KLASIFIKASI *MIXED TYPE AUDIO* MENGGUNAKAN *BACKPROPAGATION* DENGAN *FIXED SIZE MOMENTUM*

Dicky Teguh Satrio[1], Nelly Indriani W., S.Si., M.T.[2]

[1] Teknik Informatika – Universitas Komputer Indonesia
Jalan Dipatiukur No.112-116, Coblong, Bandung, Jawa Barat 40132
E-mail : dicky.teguh48@gmail.com[1], nelly.indriani@email.unikom.ac.id[2]

ABSTRAK

Mixed type audio adalah sebuah berkas audio yang memiliki lebih dari satu jenis sumber suara, seperti speech atau suara manusia yang digabung dengan musik, ataupun digabung dengan suara lingkungan sekitar. Pada penelitian *mixed type audio* sebelumnya memberikan kesimpulan bahwa *Support Vector Machines* dan *Backpropagation* menghasilkan akurasi paling baik yaitu *Support Vector Machines* 90,84% dan *Backpropagation* 90,07% untuk kelas speech dan musik, namun untuk kelas speech dan environment akurasi yang didapat adalah *Support Vector Machines* 64,76% dan *Backpropagation* 63,81%. Penelitian ini menunjukkan bagaimana mengklasifikasi *mixed type audio* menggunakan *backpropagation* dengan *fixed size momentum.* Penggunaan *fixed size momentum* dimaksudkan untuk meningkatkan akurasi dari metode *backpropagation*. Pengujian dilakukan dengan menguji beberapa parameter algoritma dengan beberapa variasi kondisi parameter. Parameter yang menghasilkan akurasi paling tinggi adalah dengan menggunakan 5 *neuron* pada *hidden layer, learning rate* 0.3, dan *windows size* 3. Adapun hasil yang didapat dari hasil pengujian beberapa parameter yang sudah optimal menghasilkan akurasi 95% menggunakan *backpropagation* dengan *fixed size momentum*. Berdasarkan hasil pengujian yang telah dilakukan, diperoleh kesimpulan bahwa klasifikasi *mixed type audio* menggunakan *backpropagation* dengan *fixed size momentum* dapat meningkatkan akurasi sebesar 95% jika dibandingkan dengan *backpropagation* saja yang mendapatkan akurasi sebesar 80% untuk klasifikasi kelas speech dengan environment dan speech dengan musik.

Kata kunci : *Mixed Type Audio, Backpropagation, Fixed Size Momentum, jAudio, Windows size*

1. PENDAHULUAN

Secara umum audio adalah representasi dari suara atau bunyi, sedangkan suara ialah gelombang longitudinal yang merambat melalui perantara. Sedangkan untuk digital audio secara mudah ialah cara alternatif dalam mempresentasikan gelombang audio dalam bentuk digital yaitu dalam binary. Mixed Type Audio adalah sebuah berkas audio yang memiliki lebih dari satu jenis sumber suara, seperti speech atau suara manusia yang digabung dengan musik, ataupun digabung dengan suara lingkungan sekitar. Jenis audio yang beragam pun memunculkan kebutuhan akan suatu pendekatan untuk dapat mengklasifikasikan jenis audio berdasarkan isi dari audio tersebut. Untuk melakukan pengklasifikasian maka dibutuhkan berkas audio yang diolah sehingga dapat diambil kesimpulan jenis apa audio tersebut.

Penelitian Lei Chen dan kawan-kawan [1] melakukan klasifikasi pada mixed type audio yaitu speech dengan musik dan musik dengan environment menggunakan SVM sebagai classifier dan ZCR sebagai feature extraction. Penelitian tersebut juga membandingkan dengan beberapa classifier yang lain, yaitu Backpropagation, Naive Bayes, dan k-NN dan memberikan kesimpulan bahwa SVM dan Backpropagation menghasilkan akurasi paling baik, yaitu SVM 90.84% dan BP 90.07% untuk speech dan musik, namun untuk speech dan environment akurasi yang didapat adalah SVM 64.76% dan Backpropagation 63.81%. Sementara pada penelitian mengklasifikasikan genre musik dengan menggunakan Backpropagation dengan Fixed Size Momentum dapat menambah kecepatan pelatihan dan meningkatkan akurasi [2]. Fixed Size Momentum pada dasarnya dirancang untuk mengatasi beberapa keterbatasan yang terkait dengan Backpropagation. Fixed Size Momentum menggunakan update dari bobot sebelumnya untuk menentukan seberapa besar update dapat dibuat.

Dalam penelitian ini metode yang akan digunakan untuk mengklasifikasi adalah metode klasifikasi Jaringan Syaraf Tiruan Backpropagation. Dengan ditambah metode Fixed Size Momentum untuk dapat menambah kecepatan pelatihan dan meningkatkan akurasi. Ada beberapa feature untuk mendapatkan ciri dari audio, antara lain ZCR (zero crossing rate), MFCC (mel frequency cepstrum coefficient), RMS (root mean square). Dan dalam penelitian ini untuk mengekstraksi sinyal suara menggunakan metode Mel Frequency Cepstrum Coeficcients (MFCC), karena memiliki beberapa kelebihan diantaranya mampu menangkap informasi penting dalam sinyal

suara, menghasilkan data seminimal mungkin tanpa menghilangkan informasi-informasi yang ada dan mereplikasikan organ pendengaran manusia dalam melakuakan persepsi terhadap sinyal suara [3]. dan pada penelitian [2] menggunakan ANN backpropagation dengan fixed size momentum untuk mengklasifikasikan genre musik juga menggunakan MFCC sebagai feature extraction.

Diharapkan dengan menambahkan Fixed Size Momentum pada Backpropagation dan menggunakan Mel Frequency Cepstrum Coefficient (MFCC) pada feature extraction bisa menambah akurasi. Dari latar belakang tersebut maka penulis tertarik untuk mengambil pokok bahasan skripsi dengan judul "Klasifikasi Mixed Type Audio menggunakan Backpropagation dengan Fixed Size Momentum".

2. ISI PENELITIAN

2.1 *Backpropagation*

Backpropagation adalah salah satu algoritma *supervised learning* yang digunakan dalam *artificial neural networks. Backpropagation* [4] mencari kombinasi bobot untuk meminimalkan kesalahan output untuk dianggap menjadi solusi yang benar. Secara sederhana *backpropagation* dilakukan dalam dua tahap, yaitu :

1) *Feed-forward*
   Pola yang akan dilatih diset ke setiap unit di *input layer*, lalu *output* yang dihasilkan ditransmisikan ke *layer* selanjutnya terus sampai *output layer.*
2) *Backpropagation*
   Berdasarkan *output* yang diharapkan, setiap bobot disesuaikan agar menghasilkan *error* yang minimal mulai dari bobot yang terhubung ke *ouput neuron*, lalu terus mundur sampai ke *input layer* .

Salah satu *activation function* yang populer untuk jaringan *backpropagation* adalah *binary sigmoid* dengan rentang nilai (0,1) yang didefinisikan sebagai (2.1). untuk tahap *backpropagation* menggunakan fungsi turunannya yaitu (2.2).

$$f(x) = \frac{1}{1+e^{-x}} \quad (2.1)$$

$$f'(x) = f(x)(1-f(x)) \quad (2.2)$$

Alternatif selain fungsi *binary sigmoid* adalah *bipolar sigmoid* dengan rentang nilai (-1,1) yang didefinisikan sebagai (2.3). untuk tahap *backpropagation* menggunakan fungsi turunannya yaitu (2.4).

$$f(x) = \frac{2}{1+e^{-x}} - 1 \quad (2.3)$$

$$f'(x) = \frac{1}{2}(1 + f(x))(1-f(x)) \quad (2.4)$$

Berikut tahapan pada proses pealtihan algoritma *backpropagation*.

Langkah 1 : Inisialisasi bobot dan parameter

Langkah 2 : Hitung nilai aktivasi pada *hidden layer*

$$Y_net_j = \sum_{j=1}^{p} Xi\, W1_{i,j} \quad (2.5)$$

Di mana :

Y_net_j : jumlah total *input* untuk *node* ke-j

P : jumlah *node* pada *hidden layer* sebelumnya, untuk *hidden layer* yang pertama n adalah jumlah *node input*

Xi : nilai node ke-I yang dikeluarkan oleh *hidden layer* sebelumnya, untuk *hidden layer* yang pertama Xi adalah input yang diterima oleh sistem

W1i,j : bobot yang menghubungkan antara *node* ke-i dan *node* ke-j Hitung *output* setiap *node* pada *hidden layer* dengan fungsi aktivasi

$Y_j = f(Y_net_j)$ (2.6)

Di mana :

Y_j : *output* dari *node* ke-j

$$Y_j = \frac{1}{1+e^{-Y_net_j}} \quad (2.7)$$

Langkah 3 : Hitung *input* setiap *node* pada *output layer*

$$Z_net_k = \sum_{k=1}^{m} Y_j\, W2_{j,k} \quad (2.8)$$

Di mana :

Z_net_k : jumlah total input untuk *node* ke-k

m : jumlah *node* pada *hidden layer* sebelum *output layer*

W2j,k: bobot yang menghubungkan antara *node* ke-j dan *node* ke-k

Hitung *output* pada setiap *node* pada *output layer*

$Zk = f(Z_net_k)$ (2.9)

$$Z_k = \frac{1}{1+e^{-Z_net_k}} \quad (2.10)$$

Langkah 4 : Hitung *error* setiap *node* pada *output layer*

$$\delta_k = (t_k - Z_k)\, f'(Z_net_k) \quad (2.11)$$

Ket :

k : banyaknya *node* pada *output layer*

t : *output* yang diharapkan

Langkah 5 : Hitung perhitungan perubahan bobot ΔW2 (2.18)

Langkah 6: Hitung *error* setiap *node* pada *hidden layer*

$$\delta_net_j = \sum_{k=1}^{m} \delta_k w2_{j,k} \quad (2.12)$$

Ket :

m : jumlah *node* pada *layer* sebelumnya hitung *error*

$$\delta_j = \delta_net_j f'(Y_net_j) \quad (2.13)$$

Ket :

$f'(Y_net_j) : Y_net\ j * (t - Y_net\ j)$ (2.14)

t : *output* yang diharapkan

Langkah 7 : Hitung perhitungan perubahan bobot ΔW1 (2.17)

Langkah 8 : Hitung *update* bobot pada *hidden layer* dan *output layer*

$$W2_{j,k} = W2_{j,k}(lama) + \Delta W2_{j,k} \quad (2.15)$$

$$W1_{i,j} = W1_{i,j}(lama) + \Delta W1_{i,j} \quad (2.16)$$

2.2 *Fixed Size Momentum*

Fixed size momentum pada dasarnya dirancang untuk mengatasi beberapa keterbatasan yang terkait

dengan pelatihan *backpropagation.* Dalam rangka untuk mempercepat pelatihan, banyak peneliti menambah bobot update berdasarkan update bobot sebelumnya. Hal ini secara efektif meningkatkan waktu pembelajaran [2]. Banyak algoritma menggunakan informasi dari update bobot sebelumnya untuk menentukan berapa besar sebuah update yang bisa dibuat tanpa divergen.

Fixed size momentum dapat menambah kecepatan melebihi momentum standar. Fixed size momentum di rancang untuk mengatasi panjang history dari update bobot yang sudah ada untuk setiap hubungan yang ada di neural networks. Dengan menggunakan informasi tambahan ini yang disimpan, *fixed size momentum* dapat meningkatkan kecepatan pembelajaran dan juga meningkatkan akurasi [2].
Aturan standar update bobot yang digunakan dalam *backpropagation* adalah :

$\Delta W_{i,j} = \alpha \, \delta_j \, X_{i,j}$

Dimana :

i = indeks dari sumber node

j = indeks target

α = learning rate

δ_j= backpropagated error term

x= nilai input bobot

cara aturan update bobot ini sangat lambat dan membutuhkan waktu yang lama.

Fixed size momentum menggunakan fixed size window untuk mendapatkan informasi lebih banyak daripada menggunakan standar momentum. Dengan menggunakan lebih banyak memori dapat memungkinkan untuk mengatasi beberapa keterbatasan masalah.

Fixed size momentum dapat mengingat update bobot yang sering muncul dan menggunakan informasi tersebut untuk bobot yang lainnya. Dengan standar momentum , kesalahan dari update sebelumnya sebagian akan diterapkan untuk selanjutnya, untuk kasus terburuk beberapa sampel akan sering memiliki update yang berlawanan. Situasu ini dapat mengganggu momentum yang telah dibangun dan bahkan dapat membuat lebih lama dalam melatih . fixed size momentum dapat melihat history yang lebih luas.
Untuk rumus fixed size momentum adalah sebagai berikut.

$\Delta W1_{i,j}(t) = \alpha \, \delta_j \, X_i + f(\alpha \, \delta_j X_i, \Delta W1_{ij}(t-1), \Delta W1_{i,j}(t-2), \dots, \Delta W1_{i,j}(t-k))$ (2.17)

$\Delta W2_{j,k}(t) = \alpha \, \delta_k \, Y_j + f(\alpha \, \delta_j Y_j, \Delta W2_{j.k}(t-1), \Delta W2_{j.k}(t-2), \dots, \Delta W2_{j,k}(t-k))$ (2.18)

Dan untuk fungsi f nya adalah sebagai berikut.

$$f(z, \delta_1, \dots, \delta_k) = \begin{cases} -z & if, \; z * \sum \delta_i < 0 \\ 0 & if, \; z * \sum \delta_i \geq 0 \end{cases}$$

(2.19)

Terdapat argumen k+1 untuk fungsi f , yang pertama adalah update saat ini dan sisanya adalah update dari k sebelumnya dimana k adalah window size untuk algoritma fixed size momentum. δ_1

Rumus diatas dapat membantu dalam konvergensi lebih cepat dan juga meningkatkan akurasi klasifikasi dan meningkatkan ukuran dari *history window* yang digunakan untuk melatih jaringan.

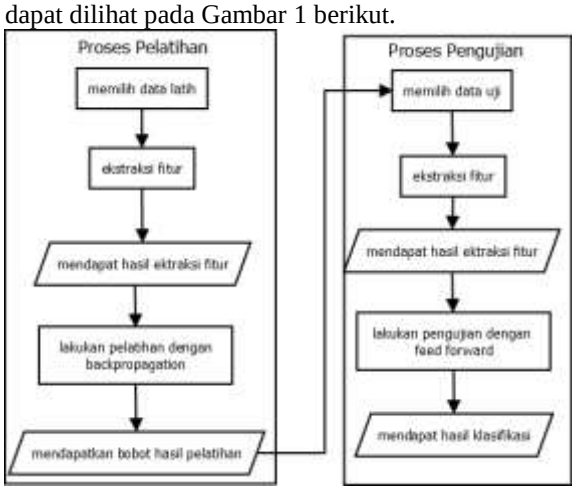
2.3 jAudio *Feature Extractor*

jAudio adalah digital signal processing yang dibangun untuk menyediakan program yang mudah digunakan untuk melakukan ekstraksi fitur pada audio[5].

Fitur audio yang di ekstrak seperti MFCC, LPC, Beat Histogram dan lain-lain. jAudio Feature Extractor bisa digunakan secara graphical atau command line. Sistem yang dibuat diintegrasikan dengan cara mengeksekusi jAudio dengan command line, yang selanjutnya hasilnya disimpan di file xml dan diambil oleh sistem sebagai input.

2.4 Pembahasan

Sistem meliputi dua tahap yaitu proses pelatihan dan proses pengujian. Proses pelatihan dilakukan sampai mendapatkan bobot yang diinginkan terpenuhi. Jika kondisi yang diinginkan sudah terpenuhi maka selanjutnya bisa dilanjutkan ke proses pengujian. Pada proses pengujian, akan dimasukan data uji untuk mengklasifikasikan hasilnya akan dibandingkan dari bobot pelatihan yang dilakukan sebelumnya. Gambaran umum sistem dapat dilihat pada Gambar 1 berikut.



Gambar 1 Gambaran Umum Sistem

Pada gambar diatas dapat dilihat tahapan pertama yaitu melakukan pelatihan dengan backpropagation setelah pelatihan selesai maka hasil dari bobot pelatihan digunakan untuk proses pengujian agar mendapatkan hasil klasifikasi.

Data masukkan yang digunakan adalah berformat wav, kemudian dilakukan ekstraksi menggunakan jAudio dan memilih metode MFCC sebagai *feature extraction*. Setelah diekstraksi maka hasilnya kemudian akan keluar dalam bentuk array dan disimpan dalam format xml yang kemudian akan dipanggil oleh sistem jika diperlukan. Data masukkan yang digunakan memiliki format sebagai berikut.

1. Data merupakan audio yang berformat (.wav)
2. Panjang audio antara 2-5 detik.
3. Ukuran data antara 200 – 600 kb.

Berikut adalah hasil ekstraksi untuk contoh satu data menggunakan jAudio dan MFCC sebagai *feature extraction* speechxenvironment1.1.wav.
[-131.0, 15.97, 7.758, 3.273, 0.1709, 1.04, 0.05388, -0.477, 0.091, 0.8632, 1.175, 0.2048, 0.01695].

Tabel 1 Index dari setiap kelas

Index	Kelas	Output
1	Speech dan environment	0
2	Speech dan music	1

Pembagian kelas diatas berdasarkan dari penelitian klasifikasi *mixed type audio* dengan *support vector machine*, pada penelitian ini akan menguji tingkat akurasi dari masing-masing kelas diatas.

Arsitektur metode *backpropagation* terdiri dari tiga layer dalam proses pembelajarannya, yaitu *input layer, hidden layer*, dan *output layer*.

1) *Input layer*

Input layer memiliki jumlah *neuron* yang sudah dipengaruhi oleh jumlah masukan pada suatu kasus, *input layer* (X_i) pada kasus ini memiliki 13 *neuron*, karena pada penelitian ini menggunakan metode MFCC sebagai fitur nya. Pada *input layer* terjadi pengiriman sinyal input X ke *hidden layer*.

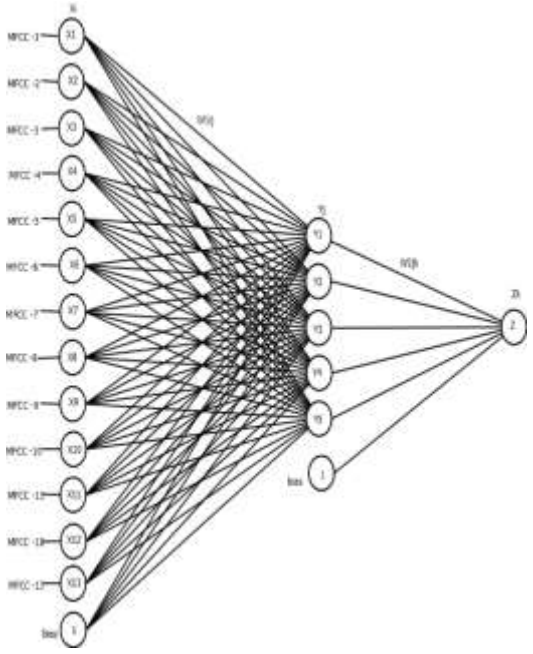
2) *Hidden layer*

Tidak ada ketentuan yang pasti dalam menentukan jumlah *neuron* pada *hidden layer*, namun jumlah *hidden layer* berpengaruh terhadap akurasi dan laju pembelajaran dari metode *Backpropagation. Hidden layer* merupakan variabel yang dapat diubah-ubah untuk mengetahui akurasi dari metode ini, pada analisis metode ini digunakan 5 *neuron* pada *hidden layer* (penggunaan 5 *hidden neuron* ini merupakan pemilihan awal, nilai lainnya juga akan diuji pada bab selanjutnya). *Hidden layer* pada *backpropagation berfungsi sebagai* untuk mengoreksi dan menyesuaikan bobot, sehingga didapatkan nilai bobot yang baru yang bisa diarahkan mendekati dengan target *output* yang diinginkan.

3) *Output layer*

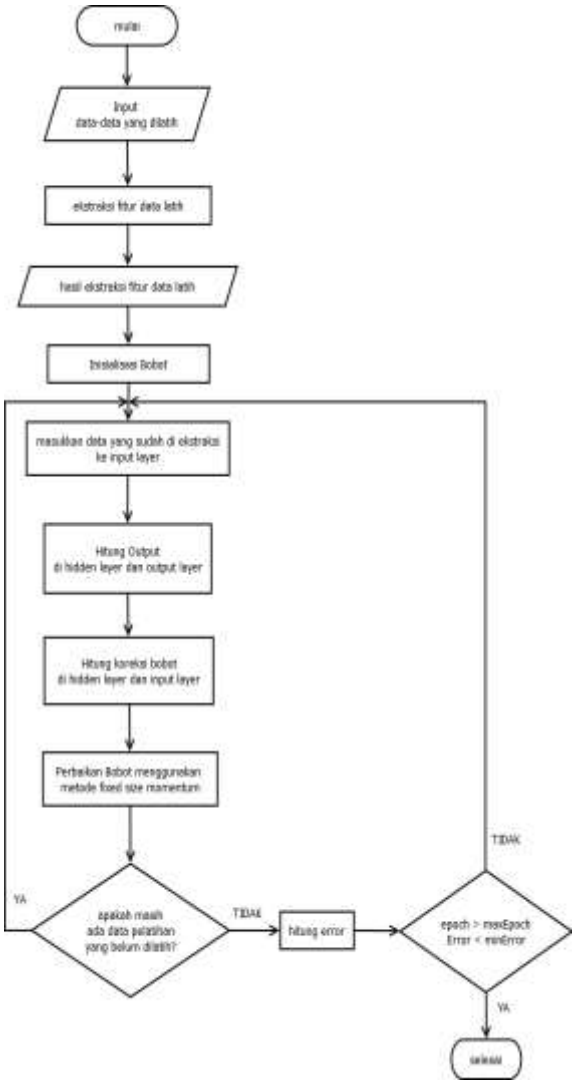
Output layer (Z_i) pada contoh kasus ini memiliki 1 *neuron*, karena jumlah keluaran pada kasus ini berupa kode dari masing-masing kelas. *Neuron* tersebut digunakan untuk mengidentifikasi dan juga *learning* sebagai *output* yang seharusnya.

Arsitektur jaringan pada contoh kasus ini dapat dilihat pada **Gambar 2**.



Gambar 2 Arsitektur Jaringan Metode Backpropagation *Mixed Type Audio Classification*

Keterangan :
Xi :lapisan input terdiri dari 13 neuron
W1i,j : bobot pada lapisan tersembunyi
Yj : Lapisan tersembunyi terdiri dari 5 neuron
W2j,k : bobot pada lapisan keluaran
i : variabel yang mewakili input layer
j : variabel yang mewakili hidden layer
k : variabel yang mewakili output layer
Zk : lapisan output terdiri dari 3 neuron
1 : konstanta bias



Gambar 3 *Flowchart* pelatihan BPFSM

Penjelasan dari flowchart tersebut akan diimplementasikan pada contoh kasus dibawah ini.

Penentuan Variabel Pelatihan

Tahap pertama pada metode *backpropagation* yang diimplementasikan pada klasifikasi *mixed type audio* adalah penentuan variabel pelatihan. Terdapat tiga variabel pelatihan yang ditentukan, yaitu learning rate, windowed size, dan minimal error.

Penentuan nilai variabel pelatihan dipengaruhi oleh *learning rate* (laju pembelajaran) dan window size untuk *fixed size momentum*. Nilai *learning rate* yang tidak sesuai berdampak pada hasil klasifikasi yang kurang optimal. Apabila nilai *learning rate* terlalu besar, jaringan akan mencapai *konvergen* dalam waktu yang singkat tetapi *error* klasifikasi menjadi besar. Sebaliknya bila *learning rate* terlalu rendah, hasil klasifikasi memiliki akurasi yang baik tetapi proses training membutuhkan waktu yang lama, batas *epoch* dan target *error* berfungsi untuk membatasi pelatihan, dimana kedua variabel ini akan dibandingkan dengan kondisi pelatihan. Berikut ini adalah variabel pelatihan yang akan digunakan pada analisis metode contoh kasus ini adalah .

Learning rate

Learning rate yang digunakan adalah 0.2, karena dibutuhkan perubahan yang besar pada setiap nilainya untuk mencapai nilai yang diharapkan. Seperti pada penelitian [2] yang sudah menguji beberapa nilai parameter learning rate. Nilai learning rate yang tidak sesuai berdampak pada hasil klasifikasi yang kurang optimal. Nilai lain pada parameter ini juga akan diuji pada bab selanjutnya.

Minimal error

Minimal error yang digunakan adalah 0.01, karena semakin kecil error maka semakin baik akurasi yang akan didapat [2], penggunaan minimal error 0.01 merupakan pemilihan awal, nilai lainnya juga akan diuji pada bab selanjutnya.

Batas *epoch*

Batas *epoch* yang digunakan adalah 1000000, pemilihan batas epoch disesuaikan dengan kebutuhan sistem.

Window size

Window size yang digunakan adalah 3, digunakan 3 karena dari penelitian sebelumnya menggunakan 3 windows size [2].

Pengambilan data pelatihan

Data pelatihan yang digunakan pada metode ini berupa pasangan sinyal input (X) dengan target output (t) yang membentuk sebuah pola. Sinyal input (X) memiliki nilai hasil dari parsing seperti yang sudah di contohkan pada analisis data masukan, nilai ini didapatkan dari hasil ekstraksi fitur menggunkan metode MFCC. Sinyal input akan digunakan pada proses perambatan maju dan perbaikan bobot. Target Output (t) akan digunakan untuk menghitung error dan juga digunakan untuk perambatan mundur.

Data masukan yang digunakan adalah hasil dari ekstraksi fitur menggunakan metode MFCC yang mempunyai 13 neuron, masih terdapat perbedaan angka yang masih sangat jauh antara satu fitur dengan fitur lainnya. Maka dari itu hasil ekstraksi selanjutnya dilakukan normalisasi pada tiap fitur, normalisasi digunakan karena dapat membuat hasil fitur menjadi tidak terlalu jauh bedanya.

Berikut adalah cara normalisasi :

Dari masing-masing fitur diambil hasil yang terbesar (max) dan terkecilnya (min), Gunakan rumus normalisasi berikut.

$X'=(X - 〚Xmink〛) / (〚Xmaxk〛 - 〚Xmink〛)$ [12]

X’ = hasil normalisasi

X = nilai yang akan dinormalisasi

Xmink = nilai terkecil dari masing-masing fitur ke-k

Xmaxk = nilai terbesar dari masing-masing fitur ke-k

K = Indeks fitur (1,2,…,n)

Lakukan normalisasi pada tiap input di masing-masing data.
Berikut adalah langkah-langkah untuk pelatihan *backpropagation* dengan *fixed size momentum.*

Langkah 1: data yang dipilih akan melalui proses ekstraksi fitur dengan menggunakan jAudio terlebih dahulu.

Langkah 2 : setelah itu dilakukan inisialisasi bobot, Inisialisasi bobot dan bias dilakukan untuk seluruh bobot dan bias yang menghubungkan antara *input layer* ke *hidden layer* dan *hidden layer* ke *output layer*. Bobot yang menghubungkan antara input layer ke hidden layer dilambangkan dengan W1i,j. Bobot yang menghubungkan antara *hidden layer* ke *output layer* dilambangkan dengan W2j,k. Inisialisasi ini dilakukan dengan memberi nilai acak terkecil antara -1 hingga 1 ke seluruh bobot dan bias.

Langkah 3 : setelah dilakukan inisialisasi bobot maka selanjutnya adalah menginputkan data yang sudah diekstraksi ke *input layer.*

Langkah 4 : setelah itu akan dilakukan proses *feed forward*, dimana pada proses ini akan mengaktivasi nilai pada *hidden layer* dan *output layer* dengan menggunakan persamaan sebagai berikut.
Aktivasi *hidden layer* menggunakan persamaan (2.5) dan (2.7).
Aktivasi *output layer* menggunakan persamaan (2.8) dan (2.10).

Langkah 5 : setelah proses *feed forward* selesai dan telah mendapatkan nilai aktivasi pada *hidden layer* dan *output layer*, selanjutnya akan dimulai proses *backpropagation* yaitu mencari nilai *error* yang digunakan untuk menghitung perubahan bobot dengan menggunakan persamaan (2.11) sampai dengan persamaan (2.18).

Langkah 6 : melakukan proses perubahan bobot dengan menggunakan *fixed size momentum* perhitungan ini terdapat pada persamaan (2.17) untuk perubahan bobot yang terjadi pada W1 dan persamaan (2.18) untuk W2.

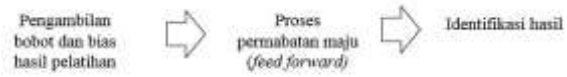
Langkah 7 : setelah didapatkan perubahan bobot yang baru maka selanjutnya akan menghitung bobot baru dimana pada proses ini akan menggunakan persamaan (2.15) dan (2.16).

Langkah 8 : kemudian akan dicek apakah masih ada data yang belum dilatih, jika masih ada maka proses akan diulangi kembali ke langkah 3.

Langkah 9: jika semua data sudah dilatih maka selanjutnya akan dicek syarat berhenti pelatihan, pelatihan akan dihentikan jika salah satu dari syarat dipenuhi yaitu yang pertama jika epoch > maxEpoch, kedua error < minError.

Langkah 10 : jika salah satu syarat terpenuhi maka pelatihan selesai, namun jika belum terpenuhi maka akan kemablai pada langkah 2.

Selanjutnya jika pelatihan telah selesai dilakukan dan didapatkan hasil bobot akhir pelatihan, maka selanjutnya akan dilanjutkan dengan pengujian. Bobot yang digunakan pada pelatihan yaitu bobot akhir dari pelatihan. Berikut adalah alur pengujian untuk mengklasifikasikan *mixed type audio.*



Gambar 4 alur proses pengujian

Langkah 1 : mengambil bobot awal dari bobot hasil akhir pelatihan yang sebelumnya dilakukan.

Langkah 2 : masukan data yang akan diuji, kemudian data akan dilakukan ekstraksi fitur dengan menggunakan jAudio sama seperti pelatihan.

Langkah 3 : setelah didapat hasil ekstraksi maka akan dilakukan proses *feed forward* dimana hasil dari aktivasi *output layer* digunakan sebagai hasil untuk mendapatkan hasil klasifikasi. Proses aktivasi menggunakan rumus persamaan pada langkah 4 proses pelatihan yaitu (2.5) untuk menghitung aktivasi *hidden layer* dan persamaan (2.8) untuk mendapatkan hasil aktivasi dari *output layer.*

2.5 Hasil Pengujian

Pengujian ini dilakukan untuk mengklasifikasi 2 kelas yaitu kelas musik environment dan speech environment, dengan menggunakan total 60 data audio dengan 40 data sebagai data latih dan 20 data sebagai data uji.

2.5.1 Pengujian *Hidden Layer*

Jumlah neuron di *hidden layer* juga berpengaruh pada keberhasilan pembelajaran. Pengujian parameter ini menggunakan 0.2 pada *learning rate*. Adapun hasil pengujian untuk jumlah *neuron* di *hidden layer* bisa dilihat pada Tabel 2.

Tabel 2 Hasil pengujian *Hidden Layer*

Data Latih	40 data		
Data Uji	20 data (10 data per kelas)		
Pengujian ke -	***Hidden layer***	**Akurasi**	**Jumlah iterasi**
1.a	3	75%	1000000
1.b	4	90%	1000000
1.c	5	95%	1000000

2.5.2 Pengujian *Learning Rate*

Learning rate yang digunakan juga berpengaruh pada keberhasilan pelatihan. Pengujian parameter ini menggunakan 5 *hidden layer* , dan 0,01 pada *minimal error*. Adapun hasil pengujian *learning rate* bisa dilihat pada Tabel 3.

Tabel 3 Hasil Pengujian *Learning Rate*

Data Latih	40 data		
Data Uji	20 data (10 data per kelas)		
Pengujian ke -	***Learning rate***	**Akurasi**	**Jumlah iterasi**
2.a	0.1	70%	168440
2.b	0.2	80%	1000000
2.c	0.3	90%	102800
2.d	0.4	80%	56640
2.e	0.5	85%	58320

2.5.3 Pengujian *Windows Size*

Windows size yang digunakan juga berpengaruh pada hasil pelatihan, pengujian parameter ini menggunakan 0.3 *learning rate,* dan 5 *hidden layer*. Adapun hasil dari pengujian *windows size* dapat dilihat pada Tabel 4.

Tabel 4 Hasil Pengujian *Windows Size*

Data Latih	40 data		
Data Uji	20 data (10 data per kelas)		
Pengujian ke -	***Windows size***	**Akurasi**	**Jumlah iterasi**
3.a	2	90%	1000000
3.b	3	95%	283120
3.c	4	75%	1000000
3.d	8	70%	1000000
3.e	10	50%	1000000

2.5.4 Pengujian perbandingan BP dan BPFSM

Berdasarkan hasil dari pengujian pengujian sebelumnya didapatkan nilai-nilai parameter yang sudah dianggap cukup bagus untuk digunakan sebagai parameter untuk kedua metode yaitu menggunakan 5 *hidden layer,* 0.3 *learning rate* dan 3 *windows size*. Adapun hasil dari pengujian *backpropagation* dengan *backpropagation* dengan FSM dapat dilihat pada Tabel 5.

Tabel 5 Hasil Pengujian Perbandingan BP dan BPFSM

Data Latih	40 data				
Data Uji	20 data (10 data per kelas)				
Pengujian	***Hidden layer***	***Learning rate***	***Windows size***	**akurasi**	**Jumlah iterasi**
BP	5	0.3	0	80%	22440
BPFSM	5	0.3	3	95%	146520

Berdasarkan pengujian parameter algoritma yang telah dilakukan, dapat disimpulkan beberapa hal yaitu.

1) Berdasarkan pengujian mencari jumlah *neuron* pada *hidden layer* yang menghasilkan akurasi paling tinggi adalah 5 dengan akurasi sebesar 80%.
2) Berdasarkan pengujian *Learning Rate* yang menghasilkan akurasi paling tinggi adalah 0.3 dengan akurasi sebesar 90%.
3) Berdasarkan pengujian *windows size* yang menghasilkan akurasi paling tinggi adalah 3 dengan akurasi 95%.
4) Berdasarkan pengujian *backpropagation* dan *backpropagation* dengan *fixed size momentum* yang mendapatkan akurasi lebih tinggi adalah *backpropagation* dengan *fixed size momentum* menghasilkan akurasi sebesar 95%.

3. PENUTUP

Berdasarkan hasil yang didapat dalam penelitian ini, maka dapat diperoleh kesimpulan bahwa klasifikasi *mixed type audio* menggunakan *backpropagation* dengan *fixed size momentum* dapat meningkatkan akurasi sebesar 95% jika dibandingkan dengan *backpropagation* saja yang mendapatkan akurasi sebesar 80%, namun memerlukan iterasi lebih banyak. Semakin banyak *windows size* yang digunakan pada *fixed size momentum* menjadikan akurasi semakin berkurang, dan menjadikan iterasi semakin banyak.

Adapun saran yang diajukan untuk dapat mengembangkan penelitian serupa antara lain:

1. Dapat menerapkan klasifikasi *mixed type audio* menggunakan *Backpropagation* dengan *fixed size momentum* untuk mengklasifikasikan kelas *mixed type* yang ada pada potongan-potongan klip film.

2. Dapat menguji kembali tentang *windwos size* yang ada pada *fixed size momentum* agar dapat menemukan solusi permasalahan yang terjadi pada iterasi.

DAFTAR PUSTAKA

[1] L. Chen, S. Gunduz, M. T. Ozsu, "Mixed Type Audio Classification With Support Vector Machine," IEEE International Confrence on Multimedia and Expo, 2006.

[2] N. Prabhu, A. Anodkar, and R. Kenkre, "Music Genre Classification Using Improved Artificial Neural Network With Fixed Sized Momentum," International Journal of Comupter Application, vol. 101, no. 14, 2014.

[3] Y. Somantri, "Pengenalan Pembicara Dengan Ekstrasi Ciri MFCC," 2006.

[4] R. Rojas, "Neural Networks A Systematic Introduction," Springer-Verlag, Berlin, 1996.

[5] D. McEnnis, C. McKay, "jAudio Official Website," [Online]. Available:

http://jaudio.sourceforge.net. [Accessed 4 October 2016].

[6] P. Davis and S. Mermelstein, "Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences," IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 28, no. 4, pp. 357-366, 1980.