyono, M.Si NIP. 196712181993031005	.....	.....
Wakil Penanggung Jawab			
Wakil Dekan I	: Dr. Muktiningsih, M.Si NIP. 196405111989032001	.....	.....
Ketua	: Dr. Sunaryo, M.Si NIP. 195503031987031002	.....	.....
Sekretaris	: Umiatin, M.Si NIP. 197901042006042001	.....	.....
Anggota			
Pembimbing I	: Dr.rer.nat Bambang Heru Iswanto, M.Si NIP. 196804011994031002	.....	.....
Pembimbing II	: Dr. Widyaningrum Indrasari, M.Si NIP. 197705102006042001	.....	.....
Penguji Ahli	: Riser Fahdiran, M.Si NIP. 198307172009121008	.....	.....

Dinyatakan lulus ujian skripsi tanggal: 11 Agustus 2017

SURAT PERNYATAAN KEASLIAN SKRIPSI

Dengan ini, saya yang bertanda tangan di bawah ini, mahasiswa Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Negeri Jakarta:

Nama : Budiman Simbolon
No.Reg. : 3225122045
Program Studi : Fisika

Menyatakan bahwa skripsi yang saya buatdengan judul **"Pengenalan Pola Getaran Rel Kereta Api dengan Metode MFCC dan Jaringan Saraf Tiruan"**, adalah :

1. Dibuat dan diselesaikan oleh saya sendiri, berdasarkan data yang diperoleh dari hasil penelitian pada bulan Maret-Juli 2017.
2. Bukan merupakan duplikat skripsi yang pernah dibuat oleh orang lain atau jiplakan karya tulis orang lain dan bukan terjemahan karya tulis orang lain.

Pernyataan ini saya buat dengan sesungguhnya dan saya bersedia menanggung segala akibat yang timbul jika pernyataan saya ini tidak benar.

Jakarta, Agustus 2017
Yang membuat pernyataan

Budiman Simbolon
NIM 3225122045

ABSTRAK

Budiman Simbolon. Pengenalan Pola Getaran Rel Kereta Api dengan Metode MFCC dan Jaringan Saraf Tiruan. Skripsi. Jakarta : Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Negeri Jakarta, 2017.

Permasalahan yang sering muncul pada perkeretaapian saat ini adalah tingginya tingkat kecelakaan pada perlintasan kereta api. Penyebab tingginya kecelakaan merupakan akumulasi dari beberapa faktor, diantaranya masalah regulasi, manajemen, kondisi prasarana dan sarana, sumber daya manusia dan lain-lain. Sebagian besar penyebab terjadinya kecelakaan dikarenakan tidak adanya palang pintu perlintasan, kegagalan menutup palang pintu pada saat yang dibutuhkan serta kegagalan operator dalam memerintahkan penutupan palang pintu (*human error*). Untuk meminimalisir *human error* tersebut dibutuhkan suatu sistem alternatif untuk mendeteksi kedatangan kereta api. Salah satu diantaranya adalah identifikasi kereta berdasarkan getaran. Sistem ini menggunakan metode MFCC sebagai ekstraksi ciri dan jaringan saraf tiruan sebagai klasifikasi. Penelitian ini menggunakan data sebanyak 160 data getaran kereta dan 160 getaran bukan kereta. Data dikelompokkan menjadi kelompok data latih dan kelompok data uji dengan perbandingan 60:40. Data latih kemudian diekstrak dengan MFFC untuk mendapatkan 13 koefisien MFCC. Pelatihan jaringan dilakukan dengan variasi jumlah neuron mulai dari 10, 20, 30, 40, 50, 60, 70, 80, 90 dan 100. Jaringan saraf tiruan yang digunakan adalah *multi layer feed-forward backpropagation.* Hasil dari simulasi data uji diperoleh akurasi jaringan saraf tiruan terbaik sebesar 95.83% dengan jumlah neuron 10.

Kata Kunci: *MFCC, Jaringan Jaraf Tiruan, Data Latih, Data Uji, Pola, Ekstraksi Ciri*

ABSTRACT

Budiman Simbolon. *Identification of Railway Vibration Patterns with MFCC Methods and Artificial Neural Networks. Minithesis. Jakarta: Faculty of Mathematics and Natural Sciences, Jakarta State University, 2017.*

The problem that often arises in the railways today is the high level of accidents at railway crossings. The cause of the high accidents is the accumulation of several factors, including the problem of regulation, management, condition of infrastructure and facilities, human resources and others. Most of the causes of accidents are due to the lack of crossing gates, failure to close the door at the time required and the failure of the operator in ordering the closing of the door (human error). To minimize human error is needed an alternative system to detect the arrival of the train. One of them is train identification based on vibration. This system uses the MFCC method as feature extraction and neural networks as a classification. This research uses 160 data vibration of train and 160 vibration not train. Data are grouped into groups of training data and group of test data with 60:40 ratio. The training data is then extracted with MFFC to obtain 13 MFCC coefficients. Network training was performed by varying the number of neurons ranging from 10, 20, 30, 40, 50, 60, 70, 80, 90 and 100. The artificial neural network used was multi layer feed-forward backpropagation. The result of simulation of test data obtained by the best artificial neural network accuracy 95.83% with the number of neuron 10.

Keywords: *MFCC, Artificial Neural Network, Train Data, Test Data, Patterns, Feature Extraction*

KATA PENGANTAR

Segala puji dan syukur saya panjatkan kepada Tuhan Yang Maha Esa atas segala kasih dan karunia-Nya maka saya dapat menyelesaikan skripsi ini. Tanpa pertolongan Dia mungkin saya tidak sanggup menyelesaikan skripsi ini dengan baik.

Skripsi ini memuat tentang "Pengenalan Pola Getaran Rel Kereta Api dengan Metode MFCC dan Jaringan Saraf Tiruan' dan sengaja dipilih karena dalam aplikasinya dalam kehidupan sehari-hari dapat dikembangkan menjadi sistem buka tutup palang pintu perlintasan kereta api yang dapat membantu tugas operator pada perlintasan kereta api. Dengan demikian *human error* yang menjadi salah satu faktor penyebab kecelakaan pada pintu perlintasan kereta api dapat berkurang.

Untuk menyelesaikan skripsi ini, saya tidak lepas dari bantuan dan kerjasama dari berbagai pihak. Oleh karenanya, saya ingin mengucapkan terimakasih yang sebesar-besarnya kepada:

1. Dr.rer.nat. Bambang Heru Iswanto, M.Si selaku Dosen Pembimbing I atas segala ide, bimbingan, arahan, kritik dan saran kepada saya.
2. Dr. Widyaningrum Indrasari, M.Si selaku Dosen Pembimbing II atas segala bimbingan, motivasi, ide, nasehat, kritik dan saran kepada saya.
3. Keluarga besar saya baik orang tua (Uler Simbolon dan Nurmina Marbun) dan saudara/i saya (Darlan Simbolon, Ramses Simbolon, Doras Malina Simbolon, Berliana R Simbolon) atas segala doa, nasehat, dukungan baik moral maupun materil kepada saya.
4. Sahabat-sahabat saya yang tergabung dalam grup 'Lima Sekawan' (Indra Permana, Ribka Uli Simbolon, Susan Travel Sinaga, Vina Agustina Gultom) atas segala doa, motivasi, nasehat, kritik dan saran kepada saya.
5. Semua keluarga dan teman-teman yang terlibat dalam pembuatan skripsi ini baik secara langsung maupun tidak langsung.

Segala upaya telah saya lakukan demi kesempurnaan skripsi ini. Namun sebagai manusia biasa saya tidak luput dari salah dan lupa. Oleh karena itu,

saran dan kritik senantiasa saya harapkan sebagai pembelajaran untuk masa yang akan datang. Akhir kata semoga skripsi ini bermanfaat bagi kita semua.

Jakarta, Juli 2017

Budiman Simbolon

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB I
PENDAHULUAN

A. Latar Belakang

Kereta api merupakan salah satu sarana transportasi masal yang dapat mengangkut banyak penumpang sekaligus. Saat ini kereta api telah tersebar hampir di seluruh penjuru Indonesia, dan masih menjadi salah satu transportasi favorit masyarakat Indonesia, karena biaya yang terjangkau dan cukup efisien waktu.

Permasalahan yang kerap kali muncul pada perkeretaapian saat ini adalah tingginya tingkat kecelakaan terutama pada perlintasan rel kereta api. Penyebab tingginya kecelakaan merupakan akumulasi dari beberapa faktor, diantaranya masalah regulasi, manajemen, kondisi prasarana dan sarana, Sumber Daya Manusia (SDM) dan lain-lain. Sebagian besar penyebab terjadinya kecelakaan dikarenakan tidak adanya pintu perlintasan, kegagalan menutup palang pintu pada saat yang dibutuhkan serta kegagalan operator dalam memerintahkan penutupan palang pintu (*human error*). Untuk meminimalisir *human error* tersebut maka dibutuhkan suatu sistem alternatif untuk mendeteksi kedatangan kereta api.

Banyak penelitian yang berhubungan dengan sistem perkeretaapian di Indonesia. Beberapa penelitian yang berhubungan dengan sistem perkeretaapian di Indonesia yaitu palang pintu otomatis dengan menggunakan sensor photodioda yang dilakukan oleh Havi Cahyanto, palang pintu kereta otomatis dengan sistem pewaktuan dalam pendeteksiannya oleh Fernando Dwi Agustia, otomatisasi palang pintu kereta api dengan *motion detection* oleh Muhammad Fayyadh dan otomatisasi palang pintu dengan sensor ultrasonic oleh Dian Eka Ratnawati. Tapi dari semua itu masih ada titik lemah yang tentunya bisa digunakan oleh pihak-pihak yang tidak bertanggung jawab untuk kepentingan diri-sendiri atau kelompok. Salah satu kelemahan dari otomatisasi pintu perlintasan kereta api dengan menggunakan sensor ultrasonik dan sensor photodioda adalah rentan terhadap gangguan (*noise*). Sensor ini tidak akan berfungsi dengan baik jika ada

gangguan yang menghalangi sensor tersebut sehingga resiko kegagalan sistem sangat tinggi.

Untuk menanggulangi kelemahan tersebut perlu dibuat suatu sistem alternatif untuk identifikasi kereta api. Salah satu diantaranya adalah sistem identifikasi kereta api berdasarkan getaran yang dapat diimplementasikan pada pintu perlintasan kereta api untuk mendeteksi kedatangan kereta. Sistem identifikasi ini didasarkan pada anggapan bahwa getaran yang dihasilkan oleh kereta dapat dianggap sebagai salah satu ciri pembeda dengan benda lainnya.

Pada penelitian ini akan digunakan metode MFCC sebagai ekstraksi ciri yang diimplementasikan pada rekaman getaran rel kereta api. Perekaman getaran rel kereta api dilakukan dengan menggunakan sensor *accelerometer*. Hasil rekaman sensor ini kemudian diubah menjadi sinyal audio berbentuk file *.wav. Ekstraksi ciri dengan menggunakan MFCC dilakukan pada setiap file untuk mendapatkan ciri khusus dari setiap sinyal. Jaringan saraf tiruan digunakan untuk mengklasifikasi ciri dari getaran rel kereta api. Melalui penelitian ini diharapkan dapat dikembangkan suatu sistem identifikasi kereta api berdasarkan getaran sehingga dapat mengurangi *human error* sebagai salah satu penyebab kecelakaan pada perlintasan kereta api.

B. Identifikasi Masalah

Berdasarkan latar belakang diatas, maka dapat diidentifikasi masalah masalah sebagai berikut.

1. Apakah MFCC dapat diimplementasikan untuk ekstraksi ciri getaran rel kereta api?
2. Apakah Jaringan Saraf Tiruan dapat digunakan untuk identifikasi getaran rel kereta api?
3. Model Jaringan seperti apa yang dapat digunakan untuk identifikasi getaran rel kereta api?

C. Pembatasan Masalah

Permasalahan dari penelitian ini dibatasi sebagai berikut.

1. Ekstraksi ciri yang digunakan adalah *Mel-Frequency Cepstrum Coefficient* .
2. Jaringan saraf tiruan sebagai metode klasifikasi.
3. Ekstraksi ciri MFCC menggunakan koefisien 13.
4. Sensor yang digunakan adalah sensor *Accelerometer* yang ada pada telepon pintar.

D. Perumusan Masalah

Masalah yang akan dikaji dari penelitian ini, yaitu :

1. Berapa persen akurasi dari sistem pengenalan getaran kereta api dengan jaringan saraf tiruan sebagai klasifikasi?
2. Bagaimana algoritma ekstraksi ciri dapat mengekstrak ciri getaran kereta api?
3. Bagaimana algoritma klasifikasi ciri dapat mengidentifikasi getaran kereta api?

E. Tujuan Penelitian

Tujuan dari penelitian itu, yaitu :

1. Mengembangkan sistem untuk memprediksi kereta api berdasarkan getaran
2. Meneliti akurasi jaringan saraf tiruan untuk prediksi jenis kereta api berdasarkan getaran

F. Manfaat Penelitian

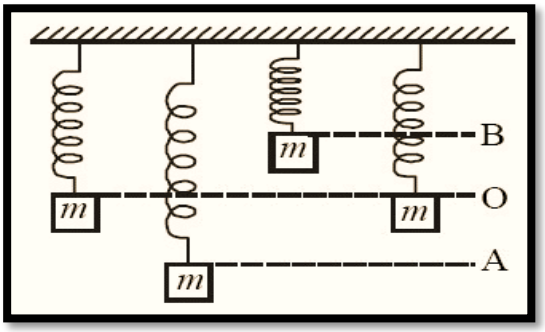
Manfaat dari penelitian ini, yaitu :

1. Untuk menentukan jenis kereta atau bukan yang melintas di rel kereta api
2. Sebagai sistem alternatif untuk mendeteksi kedatangan kereta api sehingga dapat menekan tingkat kecelakaan yang diakibatkan oleh *human error.*

BAB II
KAJIAN TEORI

A. Getaran Mekanik

Getaran adalah gerakan yang teratur dari benda atau media dengan arah bolak-balik dari kedudukan keseimbangan. Gerakan dapat berupa benturan yang berulang secara kontinu atau dengan kata lain dapat juga berupa gerakan tidak beraturan atau acak.



Gambar 2. 1 Ilustrasi Getaran

Ilustrasi sederhana dari getaran adalah pegas dengan sebuah beban, seperti yang ditunjukkan pada gambar 1. Pada posisi netral (Posisi O) maka pegas akan merenggang untuk mengimbangi beban. Jika pegas diberi gaya seketika dengan menarik beban ke posisi bawah dan kemudian langsung dilepaskan akan membuat beban bergerak bolak-balik dari posisi bawah menuju posisi atas dan seterusnya guna mengimbangi gaya seketika tersebut. Gerakan dari beban yang bergerak bola-balik akan membentuk sebuah gelombang dengan domain waktu.

Ada beberapa karakteristik dalam getaran yaitu:

a) Frekuensi getaran

Frekuensi dalam getaran selalu berhubungan dengan periode. Dinyatakan dalam persamaan :

$$f = \frac{1}{T} \tag{2.1}$$

Dimana f adalah frekuensi dan T adalah periode

b) Perpindahan getaran

Perpindahan getaran memiliki pengertian jarak yang ditempuh dari suatu puncak ke puncak yang lainnya atau biasa disebut *peak to peak displacement.*

c) Kecepatan getaran

Kecepatan getaran merupakan kecepatan suatu benda saat mengalami satu getaran.

d) Percepatan getaran

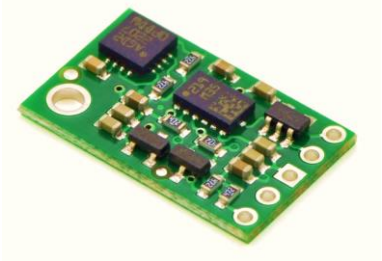
Secara umum percepatan merupakan perubahan dari kecepatan. Percepatan dinyatakan dalam satuan g, dimana g merupakan percepatan yang disebabkan oleh gravitasi bumi.

e) Fase getaran

Fase getaran akan memberikan informasi benda atau bagian yang bergetar relatif terhadap benda atau bagian lain yang bergetar dengan frekuensi yang sama dan salah satunya dijadikan sebagai referensi.

Untuk mengukur getaran diperlukan transduser getaran yang berfungsi untuk mengubah sinyal getaran menjadi sinyal listrik. Masing-masing amplitudo getaran yang dapat diukur menjadi parameter perpindahan, kecepatan dan percepatan mempunyai transduser tersendiri. Salah satu sensor yang digunakan dalam pengukuran getaran khususnya percepatan adalah sensor *accelerometer*.

Accelerometer adalah sebuah perangkat yang mampu mengukur sebuah kekuatan akselerasi. Kekuatan ini mungkin statis (diam) seperti halnya kekuatan konstan dari gravitasi bumi, atau juga bisa bersifat dinamis karena gerakan atau getaran dari sebuah akselerometer.



Gambar 2.2 Sensor *Accelerometer*

Accelerometer adalah sebuah transduser yang berfungsi untuk mengukur percepatan, mendeteksi dan mengukur getaran, ataupun untuk mengukur persepatan akibat gravitasi bumi. *Accelerometer* juga dapat digunakan untuk mengukur getaran yang terjadi pada kendaraan, bangunan, mesin, dan juga bisa digunakan, untuk mengukur getaran yang terjadi di dalam bumi, getaran mesin, jarak yang dinamis, dan kecepatan dengan ataupun tanpa pengaruh gravitasi bumi.

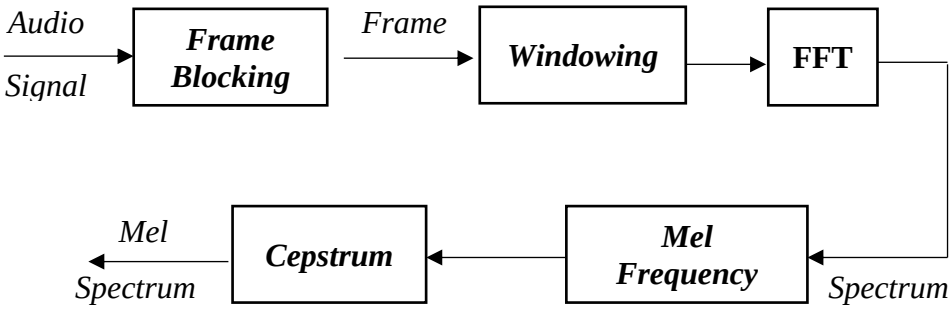
Prinsip kerja dari transduser ini berdasarkan hukum fisika bahwa apabila suatu konduktor digerakkan melalui suatu medan magnet, atau jika suatu medan magnet digerakkan melalui suatu konduktor, maka akan timbul suatu tegangan induksi pada konduktor tersebut.

B. *Mel-Frequency Cepstrum Coefficients* (MFCC)

MFCC didasarkan pada variasi yang telah diketahui dari jangkauan kritis telinga manusia terhadap frekuensi. Filter dipisahkan secara linear pada frekuensi rendah dan logaritmik pada frekuensi tinggi. Hal ini telah dilakukan untuk menangkap karaktersistik penting dari sinyal suara. Tujuan utama MFCC adalah untuk meniru perilaku telinga manusia. Selain itu MFCC telah terbukti bisa menyebutkan variasi dari gelombang suara itu sendiri. Beberapa keunggulan dari metode MFCC adalah (Manunggal, 2005):

1. Mampu menangkap karakteristik suara yang sangat penting bagi pengenalan suara, atau dengan kata lain dapat mengangkap informasi-informasi penting yang terkandung dalam *signal* suara.
2. Menghasilkan data seminimal mungkin, tanpa menghilangkan informasi-informasi penting yang terkandung didalamnya.
3. Mereplikasi organ pendengan manusia dalam melakukan persepsi terhadap *signal* suara.

Diagram alir dari proses MFCC dapat dilihat pada Gambar 2.3.



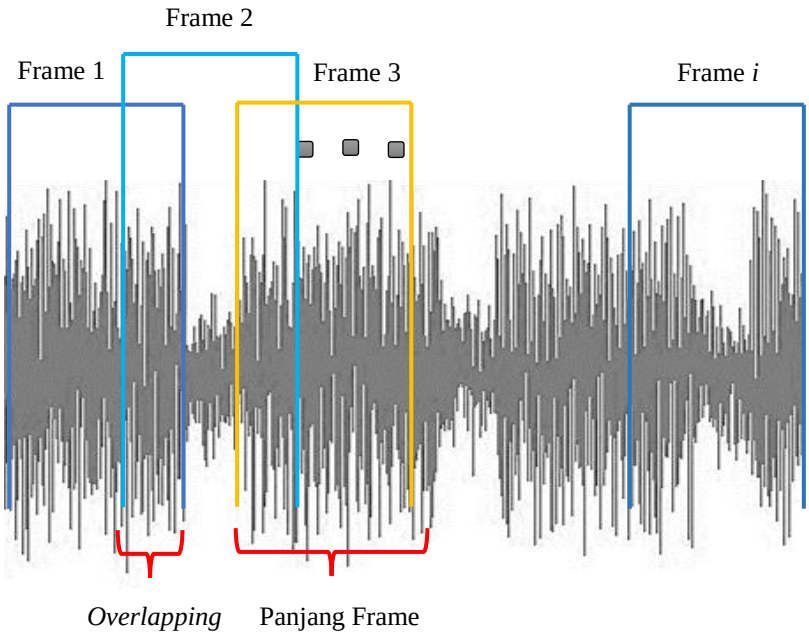
Gambar 2.3 Diagram blok proses MFCC

Penjelasan tiap tahapan pada proses MFCC adalah sebagai berikut:

1. *Frame Blocking*

Karena *signal* suara terus mengalami perubahan akibat adanya pergeseran artikulasi dari organ produksi vokal, *signal* harus diproses secara *short segments (short frame).* Panjang *frame* yang digunakan sangat mempengaruhi keberhasilan dalam analisa spectral. Di satu sisi, ukuran dari *frame* harus sepanjang mungkin untuk dapat menunjukkan resolusi frekuensi yang baik. Tetapi di lain sisi, ukuran *frame* juga harus cukup pendek untuk dapat menunjukkan resolusi waktu yang baik.

Proses *frame* ini dilakukan terus sampai seluruh *signal* dapat diproses. Selain itu, proses ini umumnya dilakukan secara *overlapping* untuk setiap *frame*-nya. Panjang daerah *overlap* yang umum digunakan adalah kurang lebih 30% sampai 50% dari panjang *frame. Overlapping* dilakukan untuk menghindari hilangnya ciri atau karakteristik suara pada perbatasan perpotongan setiap *frame.*



Gambar 2.4 Analisis *Frame Blocking*

2. *Windowing*

Proses selanjutnya adalah melakukan *windowing* pada tiap *frame* untuk meminimalkan diskontinuitas sinyal pada awal dan akhir tiap *frame.* Konsepnya adalah meminimalisasi distorsi spektral dengan menggunakan *window* untuk memperkecil sinyal hingga mendekati nol pada awal dan akhir setiap *frame.* Jika *window* didefenisikan sebagai *w(n), 0 ≤ n ≤ N-1*, dengan *N* adalah banyaknya sample tiap *frame* , maka hasil dari *windowing* adalah sinyal dengan persamaan:

$$Y_t(n) = x_t(n)w(n), 0 \leq n \leq N\text{-}1 \qquad (2.2)$$

Pada umumnya, *window* digunakan adalah *hamming window*, dengan persamaan :

$$w(n) = 0.54 - 0.46\, cos\left(\frac{2\pi n}{N-1}\right),\ 0 \leq n \leq\ N-1 \qquad (2.3)$$

3. *Fast Fourier Transform* (FFT)

Tahap ini mengkonversi tiap frame dengan *N* sampel dari *time domain* menjadi *frekuensi domain.* FFT adalah suatu algoritma untuk mengimplementasikan *Discrete Fourier Transform* (DFT) yang didefenisikan pada himpuna *N* sample $\{x_n\}$ sebagai berikut:

$$X_n = \sum_{k=0}^{N-1} x_k e^{-2\pi jkn/N}\ ,\ n= 0,1,2,\ldots,N\text{-}1 \qquad (2.4)$$

j digunakan untuk menotasikan unit imaginer, yaitu $j=\sqrt{-1}$. Secara umum X_n adalah bilangan kompleks. Barisan $\{x_n\}$ yang dihasilkan diartikan sebagai berikut:

frekuensi nol berkorespondensi dengan n=0, frekuensi positif $0 < f < F_s/2$ berkorespondensi dengan nilai $1 \leq n \leq N/2-1$, sedangkan frekuensi negetif $-F_s/2 < f < 0$ berkorespondensi dengan $N/2+1 < n < N-1$. Dalam hal ini *Fs* adalah *sampling frequency*. Hasil yang didapatkan dalam tahap ini bisa disebut dengan spektrum sinyal atau *periodogram.*

4. *Mel-frequency Wrapping*

Studi psikofisik menunjukkan bahwa persepsi manusia terhadap frekuensi sinyal suara tidak berupa skala linier. Oleh karena itu , untuk setiap nada dengan frekuensi actual *f* (dalam Hertz), tinggi subjektifnya diukur dengan skala 'mel'. Skala *mel-frequency* adalah selang frekuensi dibawah 1000 Hz dan selang logaritmik untuk frekuensi di atas 1000 Hz, sehingga pendekatan berikut dapat digunakan untuk menghitung *mel-frequency* untuk frekuensi *f* dalam Hz:

$$\text{Mel}(f)=2595*\log_{10}(1+f/700) \tag{2.5}$$

5. *Cepstrum*

Langkah terakhir, konversikan log *mel spectrum* ke domain waktu. Hasilnya disebut *mel frequency cepstrum coefficients.* Representasi *cepstral* spektrum suara merupakan representasi properti spectral local yang baik dari suatu sinyal untuk analisis *frame. Mel spectrum coefficients* (dan logaritmanya) berupa bilangan riil, sehingga dapat dikonversikan ke domain waktu dengan menggunakan *Descrete Cosine Transform* (DCT).

DCT merupakan langkah terakhir dari proses utama MFCC *feature extraction.* Konsep dasar dari DCT adalah mendekorelasikan *mel spectrum* sehingga menghasilkan representasi yang baik dari properti spektral lokal. Pada dasarnya konsep dari DCT sama dengan *inverse fourier transform.* Namun hasil dari DCT mendekati PCA (*principle component analysis*). PCA adalah metode statik klasik yang digunakan secara luas dalam analisa data dan kompresi. Hal inilah yang menyebabkan seringkali DCT menggantikan *inverse fourier transform* dalam proses ekstraksi ciri MFCC.

C. Pengenalan Pola

Pengenalan pola (*pattern recognition*) dapat diartikan sebagai proses klasifikasi dari objek atau pola menjadi beberapa kategori atau kelas.

Pola adalah bentuk atau model (atau, lebih abstrak, suatu set peraturan) yang dapat dipakai untuk membuat atau untuk menghasilkan suatu atau bagian daru sesuatu, khususnya jika sesuatu yang ditimbulkan mempunyai sejenis pola dasar yang dapat ditunjukkan atau terlihat, yang mana dapat dikatakan mempertunjukkan pola.

Metode klasifikasi yang digunakan pada system pengenalan pola memiliki dua jenis pendekatan. Pendekatan statistik dan pendekatan struktural (atau sintaktik). Pengenalan pola statistik didasarkan pada karakteristik statistikal dari pola-pola yang ada dengan asumsi bahwa pola-pola tersebut dihasilkan oleh sebuah sistem probabilistic. Pengenalan pola struktural berdasarkan pada hubungan struktural dari fitur setiap pola.

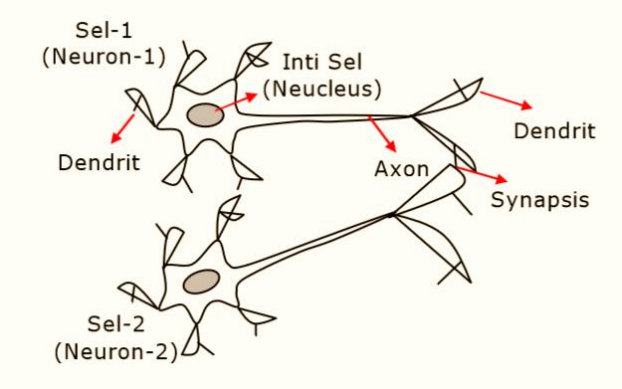
Sebuah sistem pengenalan pola terdiri dari sensor yang mengumpulkan pola yang akan diproses dan mengukur variabel dari setiap pola, *pre-processing* yang menghilangkan *noise* dalam data, mekanisme ekstraksi fitur untuk mendapatkan informasi numerik atau simbolik dari pola-pola tersebut, model pembelajaran yang mempelajari pemetaan antara fitur dan kelompok pola, metode klasifikasi yang memisah-misahkan pola-pola tersebut ke dalam kategori berdasarkan fitur dan model pembelajaran, dan *post-processing* yang mengevaluasi benar tidaknya hasil yang didapat.

Pengenalan pola merupakan bidang dalam pembelajaran mesin dan dapat diartikan sebagai tindakan mengambil data mentah dan bertindak berdasarkan klasifikasi data. Dengan demikian, hal tersebut merupakan himpunan kaidah bagi pembelajaran yang diawasi (*supervised learning).*

D. Jaringan Saraf Tiruan

Jaringan Saraf Tiruan (JST) merupakan implementasi dari teknologi *artificial intelligence.* Jaringan saraf tiruan adalah salah satu representasi buatan dari otak manusia yang selalu mencoba untuk mensimulasikan proses

pembelajaran pada otak manusia tersebut. Istilah buatan digunakan karena jaringan saraf ini diimplementasikan dengan menggunakan program komputer yang mampu menyelesaikan sejumlah proses perhitungan selama proses pembelajaran. Jaringan saraf tiruan merupakan sistem pemroses informasi yang memiliki karekteristik mirip dengan jaringan saraf biologi. Menurut subyanto, jaringan saraf tiruan adalah membuat model sistem komputasi yang dapat menirukan cara kerja jaringan saraf biologi. Secara umum Haykin mendefenisikan sebuah jaringan saraf tiruan adalah sebuah mesin yang dirancang untuk mempolakan cara bagaimana otak mengerjakan sebuah fungsi tertentu. Jaringan biasanya diimplementasikan dengan menggunakan komponen elektronika atau disimulasikan dalam sebuah perangkat lunak pada komputer digital. Untuk mencapai tampilan yang baik, jaringan saraf tiruan memakai interkoneksi yang sangat besar antara sel-sel komputasi yang disebut "neuron" atau "unit pemroses". Sebagai mesin yang adaptif, sebuah jaringan saraf tiruan adalah sebuah prosessor besar terdistribusi yang parallel yang tersusun dari unit pemroses sederhana yang mempunyai kecenderungan untuk menyimpan pengalaman dan pengetahuan dan membuatnya siap digunakan.



Gambar 2.5 Sel Saraf Manusia

Hal itu menyerupai otak dalam dua aspek:

- Pengetahuan dibutuhkan oleh jaringan dari lingkungan melalui proses pembelajaran
- Kekuatan koneksi interneuron, dikenal sebagai bobot sinapsis, digunakan untuk menyimpan pengetahuan yang dibutuhkan.

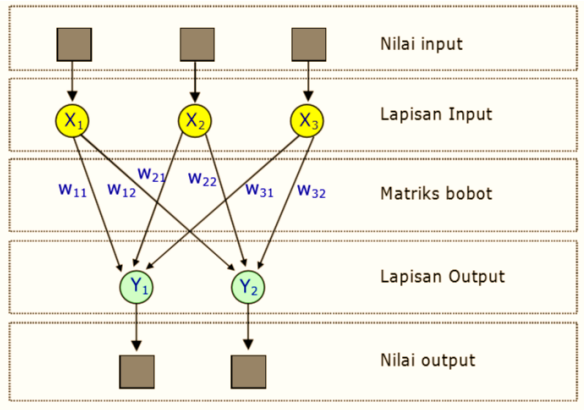
Menurut Fausset(1994), suatu JST dicirikan oleh tiga hal sebagai berikut:

1. Arsitektur jaringan saraf tiruan

Arsitektur jaringan adalah pengeturan *neuron* dalam suatu lapisan, pola hubungan dalam lapisan dan di antara lapisan. Dalam JST, *neuron-neuron* diatur dalam sebuah lapisan (*layer).* Ada 3 tipe lapisan yaitu lapisan input, lapisan tersembunyi (*hidden layer)* dan lapisan output. Jaringan *neuron* dikelompokkan sebagai lapis tunggal (*single layer*) yang terdiri atas lapisan input dan output, dan lapis banyak (*multiple layer*) yang terdiri atas lapisan input, lapisan tersembunyi, dan lapisan output.

a. Jaringan dengan lapis tunggal

Jaringan dengan lapisan tunggal hanya memiliki satu lapisan dengan bobot-bobot terhubung. Jaringan ini hanya menerima input kemudian secara langsung akan mengolahnya menjadi iutput tanpa harus melalui lapisan tersembunyi.

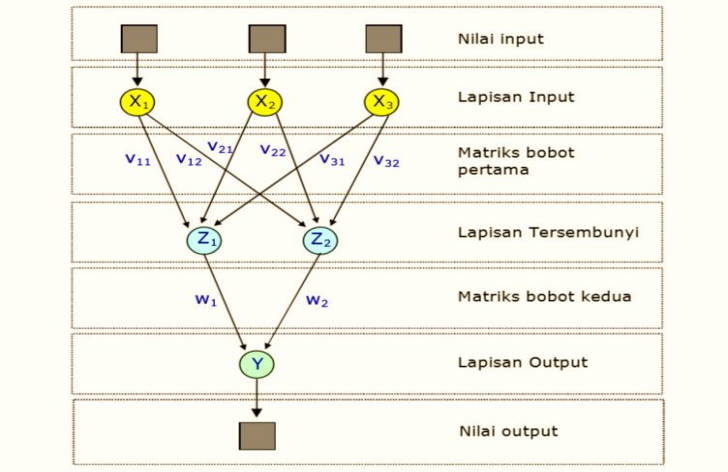


Gambar 2.6 JST lapis tunggal

b. Jaringan dengan banyak lapisan

Jaringan dengan banyak lapisan memiliki satu atau lebih lapisan yang terletak diantara lapisan input dan lapisan output (memiliki satu atau lebih lapisan tersembunyi). Umumnya, ada lapisan bobot-bobot yang terletak antara 2 lapisan yang bersebelahan. Jaringa dengan banyak lapisan ini dapat menyelesaikan permasalahan yang lebih sulit daripada lapisan dengan lapisan tuggal, tentu saja dengan pembelajaran yang lebih rumit.

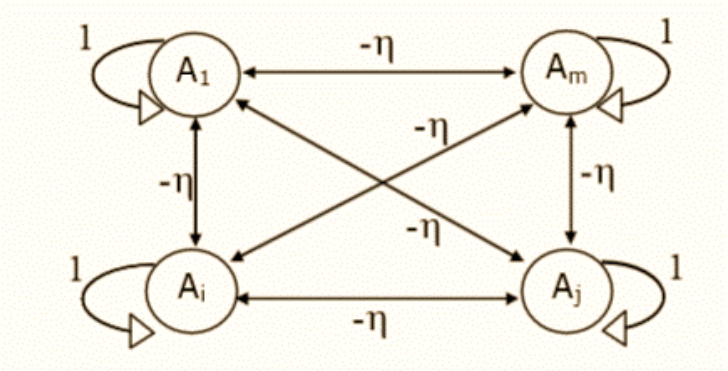
Namun demikian, pada banyak kasus, pembelajaran pada jaringan dengan banyak lapisan ini lebih sukses dalam menyelesaikan masalah.



Gambar 2.7 JST Lapis Banyak

c. Jaringan dengan lapisan kompetitif

Umumnya, hubungan antar neuron pada lapisan kompetitif ini tidak diperlihatkan pada diagram arsitektur. Gambar menunjukkan salah satu arsitektur jaringan dengan lapisan kompetitif yang memiliki bobot.



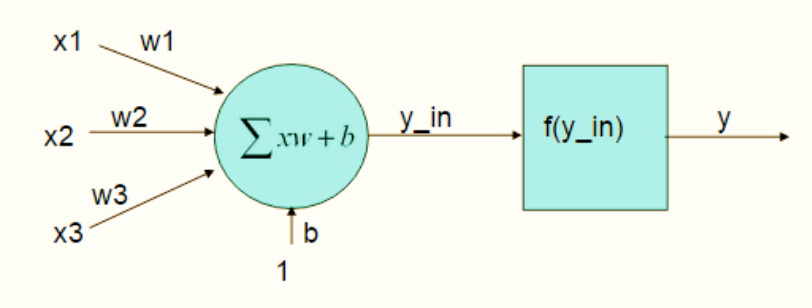
Gambar 2.8 JST lapisan kompetitif

2. Metode pembelajaran untuk penentuan pembobot koneksi.

Metode pembelajaran digunakan untuk menentukan nilai pembobot yang akan digunakan pada saat pengujian. Ada dua tipe pembelajaran, yaitu dengan pengarahan (*supervised learning*) dan tanpa pengarahan (*unsupervised*). Sedangkan metode pembelajaran JST, diantaranya : *perceptron*, Aturan Delta (Adaline/Madaline), Propagasi Balik, *Self Organizing Map* (SOM), dan *Learning Vector Quantization* (LVQ).

a. Perceptron

Biasanya digunakan untuk mengklasifikasikan suatu tipe pola tertentu yang sering dikenal dengan pemisahan secara linear. Algoritma yang digunakan akan mengatur parameter-parameter bebasnya melalui proses pembelajaran. Perceptron juga termasuk salah satu bentuk jaringan saraf yang sederhana. Pada dasarnya, perceptron pada jaringan saraf dengan satu lapisan memiliki bobot yang bisa diatur dan suatu nilai ambang (*threshold*). Nilai *threshold* pada fungsi aktivasi adalah non negatif. Fungsi aktivasi ini dibuat sedemikian rupa sehingga terjadi pembatasan antara daerah positif dan daerah negatif.



Gambar 2.9 *Perceptron*

b. Heb Rule

Heb Rule adalah metode pembelajaran yang paling sederhana. Pada metode ini pembelajaran dilakukan dengan cara memperbaiki nilai bobot sedemikian rupa sehingga jika ada neuron yang terhubung, dan keduanya pada kondisi hidup pada saat yang sama, maka bobot antara keduanya dinaikkan.

c. *Delta Rule*

Pada *Delta Rule* akan mengubah bobot yang menghubungkan antara jaringan input ke unit *output* dengan nilai target. Hal ini dilakukan untuk meminimalkan error selama pelatihan pola.

d. *Back Propagation*

Merupakan algoritma pembelajaran yang terawasi dan biasanya digunakan oleh perceptron dengan banyak lapisan untuk mengubah bobot-bobot yang terhubung dengan neuron-neuron yang ada pada lapisan tersembunyinya. Algoritma *backpropagation* menggunakan error output untuk mengubah nilai bobot-bobotnya dalam arah mundur (*backward*). Untuk mendapatkan error ini, tahap perambatan maju (*forward propagation*) harus dikerjakan terlebih dahulu.

e. *Hetroassociative Memory*

Merupakan jaringan yang bobot-bobotnya ditentukan sedemikian rupa sehingga jaringan tersebut dapat menyimpan kumpulan pengelompokan pola. Algoritma pembelajaran yang biasa digunakan oleh jaringan ini adalah *Hebb Rule* dan *Delta Rule.*

f. *Biodirectional Associative Memory*

Biodirectional Associative Memory (BAM) adalah model jaringan saraf yang memiliki 2 lapisan dan terhubung penuh dari suatu lapisan ke lapisan yang lainnya. Pada jaringan ini dimungkinkan adanya hubungan timbal balik antara lapisan input dan lapisan output. Namu demikian, bobot yang menghubungkan antara satu neuron (A) di satu lapisan dengan neuron (B) di lapisan lainnya akan sama dengan bobot yang menghubungkan neuron (B) ke neuron (A). Bisa juga dikatakan bahwa, matriks bobot yang menghubungkan neuron-neuron pada lapisan output kelapisan input sama dengan transpose matriks bobot neuron-neuron yang menghubungkan lapisan input ke lapisan output.

g. *Learning Vector Quantitation*

Learning Vector Quantitation (LVQ) adalah suatu metode untuk melakukan pembelajaran pada lapisan kompetitif yang terawasi. Suatu lapisan kompetitif akan secara otomatis belajar untuk mengklasifikasikan vektor-vektor input. Kelas kelas yang didapatkan sebagai hasil dari lapisan kompetitif ini hanya tergantung pada jarak antara vektor-vektor input. Jika

2 vektor input mendekati sama, maka lapisan kompetitif akan meletakkan kedua vector input ke dalam kelas yang sama.

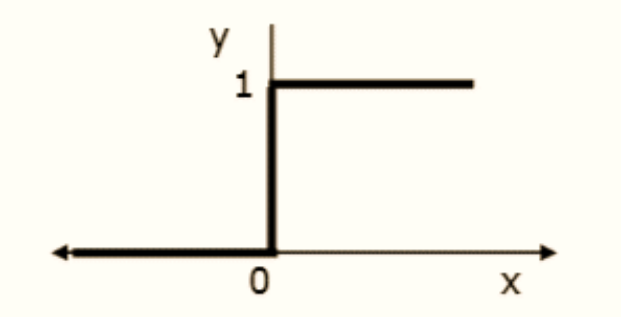
3. Fungsi aktivasi yang digunakan

Fungsi aktivasi merupakan fungsi yang menentukan level aktivasi, keadaan internal sebuah *neuron* dalam JST. Keluaran aktivasi ini biasanya dikirim sebagai sinyal ke *neuron* lainnya. Contoh fungsi aktivasi ialah fungsi identitas fungsi tangga biner (*binary step function*), fungsi tangga bipolar, fungsi sigmoid biner dan fungsi sigmoid bipolar.

a. Fungsi *Hard Limit* (Undak Biner)

Jaringan dengan lapisan tunggal sering menggunakan fungsi undak (*step function*) untuk mengkonversikan input dari suatu variable yang bernilai kontinu ke suatu output biner (0 atau 1). Fungsi undak biner dirumuskan sebagai:

$$y = \begin{cases} 0, & \text{jika } x \leq 0 \\ 1, & \text{jika } x > 0 \end{cases} \tag{2.6}$$

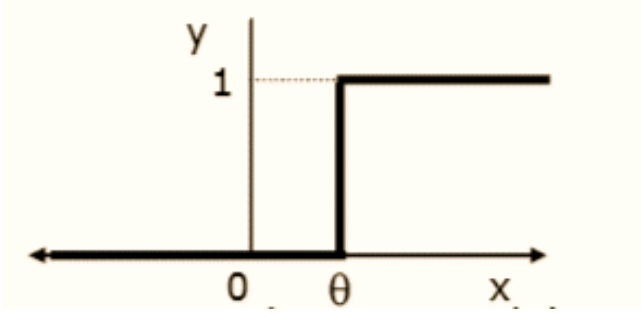


Gambar 2.10 Fungsi *Hard Limit*

b. Fungsi Nilai Ambang (*threshold)*

Fungsi undak biner dengan menggunakan nilai ambang sering juha disebut dengan nama fungsi nilai ambang (*threshold*) atau fungsi *Heaviside.* Fungsi undak biner (dengan nilai ambang) dirumuskan sebagai:

$$y = \begin{cases} 0, & \text{jika } x < \theta \\ 1, & \text{jika } x \geq \theta \end{cases} \tag{2.7}$$

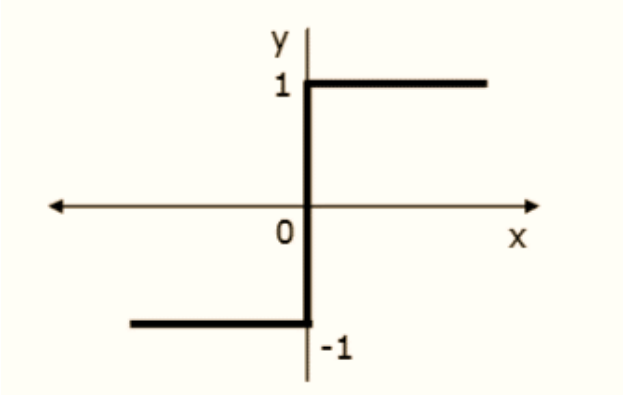


Gambar 2.11 Fungsi Nilai Ambang

c. Fungsi Bipolar (*Symetric Hard Limit*)

Fungsi bipolar sebenarnya hampir sama dengan fungsi undak biner, hanya saja output yang dihasilkan berupa 1, 0, -1. Fungsi bipolar dirumuskan sebagai:

$$y = \begin{cases} 1, & \text{jika } x > 0 \\ 0, & \text{jika } x = 0 \\ -1, & \text{jika } x < 0 \end{cases} \tag{2.8}$$

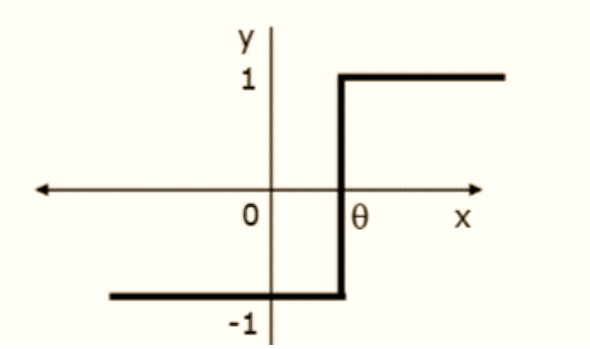


Gambar 2.12 Fungsi Bipolar

d. Fungsi Bipolar (dengan *threshold*)

Fungsi Bipolar hampir sama dengan fungsi undak biner dengan *threshold*, hanya saja output yang dihasilkan berupa 1, 0, -1. Fungsi bipolar (dengan nilai ambang) dirumuskan sebagai:

$$y = \begin{cases} 1, & \text{jika } x \geq \theta \\ 1, & \text{jika } x < \theta \end{cases} \tag{2.9}$$

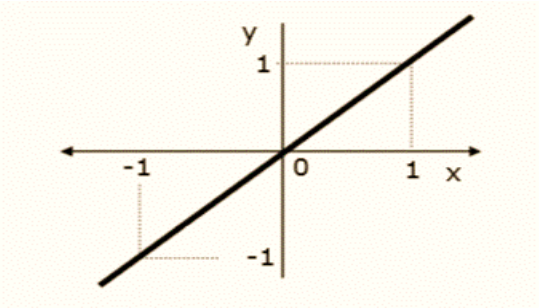


Gambar 2.13 Fungsi Bipolar (dengan *threshold*)

e. Fungsi Linerar (identitas)

Fungsi liner memiliki nilai output yang sama dengan nilai inputnya. Dirumuskan sebagai:

$$y = x$$

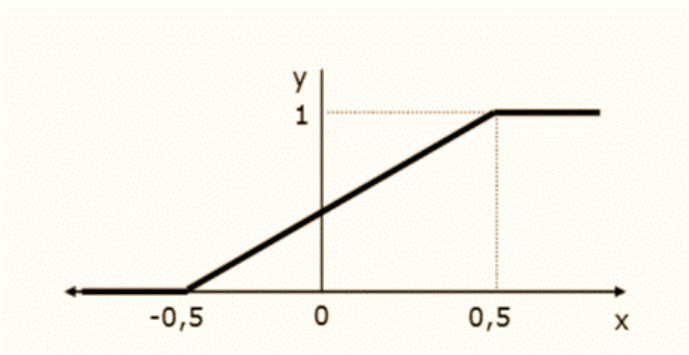


Gambar 2.14 Fungsi Liner (identitas)

f. Fungsi *Saturating Linear*

Fungsi ini akan bernilai 0 jika inputnya kurang dari -1/2 dan akan bernilai 1 jika inputnya lebih dari ½. Sedangkan jika nilai input terletak antara -1/2 dan 1//2, maka outputnya akan bernilai sama dengan nilai input ditambah ½. Dirumuskan sebagai :

$$y = \begin{cases} 1, & \text{jika } x \geq 0.5 \\ x + 0.5, & \text{jika } -0.5 \leq x \leq 0.5 \\ 0, & \text{jika } x \leq 0.5 \end{cases} \qquad (2.10)$$

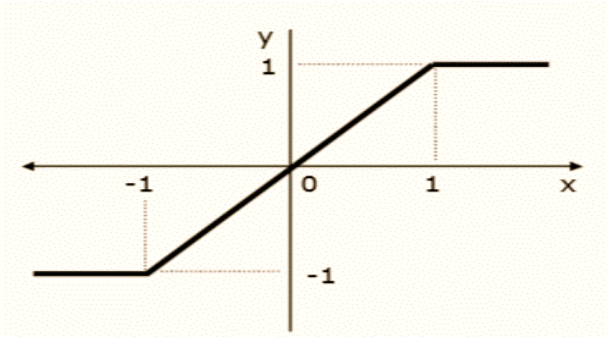


Gambar 2.15 Fungsi *Saturating Linerar*

g. Fungsi *Symetric Saturating Lineear*

Fungsi ini akan bernilai -1 jika inputnya kurang dari -1, dan akan bernilai 1 jika inputnya lebih dari 1. Sedangkan jika inputnya terletak antara -2 dan 1, maka outputnya akan bernilai sama dengan nilai inputnya. Dirumuskan sebagai:

$$y = \begin{cases} 1, & \text{jika } x \geq 1 \\ x, & \text{jika } -1 \leq x \leq 1 \\ -1, & \text{jika } x \leq -1 \end{cases} \quad (2.11)$$



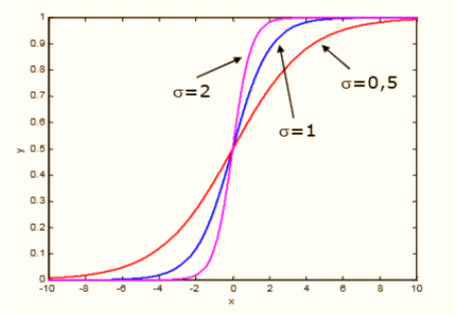
Gambar 2.16 Fungsi *Symetric Saturating Liner*

h. Fungsi *Sigmod Biner*

Fungsi ini digunakan untuk jaringan saraf yang dilatih dengan menggunakan metode *backpropagation.* Fungsi *Sigmod Biner* memiliki nilai pada rentang 0 sampai 1. Oleh karena itu, fungsi ini sering digunakan untuk jaringan saraf yang membutuhkan nilai output yang terletak pada interval 0 sampai 1. Namun fungsi ini bisa juga digunakan oleh jaringan saraf yang nilai outputnya 0 atau 1. Dirumuskan sebagai:

$$y = f'(x) = \frac{1}{1+e^{-\sigma x}} \tag{2.12}$$

dengan $f'(x) = \sigma f(x)[1 - f(x)]$



Gambar 2.17 Fungsi *Sigmod Biner*

i. Fungsi *Sigmod Bipolar*

Fungsi *sigmod bipolar* hampir sama dengan fungsi sigmod biner, hanya saja output dari fungsi ini memiliki rentang antara 1 sampai -1. Dirumuskan sebagai:

$$y = f(x) = \frac{1-e^{-x}}{1+e^{-x}} \tag{2.13}$$

dengan $f'(x) = \frac{\sigma}{2}[1 + f(x)][1 - f(x)]$

j. Fungsi *Hyperbolic Tangent*

Fungsi *Hyperbolic Tangent* hampir sama dengan fungsi sigmod bipolar, hanya saja output dari fungsi ini memiliki rentang antara -1 sampai 1. Dirumuskan sebagai:

$$y = f(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \tag{2.14}$$

atau

$$y = f(x) = \frac{1-e^{-2x}}{1+e^{-2x}} \tag{2.15}$$

dengan $f'(x) = [1 + f(x)][1 - f(x)]$

4. Proses Pembelajaran

Pada otak manusia, informasi yang dilewatkan dari satu neuron ke neuron yang lainnya berbentuk rangsangan listrik melalui dendrit. Jika rangsangan tersebut diterima oleh suatu neuron, maka neuron tersebut akan membangkitkan

output ke semua neuron yang berhubungan dengannya sampai informasi tersebut sampai ke tujuannya yaitu terjadinya suatu reaksi. Jika rangsangan yang diterima terlalu halus, maka output yang dibangkitkan oleh neuron tersebut tidak akan direspon. Tentu saja sangatlah sulit untuk memahami bagaimana otak manusia bisa belajar. Selama proses pembelajaran, terjadi perubahan yang cukup berarti pada bobot-bobot yang menghubungkan antar neuron. Apabila ada rangsangan yang sama dengan rangsangan yang telah diterima oleh neuron, maka neuron akan memberikan reaksi dengan cepat. Namun apabila kelak ada rangsangan yang berbeda dengan apa yang telah diterima oleh neuron, maka neuron akan segera beradaptasi untuk memberikan reaksi yang sesuai. Jaringan saraf akan mencoba untuk mensimulasikan kemampuan otak manusia untuk belajar. Jaringan saraf tiruan juga tersusun atas neuron-neuron dan dendrit. Tidak seperti model biologis, jaringan saraf memiliki struktur yang tidak dapat diubah, dibangun oleh sejumlah neuron, dan memiliki nilai tertentu yang menunjukkan seberapa besar koneksi antara neuron (yang dikenal dengan nama bobot). Perubahan yang terjadi selama proses pembelajaran adalah perubahan nilai bobot. Nilai bobot akan bertambah, jika informasi yang diberikan oleh neuron yang bersangkutan tersampaikan, sebaliknya jika informasi tidak disampaikan oleh suatu neuron ke neuron yang lainnya, maka nilai bobot yang menghubungkan keduanya akan dikurangi. Pada saat pembelajaran dilakukan pada input yang berbeda, maka nilai bobot akan diubah secara dinamis hingga mencapai suatu nilai yang cukup seimbang. Apabila nilai ini telah tercapai mengindikasikan bahwa tiap-tiap input telah berhubungan dengan output yang diharapkan.

1. Pembelajaran terawasi

   Metode pembelajaran pada jaringan saraf disebut terawasi jika output yang diharapkan telah diketahui sebelumnya.

   Pada proses pembelajaran, satu pola input akan diberikan ke satu neuron pada lapisan input. Pola ini akan dirambatkan di sepanjang jaringan saraf hingga sampai ke neuron pada lapisan output. Lapisan output ini akan membangkitkan pola output yang nantinya akan dicocokkan dengan pola output targetnya. Apabila terjadi perbedaan antara

pola output hasil pembelajaran dengan pola target, maka disini akan muncul error. Apabila nolai error ini masih cukup besar, mengindikasikan bahwa masih perlu dilakukan lebih banyak pembelajaran lagi.

2. Pembelajaran Tak Terawasi

Pada metode pembelajaran yang tak terawasi ini tidak memerlukan target output. Pada metode ini, tidak dapat ditentukan hasil yang seperti apakah yang diharapkan selama proses pembelajaran. Selama proses pembelajarn, nilai bobot disusun dalam suatu range tertentu tergantung pada nilai input yang diberikan. Tujuan pembelajaran ini adalah mengelompokkan unit-unit yang hampir sama dalam suatu area tertentu. Pembelajaran ini biasanya sangat cocok untuk pengelompokan (klasifikasi) pola.

Jaringan saraf tiruan dikarakteristikkan dengan pola koneksi antarneuron yang disebut arsitektur, metode penentuan bobot pada setiap koneksinya (yang disebut *training* atau *learning* algoritma) dan fungsi aktivasinya. Jaringan saraf terdiri dari elemen pemroses sederhana yang dinamakan neuron, unit, sel atau node. Setiap neuron terkoneksi dengan neuron yang lain masing-masing dengan bobot terhubung. Bobot merepresentasikan informasi yang digunakan oleh jaringan untuk menyelesaikan masalah. Jaringan saraf dapat diaplikasikan untuk masalah yang sagat luas, seperti penyimpanan dan pemanggilan kembali data atau pola, mengklasifikasikan pola, menampilkan pemetaan secara umum dari pola input menjadi pola output, mengelompokkan pola yang sama, atau menemukan solusi untuk mengoptimalkan masalah. Setiap neuron mempunyai fungsi aktivasi atau level aktifitas, yang merupakan fungsi dari input yang telah diterima. Neuron mengirimkan aktivasinya sebagai sebuah sinyal ke beberapa neuron yang lain. Sebuah neuron hanya dapat mengirimkan sebuah sinyal dalam satu waktu, walaupun sinyalnya disebarkan pada beberapa neuron yang lain. Ciri utama yang dimiliki oleh jaringan saraf tiruan adalah kemampuannya untuk belajar. Belajar (*learning)* pada jaringan saraf tiruan dapat diartikan sebagai proses

penyesuaian parameter pembobot karena keluaran yang diinginkan tergantung pada harga pembobot interkoneksi yang dimiliki oleh sel. Proses belajar akan dihentikan jika nilai kesalahan atau error sudah dianggap cukup kecil untuk semua pasangan data latihan. Jaringan yang sedang melakukan proses belajar disebut berada dalam tahap latihan (*training*). Pada tahap awal pelatihan ini perlu dilakukan terlebih dahulu sebelum melakukan pengujian suatu objek.

Berdasarkan tingkat kemampuannya, jaringan saraf tiruan dapat diterapkan pada beberapa aplikasi yang cocok bila diterapkan pada klasifikasi pola, yakni memilih suatu input data ke dalam suatu kategori tertentu yang diterapkan. Di saamping itu jaringan saraf tiruan dapat diterapkan pada prediksi dan *self organizing* , yakni menggambarkan suatu objek secara keseluruhan hanya dengan mengetahui bagian dari objek lain dan memiliki kemampuan untuk mengolah data-data tanpa harus memilih data sebagai target. Selanjutnya jaringan saraf tiruan juga mampu diterapkan pada masalah optimasi, yakni mencari jawaban atau solusi terbaik dari suatu masalah.

Prosedur yang digunakan untuk menampilkan proses pembelajaran disebut algoritma pembelajaran yang fungsinnya memodifikasi bobot sinapsis pada jaringan dalam sebuah cara yang teratur untuk mencapai rancangan objek yang diinginkan. Modifikasi bobot sinapsis menyediakan sebuah metode untuk merancang jaringan saraf tiruan. Seperti sebuah pendekatan yang mendekati teori linear adaptif, yang telah didirikan dan sukses diaplikasikan pada bermacam-macam bidang. Bagaimanapun, sangat mungkin bagi jaringan saraf tiruan untuk memodifikasi topologinya sendiri yang dimotivasi dengan fakta bahwa neuron pada otak manusia dapat mati dan koneksi sinapsis baru dapat bertumbuh.

Kemampuan JST untuk belajar dan memperbaiki dirinya telah menghasilkan banyak algoritma atau aturan belajar alternatif yang dapat digunakan, dari sekian banyak aturan yang ada, yang paling sering digunakan adalah aturan belajar *backpropagation* yang termasuk kategori

supervised learning yang dapat digunakan memperbaiki kinerja jaringan saraf tiruan. Namun *Learning Vector Quantitation* merupakan metode pembelajaran pada lapisan kompetitif yang akan secara otomatis belajar untuk mengklasifikasi vector-vektor masukan (Kusmadewi,2004).

E. Penelitian Terkait

Ada beberapa penelitian yang terkait atau relevan dengan penelitian ini baik yang terkait dengan Ekstraksi Ciri MFCC maupun Jaringan Saraf Tiruan yang digunakan.

Penelitian pertama adalah penelitian dilakukan oleh Angga Setiawan pada tahun 2011 dengan judul penelitian "Aplikasi Pengenalan Ucapan dengan Ekstraksi *Mel-frequency Cesptrum Coeffisients* (MFCC) Melalui Jarigan Saraf Tiruan (JST) *Learning Vector Quantization*(LVQ) untuk Mengoperasikan Kursor Komputer". Peneliti menggunakan Ekstraksi ciri MFFC sebagai ekstraksi ciri pada sinyal suara yang telah direkam dan menggunakan jaringan saraf tiruan model *Learning Vector Quantization*(LVQ) sebagai identifikasi pola. Akurasi jaringan saraf tiruan yang dihasilkan adalah 83,99%.

Penelitian selanjutnya yang terkait dengan penelitian ini adalah penelitian yang dilakukan oleh Mada Sanjaya dan Zabidin Salleh pada tahun 2014 dengan judul penelitian " Implementasi Pengenalan Pola Suara *Menggunakan Mel-Frequency Cepstrum Coeffisients* (MFCC) dan *Adaptive Neuro-Fuzzy Inferense system* (ANFIS) Sebagai Kontrol Lampu Otomatis". Peneliti menggunakan MFCC sebagai ekstraksi ciri sinyal suara dan ANFIS sebagai metode pembelajaran untuk pengenalan pola suara. Hasil pengujian menunjukkan akurasi sebesar 95,90%

Selanjutnya adalah penelitian yang dilakukan oleh Dahriani Hakim Tanjung pada tahun 2015 dengan judul penelitian " Jaringan Saraf Tiruan dengan Backpropagation untuk Memprediksi Asma". Penelitian ini menggunakan data masukan yang mengacu pada riwayat penyakit asma seseorang dan menggunakan

jaringan saraf tiruan dengan metode *backpropagation* untuk memprediksi penyakit asma.

Selanjutnya adalah penelitian yang dilakukan oleh Tomi Budi Waluyo, Andi Setiono, dan Dwi Hanto pada tahun 2015 dengan judul penelitian "Penggunaan Akselerometer Pada Ponsel Android untuk Merekam Getaran Kabin Kereta Api". Penelitian digunakan untuk menganalisis kondisi rel yang dilalui, sebagai pembanding ataupun komplemen dari hasil pengukuran yang dilakukan oleh kereta ukur.

Selanjutnya adalah penelitian yang dilakukan oleh Zakaria Ramadhan pada tahun 2016 dengan judul penelitian "Perintah Suara Berbahasa Indonesia untuk Membuka dan Menutup Aplikasi dalam Sistem Operasi *Windows* Menggunakan Metode *Mel-Frequency Cepstrum Coefficients* dan Metode *Backpropagation*". Peneliti menggunakan MFCC sebagai ekstraksi ciri sinyal audio dan jaringan saraf tiruan *backpropagation* sebagai metode pengenalan pola. Akurasi jaringan saraf tiruan yang dibangun adalah 50 %.

Selanjutnya adalah penelitian yang dilakukan oleh Darma Putra pada tahun 2011 dengan judul penelitian " Verifikasi Biometrika Suara Menggunakan Metode MFCC dan DTW". Peneliti menggunakan MFCC sebagai ekstraksi ciri sinyal suara dan menggunakan metode DTW (*Dynamic Time Wrapping*) untuk proses pencocokan. Peneliti memperoleh akurasi tertinggi sebesar 93,254%.

BAB III
METODE PENELITIAN

A. Tujuan Penelitian

Tujuan dari penelitian ini adalah mengidentifikasi kereta api berdasarkan getaran dengan menggunakan metode *Mel-Frequency Cepstrum Coefficients* (MFCC) dan Jaringan Saraf Tiruan.

B. Tempat dan Waktu Penelitian

Penelitian ini dilakukan di Laboratorium Instrumentasi dan Komputasi Jurusan Fisika FMIPA Universitas Negeri Jakarta. Waktu Pelaksanaan Penelitian adalah Maret 2017-Juli 2017.

C. Metodologi Penelitian

Metode yang digunakan pada penelitian ini adalah Metode Penelitian dan Pengembangan (*Research and Development*). Tahapan penelitian secara umum meliputi studi literatur, pengembangan dan penyiapan instrumen, pengambilan data, pengolahan data dan analisa hasil.

Penelitian ini fokus pada ekstraksi ciri getaran dengan menggunakan MFCC dan identifikasi getaran dengan jaringan saraf tiruan.

D. Alat dan Bahan

Alat yang digunakan dalam penelitian ini, yaitu :

1. Seperangkat Komputer dengan sistem operasi Windows 10 Pro dengan spesifikasi :
   Processor : Intel(R) Core(TM) i3 CPU M 380 @ 2.53GHz
   RAM : 2.00 GB
   System Type : 32-bit *Operating System*
2. Seperangkat telepon pintar yang didalamnya terdapat sensor *Accelerometer*
3. Software MATLAB
4. Software *Accelerometer Analyzer*

Bahan yang digunakan dalam penelitian ini yaitu, rekaman getaran (dalam bentuk numerik atau angka) yang dihasilkan oleh sensor *accelerometer*. Rekaman getaran ini kemudian diubah menjadi sinyal audio menggunakan MATLAB dan disimpan dalam format *.wav.

Kode program Ekstraksi Ciri diambil dari :
http://www.edaboard.com/thread275114.html
Kode program jaringan saraf tiruan diambil dari :
https://pemrogramanmatlab.com/2016/10/24/jaringan-syaraf-tiruan-untuk-klasifikasi-citra-daun/

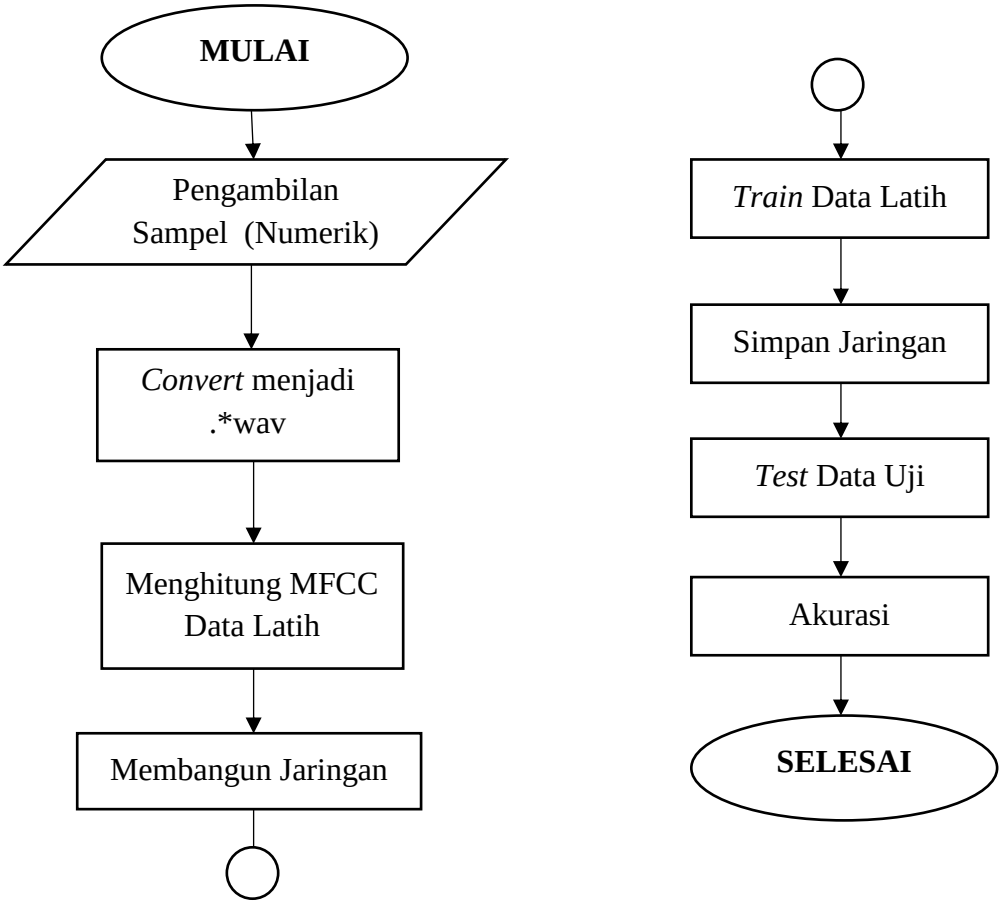
E. Prosedur Penelitian

Secara garis besar prosedur penelitian yang dilakukan adalah :

1. Persiapan Alat
2. Menjalankan aplikasi *Accelerometer Analyzer* pada telepon pintar
3. Melakukan perekaman getaran rel kereta api di salah satu stasiun kereta yang ada di provinsi DKI Jakarta (Stasiun Duren Kalibata) pada saat kereta api melintasi rel kereta
4. Perekaman getaran rel kereta dilakukan pada setiap kereta yang melintas baik kereta barang maupun kereta penumpang
5. Data rekaman getaran rel kereta api kemudian dikonversi menjadi sinyal audio menggunakan MATLAB
6. Menghilangkan noise pada sinyal (data rekaman) untuk menghasilkan sampel data yang berkualitas baik
7. Data getaran kemudian diekstrak dengan menggunakan ekstraksi ciri MFCC (*Mel Frequency Cepstral Coefficient)*
8. Sampel data sinyal audio dibagi menjadi dua kelompok yaitu data latih dan data uji dengan perbandingan 60 : 40
9. Membangun jaringan saraf tiruan menggunakan MATLAB
10. Mengatur parameter jaringan

11. Melatih sampel Data Latih pada jaringan
12. Menyimpan jaringan yang selanjutnya akan digunakan untuk menguji sampel Data Uji
13. Menguji sampel Data Uji pada jaringan untuk mengetahui akurasi jaringan saraf tiruan yang telah dibangun
14. Menghitung akurasi Jaringan
15. Menganalisa hasil jaringan saraf tiruan

F. Diagram Alir Penelitian



Gambar 3.1 Diagram Alir Penelitian

BAB IV
ANALISA DAN PEMBAHASAN

A. Pengambilan dan Pengolahan Data

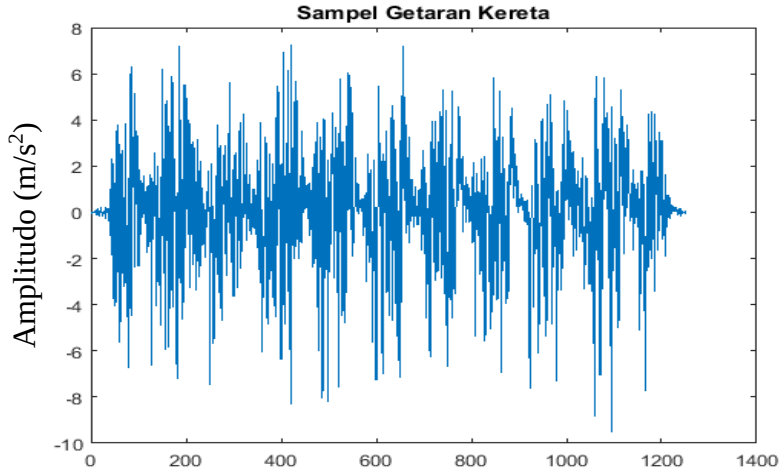
Proses pengambilan data dilakukan dengan menggunakan telepon pintar Android Samsung Galaxy Ace3 yang sudah terpasang aplikasi *Accelerometer Analyzer*. Aplikasi ini berfungsi untuk membaca nilai sensor *Accelerometer* yang terpasang pada telepon pintar. Sensor *Accelerometer* adalah salah satu sensor yang digunakan untuk mengukur getaran. Berikut adalah tampilan dasar aplikasi *Accelerometer Analyzer*.

Gambar 4.1 Tampilan *Accelerometer Analyzer*

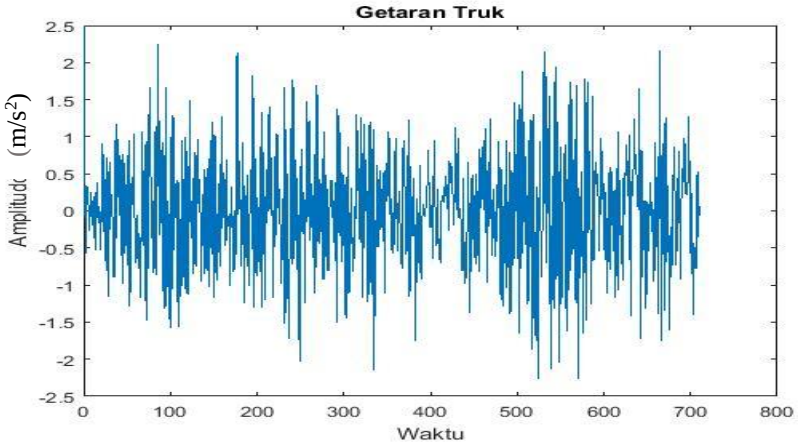
Perekaman data dilakukan pada setiap kereta yang melintas pada rel dan beberapa data dilakukan perekaman pada truk, metromini, motor dan pejalan kaki yang sebagai data bukan kereta. Data hasil rekaman getaran bukan kereta digunakan sebagai sampel pembeda dengan getaran kereta yang akan digunakan saat membangun sistem jaringan.

Data yang direkam dengan telepon pintar masih berupa file *.txt sehingga perlu diubah menjadi file *.wav agar dapat diekstrak dengan ekstraksi ciri *Mel*

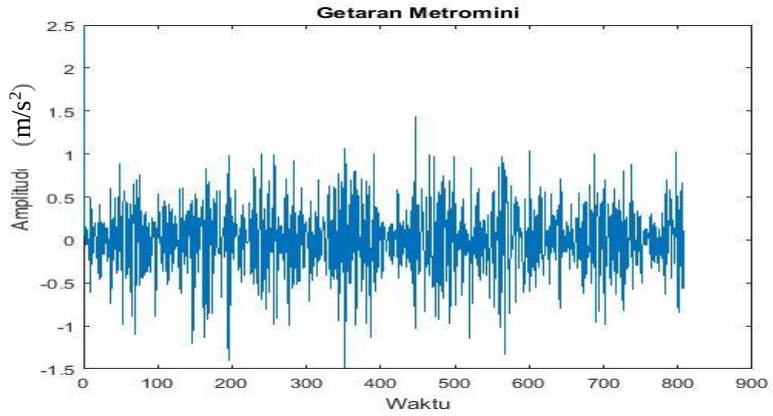
Frequency Cepstrum Coefficients (MFCC). Proses mengubah file dari *.txt menjadi file *.wav dilakukan dengan menggunakan MATLAB. Berikut adalah Grafik sampel getaran kereta (Gambar 4.2) dan getaran bukan kereta (Gambar 4.3). Dalam satu detik, sensor *Accelerometer* mampu merekam data 100 kali.



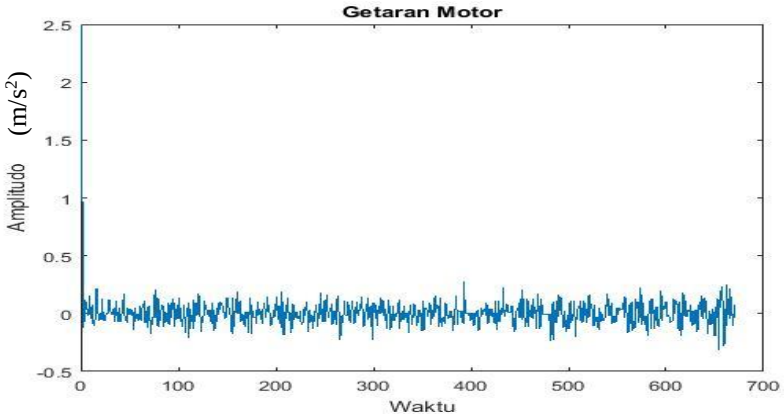
Gambar 4.2 Sampel Getaran Kereta



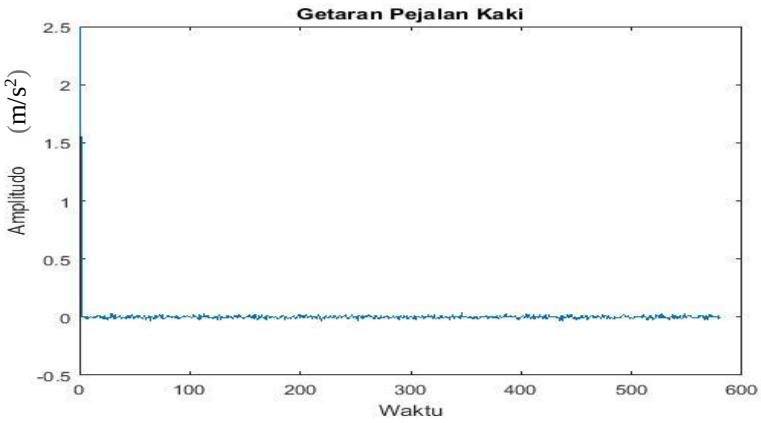
(a)



(b)



(c)



(d)

Gambar 4.3 Sampel Getaran bukan kereta

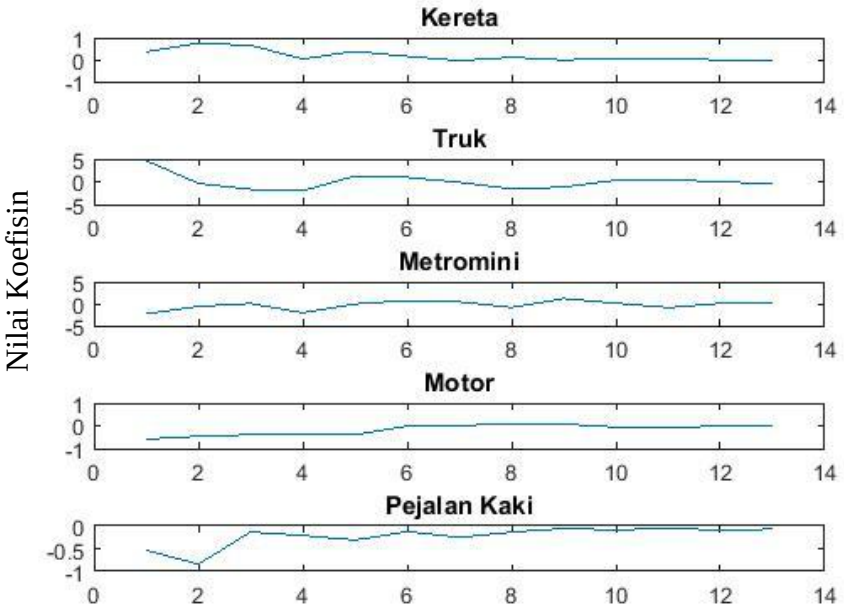
(a). Getaran Truk; (b).Getaran Metromini;

(c). Getaran Motor; (d). Getaran Pejalan Kaki

Data yang digunakan dalam penelitian ini sebanyak 320 data dengan pembagian 160 kereta yang terdiri dari beberapa jenis kereta api dan 160 bukan kereta yang terdiri dari dari Truk, Metromini, Motor dan Pejalan kaki. Data ini kemudian dibagi lagi menjadi dua kelompok data yaitu Data Latih sebanyak 200 data dan Data uji sebanyak 120 data.

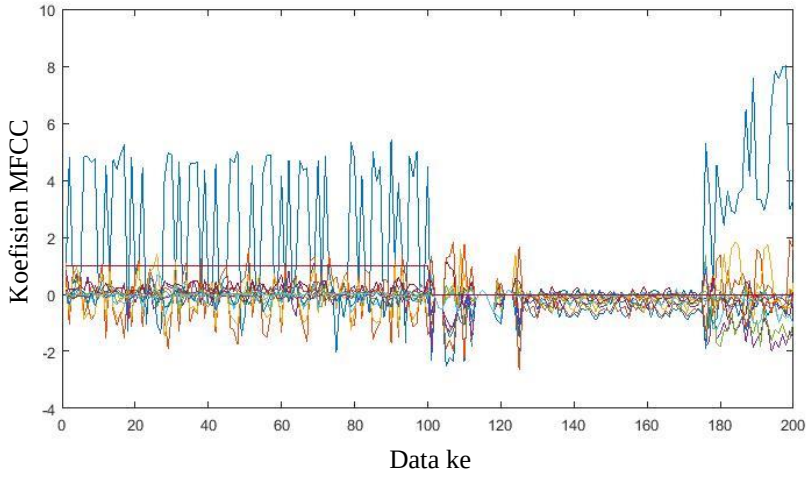
B. Ekstraksi Ciri MFCC

Setelah dilakukan pengelompokan data menjadi Data Latih dan Data Uji, Data Latih kemudian diekstrak dengan menggunakan ekstraksi ciri MFCC *(Mel Frequency Cepstral Coefficients).* Tujuan dari proses ini adalah untuk mendapatkan ciri khusus atau pola yang membedakan suatu sinyal dengan sinyal lain. Dengan ekstraksi ciri menggunakan metode MFFC ini, sinyal akan disederhanakan tanpa menghilangkan informasi penting yang terkadung dalam suatu sinyal. Pada penelitian ini setiap sinyal akan diekstrak menjadi 13 koefisien MFCC. Artinya setiap getaran kereta maupun bukan kereta akan menghasilkan 13 vektor ciri yang sekaligus menjadi pembeda dengan getaran lain.



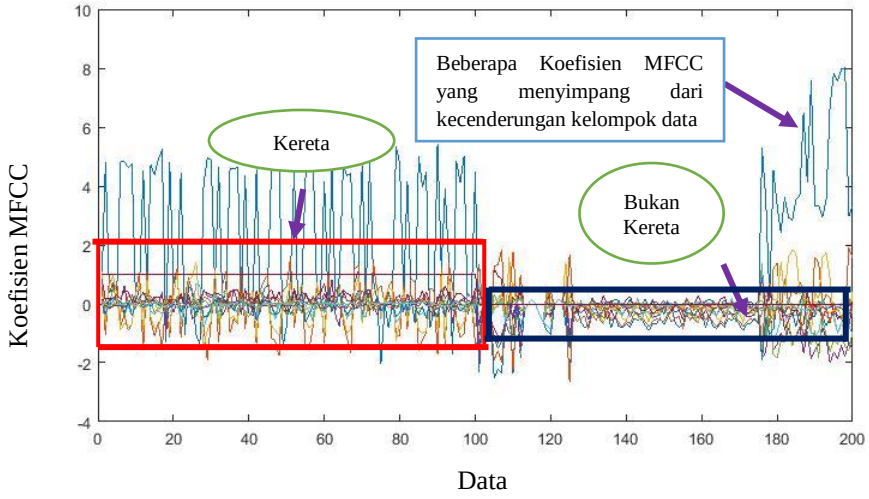
Gambar 4. 4 Koefisien MFCC Sampel Getaran

Gambar 4.4 menunjukkan contoh koefisien dari sampel data bukan kereta. Dari gambar dapat dilihat bahwa nilai koefisien MFCC truk lebih besar dari metromini, motor, dan pejalan kaki.



Gambar 4.5 Ekstraksi ciri MFCC sampel data latih

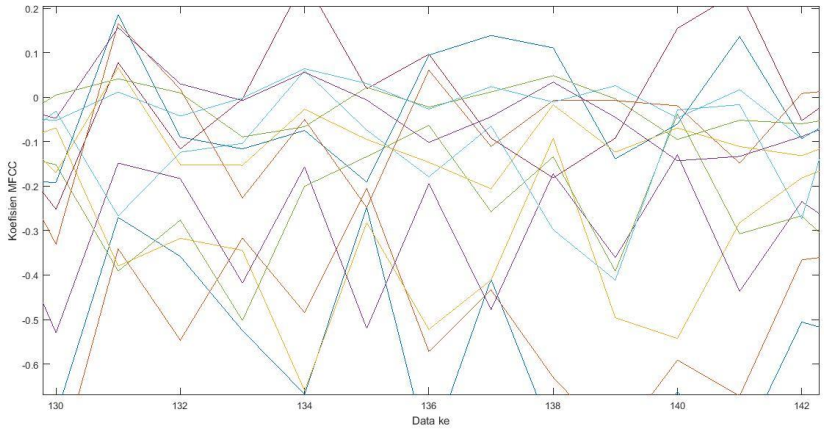
Gambar 4.5 menunjukkan Koefisien MFCC (K13) dari Data latih sebanyak 200 data. Dari grafik terlihat bahwa secara umum setiap kelompok data, baik getaran kereta (1 sampai 100 pada grafik) maupun bukan kereta (101 sampai 200 pada grafik) memiliki pola dengan kecenderungan yang sama. Pada gambar 4.4, data ke 1-100 menunjukkan getaran kereta api, 101-125 menunjukkan getaran Metromini, 126-150 menunjukkan getaran motor, 151-175 menjukkan getaran pejalan kaki dan 176-200 menunjukkan getaran truk.



Gambar 4.6 Ekstraksi ciri MFCC getaran kereta dan bukan kereta

Dari gambar 4.6 terlihat bahwa beberapa hasil ekstraksi ciri MFCC getaran truk kelihatan lebih besar daripada ekstraksi ciri MFCC getaran kereta. Hal ini bisa saja terjadi untuk dua jenis getaran yang berbeda untuk satu koefisien MFCC yang sama dari 13 parameter yang ada. Tapi akan berbeda untuk parameter yang lain.

Berikut adalah hasil perbesaran salah satu bagian grafik koefisien MFCC yang terdiri dari 13 koefisien (Data ke 130-142).



Gambar 4.7 Koefisien MFCC getaran bukan kereta

Hasil ekstraksi ciri Data Latih dengan menggunakan MFCC selanjutnya akan di latih pada jaringan yang akan dibangun. Data ini akan menjadi acuan sehingga jaringan yang akan dibangun mampu berjalan dengan baik ketika dilakukan simulasi atau pengujian data.

C. Klasifikasi Pola Getaran Kereta dengan Jaringan Saraf Tiruan

Proses klasifikasi pola Getaran dengan Jaringan Saraf Tiruan (*Artificial Neural Network*) secara umum meliputi ekstraksi ciri, pelatihan jaringan dan pengujian jaringan. Proses klasifikasi pada dasarnya mengelompokkan pola getaran yang sama (mirip) ke dalam kelompok yang sama berdasarkan pola getaran yang sudah dilatih pada jaringan saraf tiruan. Berikut tampilan antar muka

jaringan yang dibangu dengan MATLAB yang meliputi proses ekstraksi ciri, pelatihan dan pengujian.

Gambar 4.8 Tampilan awal aplikasi klasifikasi dengan ANN

Pada tahap ekstraksi ciri semua data latih akan diekstrak dengan metode MFCC untuk mendapatkan ciri atau pola khusus sebagai pembeda dengan sinyal lain.

Gambar 4.9 Tampilan antar muka ekstraksi ciri

Pada tahap pelatihan jaringan, algoritma yang digunakan adalah *Backpropagation* atau propagasi balik.

Pelatihan dilakukan pada data hasil ekstraksi ciri dengan metode MFCC. Beberapa parameter jaringan yang dibangun adalah sebagai berikut:

input	13
Hidden layer	2
Neuron Hidden Layer	10-100
Neuron Output	1
Maksimum Epoch	1000
Performance Goal	1e-6
Fungsi *Transfer Hidden Layer*	tansig
Fungsi *Transfer Output Layer*	tansig

Pelatihan dengan algoritma propagasi balik merupakan salah satu proses pembelajaran yang terawasi. Pada penelitian ini digunakan dua target dalam proses pembelajaran. Getaran kereta diarahkan pada target angka 1 pada proses pembelajaran sedangkan getaran bukan kereta diarahkan pada target angka 0. Artinya setiap data yang mempunyai pola yang mirip dengan pola kelompok data latih getaran kereta akan diidentifikasi sebagai kelompok angka 1. Begitu juga dengan target angka 0, setiap data yang mempunyai pola yang mirip dengan pola kelompok data latih bukan kereta akan diidentifikasi sebagai kelompok angka 0.

Keluaran dari pelatihan ini diharapkan agar hasil simulasi data latih menghasilkan angka yang sama (hampir sama) dengan target yang sudah diarahkan. Hal ini bertujuan agar akurasi jaringan semakin bagus.

Pada penelitian ini pelatihan jaringan dilakukan dengan variasi jumlah neuron dimulai dari 10, 20, 30, 40, 50, 60, 70, 80, 90 dan 100..

Simulasi Jaringan

Pada tahap ini data akan disimulasikan pada jaringan untuk menghitung akurasi jaringan. Perhitungan akurasi jaringan menggunakan metode *Confusion*

Matrix. Confusion Matriks adalah salah satu metode untuk menghitung akurasi, presisi, dan tingkat kesalahan pada konsep data *mining.*

Dalam bidang pengenalan pola (*pattern recognition*), ada beberapa perhitungan yang digunakan untuk mengukur kinerja dari sistem atau metode yang digunakan. Salah satunya adalah *confusion matrix. Confusion matrix* adalah suatu metode yang digunakan untuk menghitung akurasi, presisi dan tingkat kesalahan dalam suatu sistem.

Presisi adalah tingkat ketepatan antara informasi yang diminta oleh pengguna dengan jawaban yang diberikan oleh sistem. Akurasi didefenisikan sebagai tingkat kedekatan antara nilai prediksi dengan nilai aktual. Sedangkan tingkat kesalahan menyatakan tingkat kesalahan dalam identifikasi kasus dari keseluruhan kasus.

		Nilai Sebenarnya	
		TRUE	***FALSE***
Nilai Prediksi	***TRUE***	**TP** (*True Positive*) *Corect result*	**FP** (*False Positive*) *Unexpected result*
	FALSE	**FN** (*False Negative*) *Missing result*	**TN** (*True Negative*) *Corect absence of result*

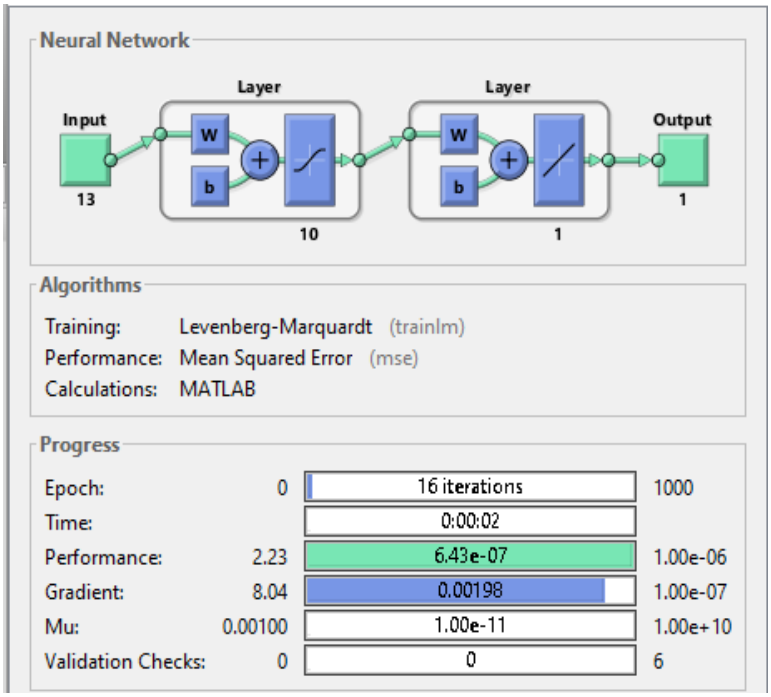
$$Presisi = \frac{TP}{TP+FP} \tag{4.1}$$

$$Akurasi = \frac{TP+TN}{TP+TN+FP+FN} \tag{4.2}$$

$$Tingkat\ Kesalahan = \frac{FP+FN}{FP+FN+TP+TN} \tag{4.3}$$

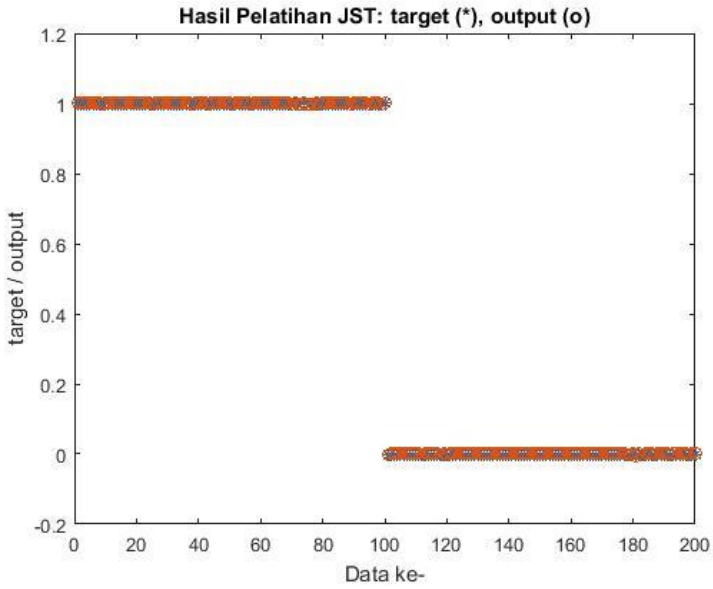
1. Klasifikasi dengan 10 *Hidden Neuron*

Pada tahap pelatihan ANN dengan jumlah *hidden neuron* 10, jaringan dilatih dengan data latih sebanyak 200 data yang terbagi menjadi 100 getaran kereta dan 100 getaran non kereta. Pelatihan berlangsung sampai pada iterasi ke 16 dengan nilai *Training Performance* 6.43 x 10^{-7} dari maksimal 10^{-6}. Artinya nilainya jauh lebih kecil dari batas maksimal sehingga dapat dikatakan bahwa proses pelatihan cukup cepat dan efektif.



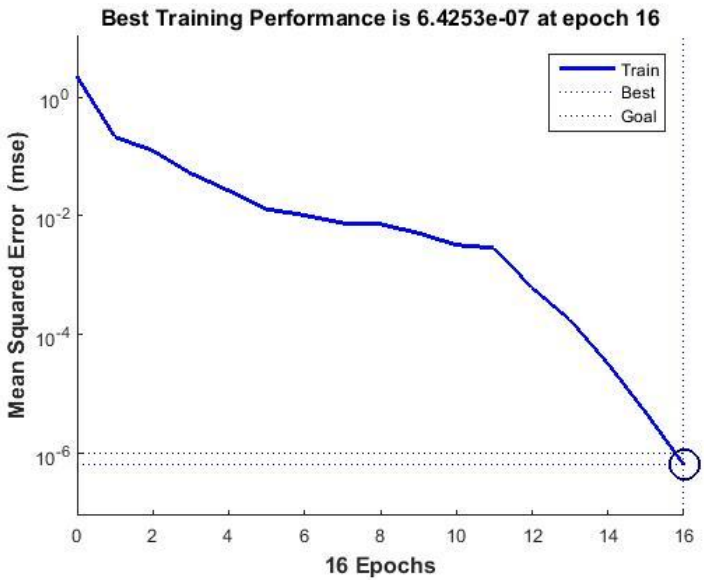
Gambar 4.10 Tampilan proses pelatihan ANN dengan *hidden neuron* 10

Gambar 4.10 menunjukkan proses pelatihan sedang berlangsung. Disini akan ditampilkan informasi tentang jaringan saraf tiruan yang sedang dibangun.



Gambar 4.11 Simulasi data latih pada ANN dengan 10 *hidden neuron*

Gambar 4.11 menunjukkan bahwa nilai simulasi dan target memiliki nilai yang sama (hampir sama). Semakin kecil selisih (*error*) antara hasil simulasi dan target yang diarahkan, maka hasil pelatihan dapat dianggap baik.



Gambar 4.12 *Mean Squared Error* ANN dengan 10 *hidden neuron*

Gambar 4.12 menunjukkan bahwa proses pelatihan berhenti pada iterasi (*epoch*) yang ke 16 dengan nilai *Training Performance* 6.4253 x 10^{-7} dari maksimal yang diberikan pada saat pelatihan adalah 1e-6. Artinya nilainya jauh

lebih kecil dari batas maksimal. Dapat dikatakan bahwa proses pelatihan cukup cepat dan efektif.

Hasil pengujian klasifikasi ANN dengan 10 *hidden neuron* dengan *confusion matrix* adalah sebagai berikut:

Getaran Kereta = 60 Getaran Bukan Kereta=60		**Nilai Sebenarnya**	
		TRUE	***FALSE***
Nilai Prediksi	***TRUE***	**59**	**4**
	FALSE	**1**	**56**

Gambar 4.13 Perhitungan Akurasi dengan *confusion matrix*

Sehingga diperoleh nilai presisi 93.65% dan akurasi 95.83%

2. Klasifikasi dengan 20 *Hidden Neuron*

Pelatihan berlangsung sampai pada iterasi ke 47 dengan nilai *Training Performance* 3.13 x 10^{-8} .

Hasil pengujian klasifikasi ANN dengan 20 *hidden neuron* dengan *confusion matrix* adalah sebagai berikut:

Getaran Kereta = 60 Getaran Bukan Kereta=60		**Nilai Sebenarnya**	
		TRUE	***FALSE***
Nilai Prediksi	***TRUE***	**60**	**11**
	FALSE	**0**	**49**

Gambar 4.14 Perhitungan Akurasi dengan *confusion matrix*

Sehingga diperoleh nilai presisi 84.51% dan akurasi 90.83%

3. Klasifikasi dengan 30 *Hidden Neuron*

Pelatihan berlangsung sampai pada iterasi ke 37 dengan nilai *Training Performance* 1.96×10^{-7} .

Hasil pengujian klasifikasi ANN dengan 30 *hidden neuron* dengan *confusion matrix* adalah sebagai berikut:

Getaran Kereta = 60 Getaran Bukan Kereta=60		**Nilai Sebenarnya**	
		TRUE	***FALSE***
Nilai Prediksi	***TRUE***	**59**	**2**
	FALSE	**1**	**48**

Gambar 4.15 Perhitungan Akurasi dengan *confusion matrix*

Sehingga diperoleh nilai presisi 96.72% dan akurasi 89.17 %

4. Klasifikasi dengan 40 *Hidden Neuron*

Pelatihan berlangsung sampai pada iterasi ke 10 dengan nilai *Training Performance* 4.15×10^{-8} .

Hasil pengujian klasifikasi ANN dengan 40 *hidden neuron* dengan *confusion matrix* adalah sebagai berikut:

Getaran Kereta = 60 Getaran Bukan Kereta=60		**Nilai Sebenarnya**	
		TRUE	***FALSE***
Nilai Prediksi	***TRUE***	**52**	**12**
	FALSE	**8**	**48**

Gambar 4.16 Perhitungan Akurasi dengan *confusion matrix*

Sehingga diperoleh nilai presisi 81.25% dan akurasi 83.33%

5. Klasifikasi dengan 50 *Hidden Neuron*

Pelatihan berlangsung sampai pada iterasi ke 19 dengan nilai *Training Performance* 1.04 x 10^{-7} .

Hasil pengujian klasifikasi ANN dengan 50 *hidden neuron* dengan *confusion matrix* adalah sebagai berikut:

Getaran Kereta = 60 Getaran Bukan Kereta=60		**Nilai Sebenarnya**	
		TRUE	***FALSE***
Nilai Prediksi	***TRUE***	**52**	**17**
	FALSE	**8**	**43**

Gambar 4.17 Perhitungan Akurasi dengan *confusion matrix*

Sehingga diperoleh nilai presisi 75.36% dan akurasi 79.17 %

6. Klasifikasi dengan 60 *Hidden Neuron*

Pelatihan berlangsung sampai pada iterasi ke 17 dengan nilai *Training Performance* 1.05 x 10^{-7}.

Hasil pengujian klasifikasi ANN dengan 60 *hidden neuron* dengan *confusion matrix* adalah sebagai berikut:

Getaran Kereta = 60 Getaran Bukan Kereta=60		**Nilai Sebenarnya**	
		TRUE	***FALSE***
Nilai Prediksi	***TRUE***	**55**	**13**
	FALSE	**5**	**47**

Gambar 4.18 Perhitungan Akurasi dengan *confusion matrix*

Sehingga diperoleh nilai presisi 80.88% dan akurasi 85 %

7. Klasifikasi dengan 70 *Hidden Neuron*

Pelatihan berlangsung sampai pada iterasi ke 19 dengan nilai *Training Performance* 2.82 x 10^{-9} .

Hasil pengujian klasifikasi ANN dengan 70 *hidden neuron* dengan *confusion matrix* adalah sebagai berikut:

Getaran Kereta = 60 Getaran Bukan Kereta=60		**Nilai Sebenarnya**	
		TRUE	***FALSE***
Nilai Prediksi	***TRUE***	**54**	**25**
	FALSE	**6**	**35**

Gambar 4.19 Perhitungan Akurasi dengan *confusion matrix*

Sehingga diperoleh nilai presisi 68.35% dan akurasi 74.17 %

8. Klasifikasi dengan 80 *Hidden Neuron*

Pelatihan berlangsung sampai pada iterasi ke 16 dengan nilai *Training Performance* 2.73 x 10^{-7} .

Hasil pengujian klasifikasi ANN dengan 80 *hidden neuron* dengan *confusion matrix* adalah sebagai berikut:

Getaran Kereta = 60 Getaran Bukan Kereta=60		**Nilai Sebenarnya**	
		TRUE	***FALSE***
Nilai Prediksi	***TRUE***	**50**	**24**
	FALSE	**10**	**36**

Gambar 4.20 Perhitungan Akurasi dengan *confusion matrix*

Sehingga diperoleh nilai presisi 67.57% dan akurasi 71.67 %

9. Klasifikasi dengan 90 *Hidden Neuron*

Pelatihan berlangsung sampai pada iterasi ke 17 dengan nilai *Training Performance* 3.67 x 10^{-7} .

Hasil pengujian klasifikasi ANN dengan 90 *hidden neuron* dengan *confusion matrix* adalah sebagai berikut:

Getaran Kereta = 60 Getaran Bukan Kereta=60		**Nilai Sebenarnya**	
		TRUE	***FALSE***
Nilai Prediksi	***TRUE***	**54**	**20**
	FALSE	**6**	**40**

Gambar 4.21 Perhitungan Akurasi dengan *confusion matrix*

Sehingga diperoleh nilai presisi 72.97% dan akurasi 78.33 %

10. Klasifikasi dengan 100 *Hidden Neuron*

Pelatihan berlangsung sampai pada iterasi ke 13 dengan nilai *Training Performance* 6.49 x 10^{-7} .

Hasil pengujian klasifikasi ANN dengan 100 *hidden neuron* dengan *confusion matrix* adalah sebagai berikut:

Getaran Kereta = 60 Getaran Bukan Kereta=60		**Nilai Sebenarnya**	
		TRUE	***FALSE***
Nilai Prediksi	***TRUE***	**55**	**11**
	FALSE	**5**	**49**

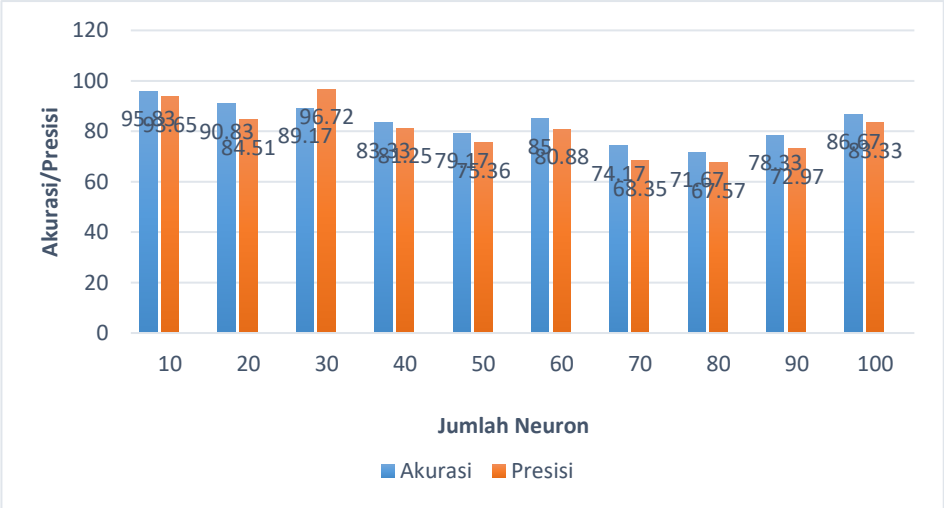
Gambar 4.22 Perhitungan Akurasi dengan *confusion matrix*

Sehingga diperoleh nilai presisi 83.33% dan akurasi 86.67 %

D. Analisis Hasil Pengujian

Berikut adalah tabel akurasi jaringan saraf tiruan dengan variasi jumlah neuron:

No	Jumlah Neuron	Akurasi	Presisi
1	10	95.83 %	93.65 %
2	20	90.83 %	84.51 %
3	30	89.17 %	96.72 %
4	40	83.33%	81.25 %
5	50	79.17 %	75.36 %
6	60	85 %	80.88 %
7	70	74.17 %	68.35 %
8	80	71.67 %	67.57 %
9	90	78.33 %	72.97 %
10	100	86.67 %	83.33 %



Gambar 4.23 Grafik Akurasi Jaringan Saraf Tiruan

Berdasarkan perhitungan *Confusin Matrix* di atas diperoleh akurasi jaringan tertinggi sebesar 95.83 % dengan jumlah *hidden neuron* 10. Dari perhitungan juga terlihat presisi tertinggi sebesar 96.72 % dengan *jumlah hidden* neuron 30.

BAB V
KESIMPULAN DAN SARAN

A. Kesimpulan

Berdasarkan hasil analisa dari penelitian yang telah dilakukan, dapat diambil beberapa kesimpulan sebagai berikut:

1. Ekstraksi Ciri dengan MFCC untuk sinyal getaran kereta dapat digunakan untuk klasifikasi dengan ANN secara efektif
2. Hasil eksperimen klasifikasi getaran kereta dengan MFCC memberikan akurasi tertinggi sebesar 95.83 ketika *hidden neuron* yang digunakan adalah 10 neuron.
3. Presisi tertinggi pada eksperimen klasifikasi getaran kereta dengan MFCC diperoleh pada *hidden neuron* 30 sebesar 96.72 %.

B. Saran

1. Penambahan jumlah data baik data latih maupun data uji akan meningkatkan akurasi jaringan saraf tiruan
2. Pemotongan data ketika dilakukan pengolahan data harus dilakukan secara hati-hati dan teliti agar data yang dihasilkan juga baik.
3. Pengujian data dapat dilakukan dengan beberapa variasi jenis data yang berbeda untuk mengetahui kemampuan jaringan saraf tiruan yang dibangun.

DAFTAR PUSTAKA

Abriyono, A. H. (2012). Pengenalan Ucapan Suku Kata Bahasa Lisan Menggunakan Ciri LPC,MFCC, dan JST. *IJCCS*, 1-11.

Aji Suroso, Y. F. (2015). Aplikasi Pengenalan Ucapan dengan Ekstraksi Ciri Mel Frequency Cepstrum Coefficients (MFCC) dan Jaringan Saraf tiruan. *Jurnal Komputer Terapan*, 1-11.

Angga Setiawan, A. H. (2011). Aplikasi Pengenalan Ucapan dengan Ekstraksi Mel-Frequency Cepstrum Coefficients (MFCC) Melalui Jaringan Syaraf Tiruan (JST) Learning Vector Quantization (LVQ) untuk Mengoperasikan Kursor Komputer. *Komputer Terapan*, 1-5.

Howard Demuth, M. B. (2002). *Neural Network Toolbox*. Natick: The MathWorks.

Isnanto R.R, H. A. (2011). *Aplikasi Pengenalan Ucapan dengan Ekstraksi Mel-Frequency Cepstrum Coefficients (MFCC) Melalui Jaringan Saraf Tiruan (JST) Learning Vector (LVQ).* Semarang: UNDIP.

Isnanto R.R, Y. I. (2012). *Sistem Identifikasi Jenis Suara Manusia Berdasarkan Jangkauan Vokal Menggunanakan Jaringan Saraf Tiruan Backpropagation.* Bandung: Universitas Telkom.

Mada Sanjaya W. S, Z. S. (2014). Implementasi Pengenalan Pola Suara Menggunakan Mel-Frequency Cepstrum Coefficients (MFCC) dan Adaptive Neuro-Fuzzy Inferense System(ANFIS) Sebagai Kontrol Lampu Otomatis. *Al-HAZEN Jurnal of Physic*, 1-9.

Mustofa, A. (2007). Sistem Pengenalan Penutur dengan Mel-Frequency Wrapping. *Jurnal Teknik Elektro*, 1-8.

Putra, D. (2011). Verifikasi Biometrika Suara Menggunakan Metode MFCC dan DTW. *Lontar Komputer*, 1-13.

Rifan Muhamad Fauzi, A. W. (2013). Pengenalan Ucapan Huruf Hujaiyah Menggunakan Mel Frequency Cepstral Coefficients(MFCC) dan Hidden Markov Model (HMM). 1-16.

Rohman, A. Z. (2015). *Rancang Bangun Alat Ukur Getaran Mesin Berbasis Arduino.* Semarang: Universitas Negeri Semarang.

Siang, J. J. (2005). *Jaringan Syaraf Tiruan dan Pemrogramannya Menggunakan Matlab.* Yogyakarta: ANDI.

Siswoyo, F. (2013). *Pengenalan Chord Pada Gitar dengan MFCC Sebagai Metode Ekstraksi Ciri dan Jaringan Saraf Tiruan Sebagai Metode Pengenalan Pola.* Bogor: IPB.

Tanjung, D. H. (2015). Jaringan Saraf Tiruan dengan Backpropagation untuk Memprediksi Penyakit Asma. *Citec Journal*, 1-10.

Theodoros Giannakopoulus, A. P. (2014). *Introduction to Audio Analysis: A MATLAB Approach* . USA: The MathWorks.

Tomi Budi Waluyo, A. S. (2015). Penggunaan Akselerometer Pada Ponsel Android untuk Merekam Getaran Kabin Kereta Api. *Annual Meeting on Testing and Quality 2015 LIPI*, 1-10.

Veronica Indrawati, Y. G. (2014). Penggunaan Algoritma Learning Vector Quantization Dalam Mengenali Suara Untuk Kendali Quadrotor. *SENTIKA*, 1-4.

Zakaria Ramadhan, S. N. (2016). Perintah Suara Berbahasa Indonesia untuk Membuka dan Menutup Aplikasi dalam Sistem Operasi Windows Menggunakan Metode Mel Frequency Cepstrum Coefficients dan Metode Backpropagation. *Seminar Nasional Ilmu Komputer*, 1-9.

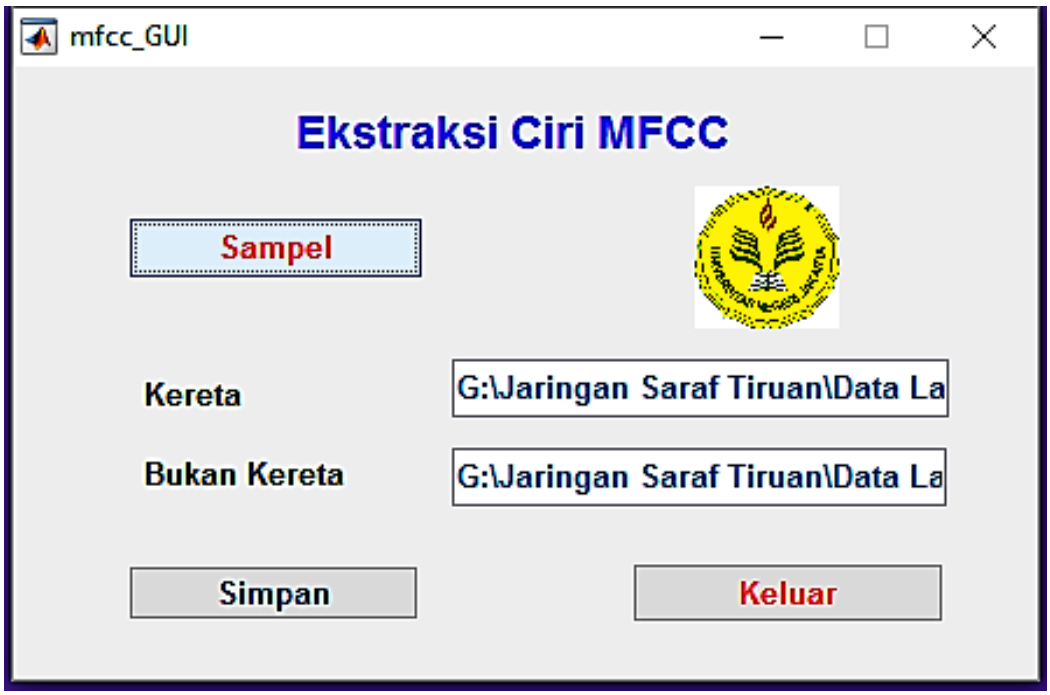
LAMPIRAN

LAMPIRAN

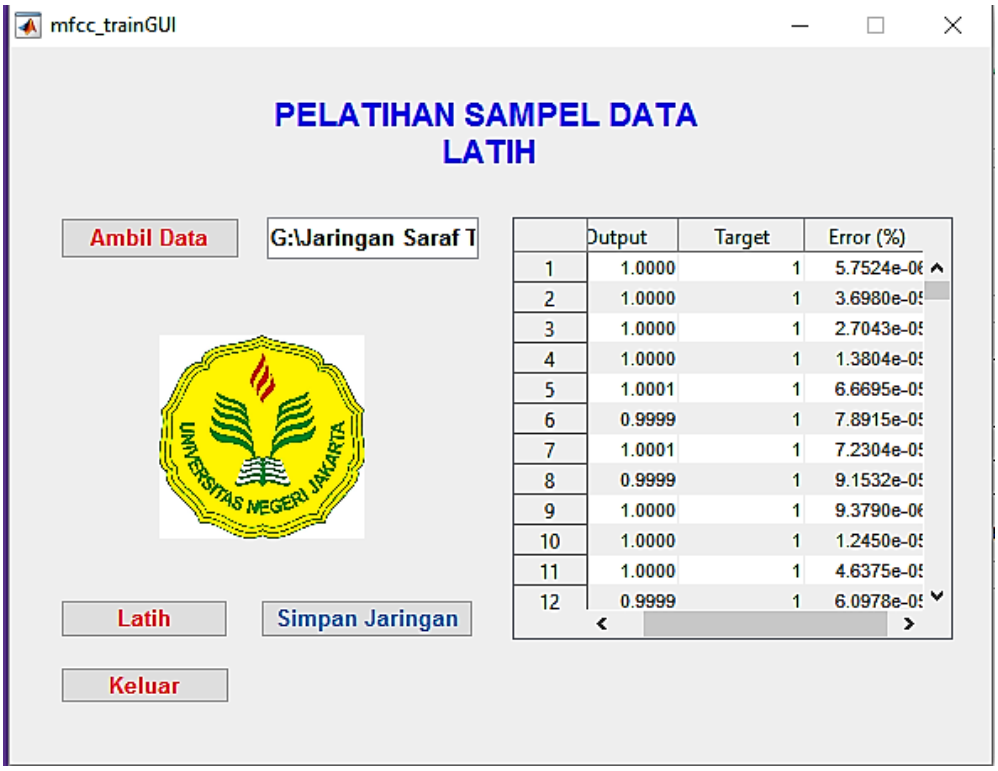
Lampiran 1. Tampilan Awal Sistem Jaringan

(a)

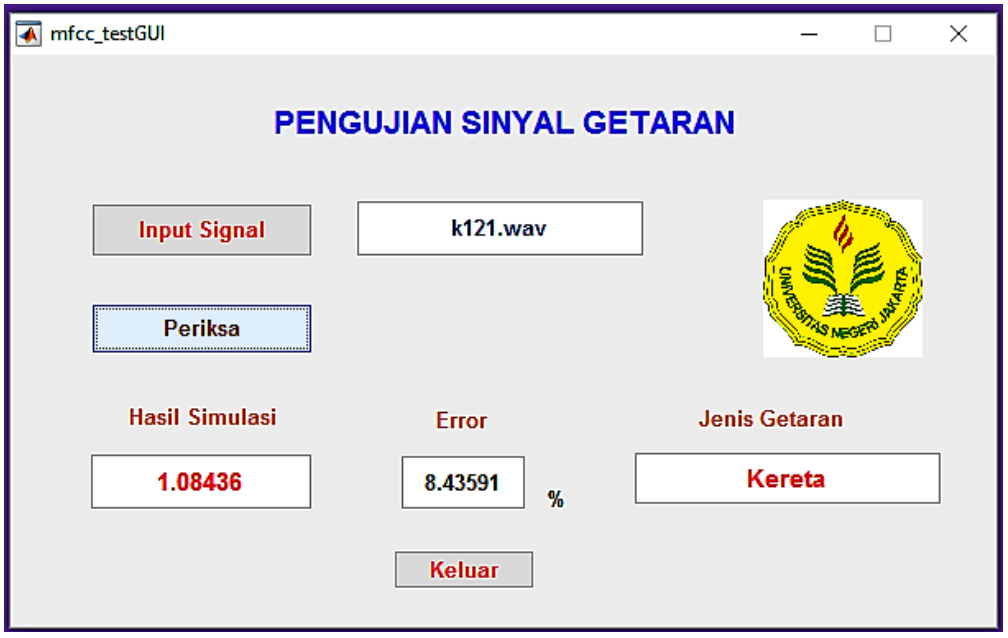
(b)



(c)



(d)



Gambar 1. (a)Tamilan dasar jaringan; (b)Tampilan dasar ekstraksi ciri; (c)Tampilan dasar pelatihan jaringan; (d)Tampilan dasar pengujian

Lampiran 2. Kode Program Estraksi Ciri MFCC

```
function varargout = mfcc_GUI(varargin)
% MFCC_GUI MATLAB code for mfcc_GUI.fig
%      MFCC_GUI, by itself, creates a new MFCC_GUI or raises the
existing
%      singleton*.
%
%      H = MFCC_GUI returns the handle to a new MFCC_GUI or the
handle to
%      the existing singleton*.
%
%      MFCC_GUI('CALLBACK',hObject,eventData,handles,...) calls
the local
%      function named CALLBACK in MFCC_GUI.M with the given input
arguments.
%
%      MFCC_GUI('Property','Value',...) creates a new MFCC_GUI or
raises the
%      existing singleton*.  Starting from the left, property
value pairs are
%      applied to the GUI before mfcc_GUI_OpeningFcn gets called.
An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to mfcc_GUI_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows
only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help mfcc_GUI

% Last Modified by GUIDE v2.5 08-Jun-2017 18:24:50

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @mfcc_GUI_OpeningFcn, ...
                   'gui_OutputFcn',  @mfcc_GUI_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
```

```
% End initialization code - DO NOT EDIT

% --- Executes just before mfcc_GUI is made visible.
function mfcc_GUI_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to mfcc_GUI (see VARARGIN)

% Choose default command line output for mfcc_GUI
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes mfcc_GUI wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = mfcc_GUI_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

img=imread('logo.jpg');
axes(handles.axes1)
imshow(img)
axis off
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
signal_folder = uigetdir;

warndlg('Ekstraksi ciri Getaran Kereta sedang
berlangsung','Loading..');

set(handles.edit1,'String',signal_folder);
filenames = dir(fullfile(signal_folder, '*.wav'));
total_signal = numel(filenames);

for n = 1:total_signal
    full_name= fullfile(signal_folder, filenames(n).name);
```

```
    AI = audioread(full_name);
    fs=12000;

%% Computing MFCC Co-efficients..
    %% (1) Frame Blocking..
    N = 2*256;   % N point FFT
    M = 2*128;   % Overlapping
    NN = floor(N/2+1); %N/2
    nbFrames = ceil((length(AI)-N)/M);
    Frames = zeros(nbFrames+1,N);
    for i = 0:nbFrames-1
        temp = AI(i*M+1:i*M+N);
        Frames(i+1,1:N) = temp;
    end

    % Last Frame..
    temp = zeros(1,N);
    lastLength = length(AI)- nbFrames*M;
    temp(1:lastLength) = AI(nbFrames*M+1:(nbFrames*M +1 +
lastLength-1));
    Frames(nbFrames+1, 1:N) = temp;
    %% (2) Windowing..
    frameSize = size(Frames);
    nbFrames = frameSize(1);
    nbSamples = frameSize(2);

    % Hamming window..
    w = hamming(nbSamples);
    Windows = zeros(nbFrames,nbSamples);
    for i = 1:nbFrames
        temp = Frames(i,1:nbSamples);
        Windows(i, 1:nbSamples) = w'.*temp;
    end
    %% (3) Fourier Transform..
    ffts = fft(Windows');
    %% (4) Mel-frequency Wrapping..
    % (a) Calculate Power spectrum..
    PowSpecs = abs(ffts).^2;
    PowSpecs = PowSpecs(1:NN-1,:);
    % (b) Mel filter generation
    nof_c = 14; % Number of channels..
    df = fs/N;
    Nmax = N/2;
    fmax = fs/2;

    % Convert to mel scale..
    melmax = 2595*log10(1+fmax/700);

    melinc = melmax/(nof_c+1);

    melcenters = (1:nof_c).*melinc;

    % Convert to frequency scale..
    fcenters = 700*((10.^(melcenters./2595))-1);
```

```
    centerf = round(fcenters./df);

    startf = [1,centerf(1:nof_c-1)];
    stopf = [centerf(2:nof_c),Nmax];

    W = zeros(nof_c,Nmax);

    % Making filter..
    for i = 1:nof_c
        increment = 1.0/(centerf(i)-startf(i));
        for j = startf(i):centerf(i)
            W(i,j) = (j-startf(i))*increment;
        end

        decrement = 1.0/(stopf(i)-centerf(i));
        for j = centerf(i):stopf(i)
            W(i,j) = (j-centerf(i))*decrement;
        end
    end
    % Normalising..
    for i = 1:nof_c
        W(i,:) = W(i,:)/sum(W(i,:));
    end

    % (c) Apply mel filters to Power spectrum coeffs..
    melPowSpecs = W*PowSpecs;
    % (d) MFCC calculations..
    melCeps = dct(log(melPowSpecs));
    melCeps(1,:) = [];
    melCeps = mean(transpose(melCeps));

    koefisien_latih(n,:) = melCeps;

    k = koefisien_latih;

end
%%

t1 = ones(1,total_signal);

warndlg('Ekstraksi Ciri Getaran Kereta Selesai. Lanjut ke Folder
Selanjutnya');

signal_folder = uigetdir;

warndlg('Ekstraksi Ciri Getaran Bukan Kereta Sedang
berlangsung','Loading..');

set(handles.edit2,'String',signal_folder);
filenames = dir(fullfile(signal_folder, '*.wav'));
total_signal = numel(filenames);

for n = 1:total_signal
```

```
    full_name= fullfile(signal_folder, filenames(n).name);
    AI = audioread(full_name);
    fs=8000;

%% Computing MFCC Co-efficients..
    %% (1) Frame Blocking..
    N = 5*256;    % N point FFT
    M = 5*128;    % Overlapping
    NN = floor(N/2+1); %N/2
    nbFrames = ceil((length(AI)-N)/M);
    Frames = zeros(nbFrames+1,N);
    for i = 0:nbFrames-1
        temp = AI(i*M+1:i*M+N);
        Frames(i+1,1:N) = temp;
    end

    % Last Frame..
    temp = zeros(1,N);
    lastLength = length(AI)- nbFrames*M;
    temp(1:lastLength) = AI(nbFrames*M+1:(nbFrames*M +1 +
lastLength-1));
    Frames(nbFrames+1, 1:N) = temp;
    %% (2) Windowing..
    frameSize = size(Frames);
    nbFrames = frameSize(1);
    nbSamples = frameSize(2);

    % Hamming window..
    w = hamming(nbSamples);
    Windows = zeros(nbFrames,nbSamples);
    for i = 1:nbFrames
        temp = Frames(i,1:nbSamples);
        Windows(i, 1:nbSamples) = w'.*temp;
    end
    %% (3) Fourier Transform..
    ffts = fft(Windows');
    %% (4) Mel-frequency Wrapping..
    % (a) Calculate Power spectrum..
    PowSpecs = abs(ffts).^2;
    PowSpecs = PowSpecs(1:NN-1,:);
    % (b) Mel filter generation
    nof_c = 14; % Number of channels..
    df = fs/N;
    Nmax = N/2;
    fmax = fs/2;

    % Convert to mel scale..
    melmax = 2595*log10(1+fmax/700);

    melinc = melmax/(nof_c+1);

    melcenters = (1:nof_c).*melinc;

    % Convert to frequency scale..
    fcenters = 700*((10.^(melcenters./2595))-1);
```

```
    centerf = round(fcenters./df);

    startf = [1,centerf(1:nof_c-1)];
    stopf = [centerf(2:nof_c),Nmax];

    W = zeros(nof_c,Nmax);

    % Making filter..
    for i = 1:nof_c
        increment = 1.0/(centerf(i)-startf(i));
        for j = startf(i):centerf(i)
            W(i,j) = (j-startf(i))*increment;
        end

        decrement = 1.0/(stopf(i)-centerf(i));
        for j = centerf(i):stopf(i)
            W(i,j) = (j-centerf(i))*decrement;
        end
    end
    % Normalising..
    for i = 1:nof_c
        W(i,:) = W(i,:)/sum(W(i,:));
    end

    % (c) Apply mel filters to Power spectrum coeffs..
    melPowSpecs = W*PowSpecs;
    % (d) MFCC calculations..
    melCeps = dct(log(melPowSpecs));
    melCeps(1,:) = [];
    melCeps = mean(transpose(melCeps));

    koefisien_latih(n,:) = melCeps;

    bk = koefisien_latih;

end
%%

t2 = 2*ones(1,total_signal);

warndlg('Ekstraksi Ciri Getaran Bukan Kereta Selesai');

koefisien_latih = [k;bk];
target = [t1 t2];
koefisien_latih = koefisien_latih(1:length(target),:);
koefisien_latih = [koefisien_latih target']

save koefisien_latih

warndlg('Hasil disimpan dengan nama: koefisien_latih.mat');
clc
```

```
function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of
edit1 as a double

% --- Executes during object creation, after setting all
properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of
edit2 as a double

% --- Executes during object creation, after setting all
properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

warndlg('Hasil Ekstraksi Ciri Sudah Disimpan sebagai
koefisien_latih.mat','Berhasil..');

% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
delete(handles.figure1)
```

Sumber : http://www.edaboard.com/thread275114.html

Lampiran 3. Kode Program Pembuatan dan Pelatihan Jaringan Saraf Tiruan

```
function varargout = mfcc_trainGUI(varargin)
%MFCC_TRAINGUI M-file for mfcc_trainGUI.fig
%      MFCC_TRAINGUI, by itself, creates a new MFCC_TRAINGUI or
raises the existing
%      singleton*.
%
%      H = MFCC_TRAINGUI returns the handle to a new MFCC_TRAINGUI
or the handle to
%      the existing singleton*.
%
%      MFCC_TRAINGUI('Property','Value',...) creates a new
MFCC_TRAINGUI using the
%      given property value pairs. Unrecognized properties are
passed via
%      varargin to mfcc_trainGUI_OpeningFcn.  This calling syntax
produces a
%      warning when there is an existing singleton*.
%
%      MFCC_TRAINGUI('CALLBACK') and
MFCC_TRAINGUI('CALLBACK',hObject,...) call the
%      local function named CALLBACK in MFCC_TRAINGUI.M with the
given input
%      arguments.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows
only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help mfcc_trainGUI

% Last Modified by GUIDE v2.5 08-Jun-2017 19:04:19

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @mfcc_trainGUI_OpeningFcn,
...
                   'gui_OutputFcn',  @mfcc_trainGUI_OutputFcn, ...
                   'gui_LayoutFcn',  [], ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
```

```
end
% End initialization code - DO NOT EDIT

% --- Executes just before mfcc_trainGUI is made visible.
function mfcc_trainGUI_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   unrecognized PropertyName/PropertyValue pairs from
the
%            command line (see VARARGIN)

% Choose default command line output for mfcc_trainGUI
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes mfcc_trainGUI wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = mfcc_trainGUI_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

img=imread('logo.jpg');
axes(handles.axes1)
imshow(img)
axis off

% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

[filename, pathname]=uigetfile({'*.mat', 'Feat File'; }, 'Pilih
data latih');
set(handles.edit1,'String',fullfile(pathname,filename));

function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of
edit1 as a double

% --- Executes during object creation, after setting all
properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

data=load(get(handles.edit1,'String'));
num=data.koefisien_latih;
input=num(:, 1:(size(num,2)-1));
target=num(:,end);
assignin('base','input', input);
assignin('base','target', target);

handles.ids=num(:, end)';
handles.data_matrix=input;
assignin('base', 'data_matrix', handles.data_matrix);
assignin('base', 'ids', handles.ids);
handles.semua_data=num;
handles.input=input;
guidata(hObject, handles);

input = input';
target = target';
% Membangun JST
nb1 = 30;
nb2 = 1;
nb = [nb1 nb2];
net =
newff(minmax(input),nb,{'tansig','tansig','purelin'},'trainlm');
%%
```

```
bobotawal_input=net.IW(1,1)
bobotawal_biasinput=net.b(1,1)
bobotawal_lapisan1=net.LW(2,1)
bobotawal_biaslapisan1=net.b(2,1)
%%
% Memberikan nilai untuk mempengaruhi proses pelatihan
net.performFcn = 'mse';
net.trainParam.goal = 1e-6;
net.trainParam.show = 1;
net.trainParam.epochs = 1000;
net.trainParam.mc = 0.95;
net.trainParam.lr = 0.1;

%%
% Proses training
net = train(net,input,target);
%[network,tr,Out,E] = train(net,input,target);

bobotakhir_input=net.IW(1,1)
bobotakhir_biasinput=net.b(1,1)
bobotakhir_lapisan1=net.LW(2,1)
bobotakhir_biaslapisan1=net.b(2,1)
output = sim(net,input);
save net net

hasil = [output;target;abs(target-output)]';

error = target-output;

figure(1)
plot((1:length(target)),target,'*',(1:length(target)),output,'o')
title('Hasil Pelatihan JST: target (*), output (o)');
xlabel('Data ke-');ylabel('target / output')

figure(2)
plot(target,output);
title('Hasil Pelatihan');
xlabel('Target');ylabel('Output')

set(handles.uitable1,'data',hasil);
clc

warndlg('Pelatihan Data Sampel Sudah Selesai',' SELESAI')

% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
load net
[filename, path]=uiputfile('*.mat','Simpan Jaringan');
if isequal(filename,0)
    return
```

```
end
save(fullfile([path,filename]),'net')

% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
delete(handles.figure1)
```

Sumber : https://pemrogramanmatlab.com/2016/10/24/jaringan-syaraf-tiruan-untuk-klasifikasi-citra-daun/

Lampiran 4. Kode Program Simulasi Jaringan Saraf Tiruan

```
function varargout = mfcc_testGUI(varargin)
% MFCC_TESTGUI MATLAB code for mfcc_testGUI.fig
%      MFCC_TESTGUI, by itself, creates a new MFCC_TESTGUI or
raises the existing
%      singleton*.
%
%      H = MFCC_TESTGUI returns the handle to a new MFCC_TESTGUI
or the handle to
%      the existing singleton*.
%
%      MFCC_TESTGUI('CALLBACK',hObject,eventData,handles,...)
calls the local
%      function named CALLBACK in MFCC_TESTGUI.M with the given
input arguments.
%
%      MFCC_TESTGUI('Property','Value',...) creates a new
MFCC_TESTGUI or raises the
%      existing singleton*.  Starting from the left, property
value pairs are
%      applied to the GUI before mfcc_testGUI_OpeningFcn gets
called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to mfcc_testGUI_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows
only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help mfcc_testGUI

% Last Modified by GUIDE v2.5 11-Jul-2017 19:04:29

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @mfcc_testGUI_OpeningFcn, ...
                   'gui_OutputFcn',  @mfcc_testGUI_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
```

```
% End initialization code - DO NOT EDIT

% --- Executes just before mfcc_testGUI is made visible.
function mfcc_testGUI_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to mfcc_testGUI (see VARARGIN)

% Choose default command line output for mfcc_testGUI
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes mfcc_testGUI wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = mfcc_testGUI_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

img=imread('logo.jpg');
axes(handles.axes1)
imshow(img)
axis off

% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
[nama_file,nama_path] = uigetfile({'*.*'});
set(handles.edit3,'String',nama_file);

if ~isequal(nama_file,0)
    I = audioread(fullfile(nama_path,nama_file));
    handles.I = I;
    guidata(hObject,handles)
else
    return
end
```

```
% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
I = handles.I;
fs=12000;

%% Computing MFCC Co-efficients..
    %% (1) Frame Blocking..
    N = 2*256;   % N point FFT 256 100
    M = 2*128;   % Overlapping

    NN = floor(N/2+1); %N/2
    nbFrames = ceil((length(I)-N)/M);
    Frames = zeros(nbFrames+1,N);
    for i = 0:nbFrames-1
        temp = I(i*M+1:i*M+N);
        Frames(i+1,1:N) = temp;
    end

    % Last Frame..
    temp = zeros(1,N);
    lastLength = length(I)- nbFrames*M;
    temp(1:lastLength) = I(nbFrames*M+1:(nbFrames*M +1 +
lastLength-1));
    Frames(nbFrames+1, 1:N) = temp;
    %% (2) Windowing..
    frameSize = size(Frames);
    nbFrames = frameSize(1);
    nbSamples = frameSize(2);

    % Hamming window..
    w = hamming(nbSamples);
    Windows = zeros(nbFrames,nbSamples);
    for i = 1:nbFrames
        temp = Frames(i,1:nbSamples);
        Windows(i, 1:nbSamples) = w'.*temp;
    end
    %% (3) Fourier Transform..
    ffts = fft(Windows');
    %% (4) Mel-frequency Wrapping..
    % (a) Calculate Power spectrum..
    PowSpecs = abs(ffts).^2;
    PowSpecs = PowSpecs(1:NN-1,:);
    % (b) Mel filter generation
    nof_c = 14; % Number of channels..
    df = fs/N;
    Nmax = N/2;
    fmax = fs/2;

    % Convert to mel scale..
    melmax = 2595*log10(1+fmax/700);
```

```
melinc = melmax/(nof_c+1);

melcenters = (1:nof_c).*melinc;

% Convert to frequency scale..
fcenters = 700*((10.^(melcenters./2595))-1);

centerf = round(fcenters./df);

startf = [1,centerf(1:nof_c-1)];
stopf = [centerf(2:nof_c),Nmax];

W = zeros(nof_c,Nmax);

% Making filter..
for i = 1:nof_c
    increment = 1.0/(centerf(i)-startf(i));
    for j = startf(i):centerf(i)
        W(i,j) = (j-startf(i))*increment;
    end

    decrement = 1.0/(stopf(i)-centerf(i));
    for j = centerf(i):stopf(i)
        W(i,j) = (j-centerf(i))*decrement;
    end
end
% Normalising..
for i = 1:nof_c
    W(i,:) = W(i,:)/sum(W(i,:));
end

% (c) Apply mel filters to Power spectrum coeffs..
melPowSpecs = W*PowSpecs;
% (d) MFCC calculations..
melCeps = dct(log(melPowSpecs));
melCeps(1,:) = [];
melCeps = transpose(melCeps);
melCeps = mean(melCeps(:,1:13));

input = melCeps';

%loading JST
load net

output = sim(net,input);

if round(output)==1
    kelas = 'Kereta';
    error = abs(output-1)/1*100;
else
    kelas = 'Bukan Kereta';
    error = abs(output-2)/2*100;
end
```

```
    set(handles.edit1,'String',kelas);
    set(handles.edit2,'String',output);
    set(handles.edit4,'String',error);

function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of
edit1 as a double

% --- Executes during object creation, after setting all
properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of
edit2 as a double

% --- Executes during object creation, after setting all
properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
```

```
    set(hObject,'BackgroundColor','white');
end

function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of
edit3 as a double

% --- Executes during object creation, after setting all
properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit4_Callback(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of edit4 as text
%        str2double(get(hObject,'String')) returns contents of
edit4 as a double

% --- Executes during object creation, after setting all
properties.
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    empty - handles not created until after all
CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in pushbutton3.
```

```
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see GUIDATA)
delete(handles.figure1)
```

DAFTAR RIWAYAT HIDUP

Budiman Simbolon merupakan anak ke empat dari Ibu Nurmina Marbun dan Bapak Uler Simbolon yang lahir di Siantar, 12 April 1992.

Pendidikan formal dimulai dari SDN 173497 Karontang (1999 - 2005), SMP Katolik Karya Bhakti Tarabintang (2005 - 2008), SMA Santa Maria Pakkat (2008 - 2011) dan Program Studi Fisika FMIPA Universitas Negeri Jakarta (September 2012 – Agustus 2017). Semasa kuliah pernah menjadi asisten laboratorium Fisika Dasar 2, peserta olimpiade OSN-Pertamina dan menjadi pemakalah dalam Seminar Nasional Fisika Universitas Negeri Jakarta 2016. Pernah bekerja *part time* sebagai pengajar *private* untuk pelajaran Matematika untuk SD dan SMA. Pernah bekerjasama dengan Badan Meteorologi Klimatologi dan Geofisika (BMKG) di divisi instrumentasi dan kalibrasi untuk melaksanakan Praktik Kerja Lapangan pada tahun 2015.