KINERJA PNN-TEROPTIMASI BERBASIS ALGORITMA GENETIKA DALAM PENGENALAN AROMA 2 CAMPURAN

Herry

Fakultas Ilmu Komputer, Universitas Indonesia
Kampus UI Depok, Jawa Barat 16424, Indonesia
email : herry198@puspa.cs.ui.ac.id

ABSTRAK

Hal yang harus diperhatikan dalam penggunaan *Probabilistic Neural Network* (PNN) adalah penentuan struktur jaringan, yaitu penentuan ukuran jaringan dan nilai parameter *smoothing*. Ukuran jaringan PNN akan semakin besar seiring dengan bertambahnya jumlah data pelatihan yang mengakibatkan biaya komputasi juga semakin tinggi. Sementara nilai parameter *smoothing* akan mempengaruhi tingkat klasifikasi dimana nilai yang optimal tergantung pada karakteristik data. Algoritma PNN-Teroptimasi (PNN-T) adalah algoritma yang dikembangkan untuk menentukan struktur PNN yang optimal. Dalam PNN-T, nilai parameter *smoothing* yang optimal dipilih dengan menggunakan Algoritma Genetika, sedangkan ukuran jaringan ditekan dengan memilih neuron yang representatif menggunakan Algoritma Orthogonal. Dilakukan perbandingan antara PNN dengan PNN-T dalam masalah pengenalan aroma 2 campuran. Dan hasilnya PNN-T mempunyai kinerja yang lebih baik yaitu tingkat pengenalan lebih tinggi dan penggunaan neuron dengan jumlah lebih rendah dibandingkan PNN.

Kata kunci : Jaringan syaraf tiruan, *Probabilistic Neural Network*, Algoritma Genetika, Algoritma Orthogonal, Sistem Penciuman Elektronik

Makalah diterima [3 Maret 2003]. Revisi akhir [22 Juli 2003].

1. PENDAHULUAN

Penggunaan jaringan neural buatan (JNB) untuk memecahkan permasalahan pengklasifikasian pola saat ini telah berkembang cukup pesat. *Probabilistic Neural Network* (PNN) adalah salah satu arsitektur JNB untuk pengklasifikasian pola dimana telah digunakan dalam pengenalan aroma pada Sistem Penciuman Elektronik [6] [7].

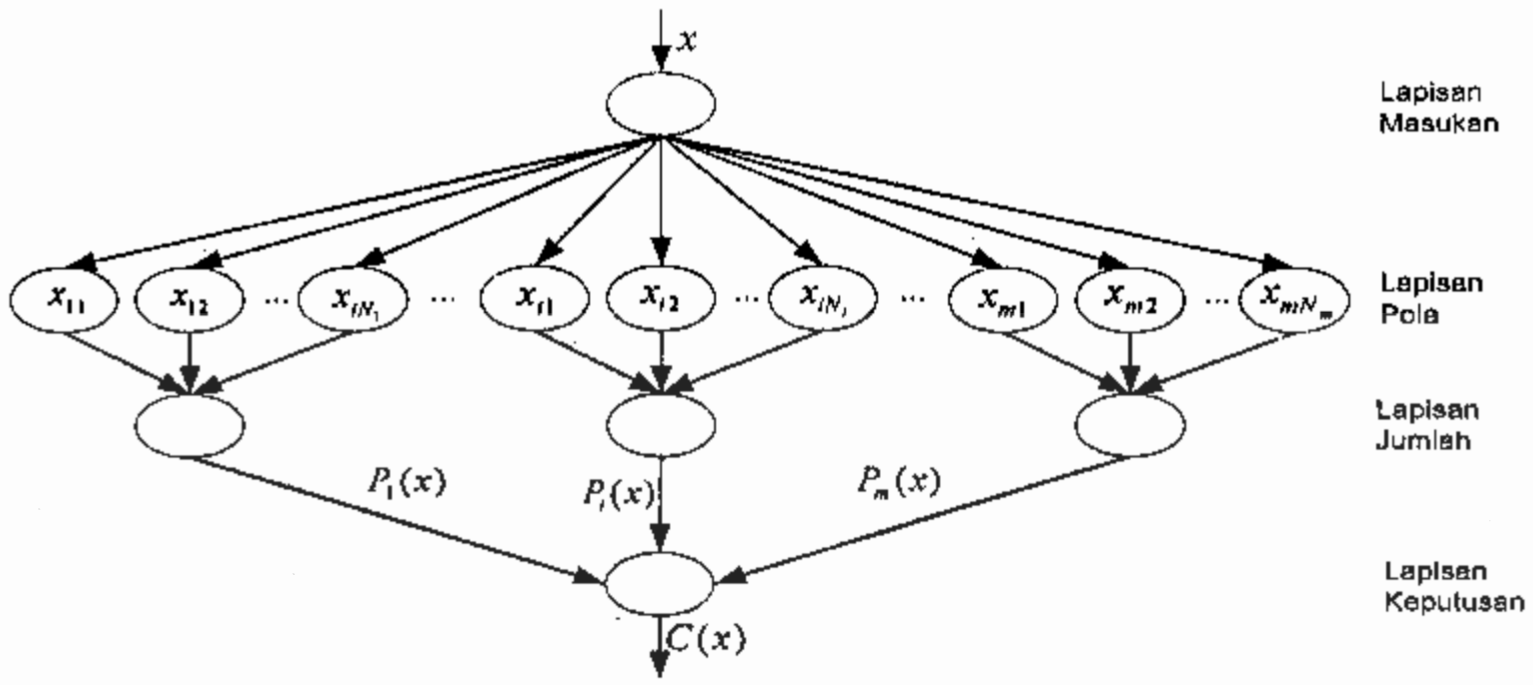
Dalam penggunaan PNN, hal yang harus diperhatikan adalah mengenai penentuan struktur jaringan yaitu penentuan ukuran jaringan dan nilai parameter *smoothing*. Dalam arsitektur PNN, setiap data pelatihan akan direpresentasikan oleh sebuah neuron pada lapisan pola [2]. Sehingga semakin banyak jumlah data pelatihan maka ukuran jaringan akan semakin besar [5] yang mengakibatkan biaya komputasi yang akan semakin tinggi [1]. Persoalan lainnya adalah penentuan nilai parameter *smoothing* (σ) dimana nilai tersebut memegang peranan penting dalam proses klasifikasi pada PNN. Dan nilai σ yang sesuai, sering tergantung pada data yang digunakan [1]. Permasalahan tersebut telah coba dipecahkan oleh para ilmuwan. Dan salah satu algoritma yang telah dikembangkan untuk memecahkannya adalah PNN-Teroptimasi (PNN-T) [1].

Dalam artikel ini akan dijelaskan penelitian perbandingan kinerja antara PNN dengan PNN-T dalam pengenalan aroma 2 campuran pada Sistem Penciuman Elektronik. Sistematika penulisan adalah sebagai berikut: pertama, adalah pendahuluan; kedua, adalah penjelasan secara singkat mengenai PNN; ketiga, adalah penjelasan algoritma optimasi PNN-T; keempat, adalah hasil eksperimen yang telah dilakukan; dan kelima, adalah kesimpulan.

2. PROBABILISTIC NEURAL NETWORK

PNN dikembangkan oleh Donald Specht pada tahun 1990 [1] [2]. Arsitektur jaringan PNN terdiri dari 4 lapisan yang dapat digambar seperti pada gambar 1. Lapisan masukan tidak melakukan operasi apapun dan hanya berfungsi untuk mendistribusikan pola yang ingin diklasifikasi ke neuron-neuron pada lapisan pola. Lapisan pola terdiri dari neuron-neuron yang masing-masing merepresentasikan satu vektor data pelatihan [2]. Semakin banyak data pelatihan maka semakin besar ukuran jaringan. Fungsi aktivasi neuron x_{ij} (neuron ke-j dari kelas ke-i) yang digunakan adalah sebagai berikut :

$$\phi_{ij}(x) = \frac{1}{(2\pi)^{d/2}\sigma^{d}} \exp\left[-\frac{(x - x_{ij})^{T}(x - x_{ij})}{2\sigma^{2}}\right] \quad (1)$$



Gambar 1. Arsitektur PNN

Dimana i adalah nomor kelas, j adalah nomor neuron, d adalah dimensi vektor pola, σ adalah parameter *smoothing*, x adalah vektor masukan, dan x_{ij} adalah vektor pelatihan untuk neuron ke-j dari kelas ke-i. Pada lapisan jumlah akan dihitung probabilitas maksimum pola masukan termasuk ke dalam suatu kelas, dengan menghitung rata-rata keluaran neuron pada lapisan pola yang termasuk ke dalam kelas yang sama. Fungsi aktivasi yang digunakan neuron ke-i adalah :

$$P_i(x) = \frac{1}{N_i} \sum_{j=1}^{N_i} \phi_{ij}(x) \qquad (2)$$

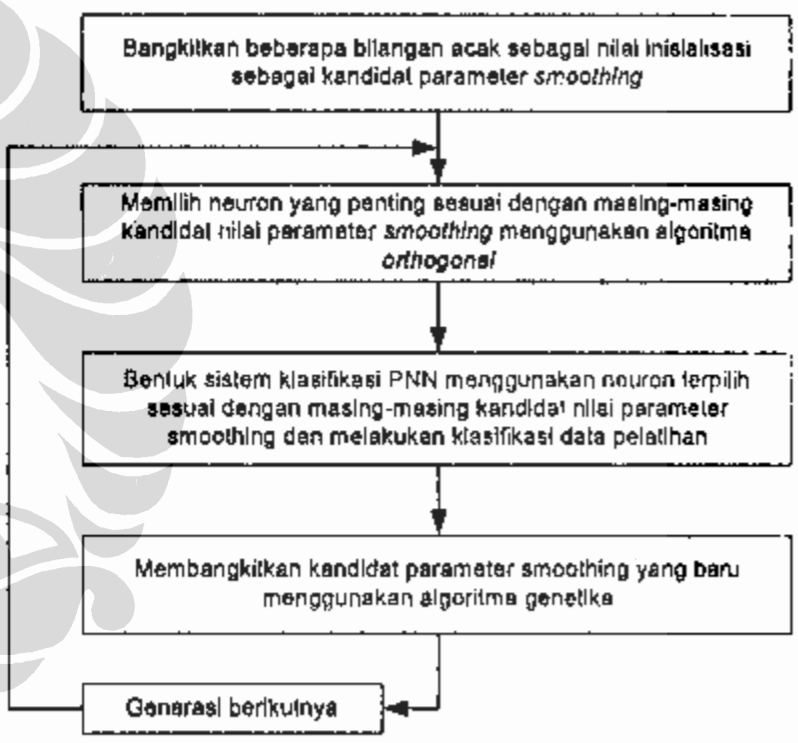
Dimana P_i adalah probabilitas maksimum vektor masukan x termasuk ke dalam kelas ke-i, dan N_i adalah jumlah neuron pada lapisan pola yang termasuk ke dalam kelas ke-i. Lapisan keputusan akan menentukan termasuk kelas manakah pola masukan berada. Keputusan diambil dengan memilih kelas yang mempunyai probabilitas tertinggi dibandingkan terhadap kelas yang lainnya, dengan menggunakan fungsi aktivasi sebagai berikut :

$$\hat{C}(x) = \text{argmax}\{ P_i(x) \} \quad i = 1, 2, 3, \ldots, m \qquad (3)$$

Dimana $\hat{C}(x)$ adalah kelas yang dipilih untuk pola masukan x dan m adalah jumlah kelas yang terdapat pada jaringan.

3. PNN-TEROPTIMASI

Algoritma PNN-Teroptimasi terdiri dari 2 bagian yang berjalan secara iteratif seperti ditunjukkan pada gambar 2. Algoritma Genetika (GA). Sedangkan bagian kedua adalah melakukan proses seleksi neuron menggunakan



Gambar 2. Diagram Algoritma PNN-Teroptimasi

Algoritma Orthogonal untuk mencari ukuran jaringan yang optimal dengan menggunakan nilai parameter *smoothing* yang telah dipilih pada bagian pertama.

3.1. SELEKSI NEURON MENGGUNAKAN ALGORITMA ORTHOGONAL

Pada bagian ini diasumsikan nilai σ telah ditentukan sebelumnya. Tujuan yang ingin dicapai adalah menyeleksi neuron-neuron yang representatif yang terdapat dalam lapisan pola. Jika vektor pelatihan ke-*k* dari kelas C_i dinotasikan dengan vektor x_{ik}, maka probabilitas terbesar vektor x_{ik} diklasifikasikan ke dalam kelas C_i adalah :

$$p_i(x_{ik}) = \frac{1}{(2\pi)^{d/2}\sigma^d}\frac{1}{N_i}\sum_{j=1}^{N_i}\exp\left[-\frac{(x_{ik}-x_{ij})^T(x_{ik}-x_{ij})}{2\sigma^2}\right]$$

$$= \sum_{j=1}^{N_i}\phi_{ij}(x_{ik}) \qquad (4)$$

dimana :

$$\phi_{ij}(x_{ik}) = \frac{1}{(2\pi)^{d/2}\sigma^d}\frac{1}{N_i}\exp\left[-\frac{(x_{ik}-x_{ij})^T(x_{ik}-x_{ij})}{2\sigma^2}\right]$$

$p_i(x_{ik})$ adalah fungsi nonlinear dari σ dan vektor lapisan pola x_{ik}. Tetapi jika σ telah ditentukan nilainya dan keluaran dari setiap neuron $\phi_{ij}(x_{ij})$ dianggap sebagai sebuah variabel, $p_i(x_{ik})$ akan menjadi sebuah kombinasi linier dari variabel-variabel tersebut seperti pada persamaan (4), yang akan digunakan untuk melakukan evaluasi terhadap derajat kepentingan dari setiap neuron.

Persamaan (4) dapat juga ditulis dalam bentuk matriks sebagai :

$$P = \Phi\,\theta \qquad (5)$$

dimana

$$\theta = [1,1,...,1]^T$$

$$P = [p_i(x_{i1}), p_i(x_{i2}),..., p_i(x_{iN_i})]^T$$

$$\Phi = \begin{bmatrix} \phi_{i1}(x_{i1}) & \phi_{i2}(x_{i1}) & \cdots & \phi_{iN_i}(x_{i1}) \\ \phi_{i1}(x_{i2}) & \phi_{i2}(x_{i2}) & \cdots & \phi_{iN_i}(x_{i2}) \\ \cdots & \cdots & \cdots & \cdots \\ \phi_{i1}(x_{iN_i}) & \phi_{i2}(x_{iN_i}) & \cdots & \phi_{iN_i}(x_{iN_i}) \end{bmatrix}$$

Dengan melakukan transformasi orthogonal terhadap matrix Φ, akan didapatkan :

$$\Phi = QR = [Q_1, Q_2,..., Q_{N_i}]R \qquad (6)$$

dimana $Q_1, Q_2,..., Q_{Ni}$ adalah sebuah basis orthogonal dan *R* adalah sebuah matriks *upper triangular*.

Derajat kepentingan dari kandidat neuron ke-j yang termasuk kelas C_i dihitung berdasarkan nilai *norm* vektor Q_j [1].

$$\Gamma_j = Q_j^T Q_j \qquad (7)$$

Dalam kondisi dimana seluruh neuron mempunyai nilai parameter *smoothing* yang sama, lebih besar nilai Γ_j maka neuron tersebut akan semakin penting karena mengindikasikan bahwa lebih banyak neuron yang dekat dengan neuron tersebut.

Prosedur untuk menghitung derajat kepentingan dan urutan dari neuron adalah sebagai berikut [1] :

1) Pilih neuron dari N_i neuron yang paling representatif untuk kelas C_i dengan mencari neuron yang mempunyai derajat kepentingan tertinggi. Dan gunakan neuron tersebut untuk menghasilkan Q_1.

$$Q_1^{(\alpha)} = \phi_\alpha, \qquad \alpha = 1, 2, ..., N_i$$

$$\phi_\alpha = [\phi_\alpha(1), \phi_\alpha(2), ..., \phi_\alpha(N_i)]^T$$

Derajat kepentingan dievaluasi sebagai berikut :

$$\Gamma_1^{(\alpha)} = [Q_1^{(\alpha)}]^T Q_1^{(\alpha)}, \qquad \alpha = 1, 2, ..., N_i$$

2) Pilih neuron ke-j yang paling representatif dari neuron yang tersisa berjumlah $N_i - j + 1$. Neuron dengan derajat kepentingan tertinggi dianggap sebagai neuron yang representatif ke-j.

$$Q_j^{(\alpha)} = \phi_{k_\alpha} - \sum_{l=1}^{j-1} r_{l\alpha}^{(\alpha)} Q_l, \quad \alpha = 1, 2, ..., N_i - j + 1$$

$$r_{l\alpha}^{(\alpha)} = Q_l^T \phi_{k_\alpha} / Q_l^T Q_l, \alpha = 1, 2, ..., N_i - j + 1, \; l < i$$

3.2. PEMILIHAN NILAI PARAMETER SMOOTHING (σ) MENGGUNAKAN ALGORITMA GENETIKA

Dalam menyelesaikan permasalahan penentuan struktur PNN dengan batasan persoalan optimalisasi, yaitu min{n}, mengacu pada [1] :

$$\mu < \delta \qquad (8)$$

Dimana n adalah jumlah neuron yang terpilih dalam lapisan pola (ukuran jaringan). μ adalah tingkat kesalahan klasifikasi. Dan δ adalah batas atas dari toleransi tingkat kesalahan yang ditentukan. Karena tidak ada hubungan kuantitatif antara ukuran jaringan, tingkat kesalahan klasifikasi, dan σ, maka digunakan Algoritma Genetika (GA) untuk memecahkan persoalan optimalisasi diatas.

GA adalah algoritma pencarian yang dikembangkan berdasarkan mekanisme seleksi alami dan genetika alami oleh John Holland [3]. Penerapan GA pada bagian ini dilakukan dengan menerapkan langkah-langkah proses dalam GA yaitu pengkodean, penghitungan nilai *fitness*, reproduksi, *crossover*, dan mutasi.

1) Pengkodean

Data yang digunakan adalah data yang telah dinormalisasi, menyebabkan nilai σ akan lebih kecil daripada 1 [1]. Sehingga hanya bit-bit desimal dibelakang

komalah yang akan dikodekan. Pengkodean 4 bit desimal digunakan dalam penelitian ini untuk mengkodekan σ. Sebagai contoh, jika sebuah individu σ = 0,5247, nilai ini bisa direpresentasikan dengan *string* desimal sebagai berikut :

$$\underbrace{5}_{b_1}\ \underbrace{2}_{b_2}\ \underbrace{4}_{b_3}\ \underbrace{7}_{b_4}$$

2) Penghitungan nilai *fitness*

Setiap individu merepresentasikan sebuah nilai parameter *smoothing*. Sejumlah kandidat struktur jaringan diperoleh melalui prosedur pada bagian 3.1 dan menggunakan nilai σ yang direpresentasikan oleh seluruh individu. Karena objektif yang ingin dicapai yaitu meminimalkan ukuran jaringan, karena itu fungsi yang digunakan untuk menghitung nilai *fitness* harus berbanding terbalik dengan jumlah neuron yang terpilih. Nilai *fitness* bisa dihitung dengan menggunakan skema berikut ini :

$$\rho_i = \rho_{max} - \frac{\rho_{max} - \rho_{min}}{\eta_{max} - \eta_{min}}(\eta_i - \eta_{min}) \qquad (9)$$

dimana :

ρ_i = nilai *fitness* untuk individu ke-*i*

η_{min} = ukuran kandidat jaringan minimum dari populasi sekarang

η_{max} = ukuran kandidat jaringan maksimum dari populasi sekarang

ρ_{min} = nilai *fitness* minimum

ρ_{max} = nilai *fitness* maksimum

Dalam penelitian ini nilai ρ_{min} dan ρ_{max} yang digunakan adalah 0,5 dan 1.

3) Reproduksi

Pendekatan dengan menggunakan *roulette wheel* digunakan dalam prosedur reproduksi. Setiap *string* disediakan sebuah slot dalam *roulette wheel* dimana sudutnya sebanding dengan nilai *fitness* yang dimilikinya. Dalam prosedur ini, sebuah bilangan acak antara 0 sampai 2π dibangkitkan. *String* diduplikasikan ke *mating pool* jika bilangan acak tersebut jatuh ke slot dimana *string* berada. Prosedur reproduksi dilakukan berulang kali sampai jumlah *string* yang ada di *mating pool* sesuai dengan jumlah yang diinginkan.

4) *Crossover*

Tujuan dari operasi *crossover* adalah untuk menghasilkan solusi baru dengan menukar bit-bit antara individu-individu. Untuk melakukan operasi ini, pilih secara random posisi bit yang akan ditukarkan, lalu tukarkan bit pada kedua *string* pada posisi tersebut yang akan menghasilkan 2 buah *offspring*.

5) Mutasi

Operasi mutasi dilakukan untuk menghasilkan individu yang tidak mudah didapatkan melalui operasi *crossover*. Hal ini dilakuakn denga mengubah nilai bit yang terpilih dengan bilangan acak antara 0 dan 9.

Dengan menggabungkan kedua bagian tersebut, maka secara garis besar algoritma dapat dirumuskan sebagai berikut :

1. Bangkitkan populasi awal P terdiri n individu dimana setiap individu merepresentasikan sebuah nilai σ. Set nomor generasi i = 1.
2. Dengan menggunakan prosedur pemilihan neuron dengan algoritma orthogonal dan nilai σ yang didefinisikan oleh setiap individu, dapat dibentuk beberapa kandidat jaringan. Hitung nilai *fitness* dari setiap kandidat. Bentuk *mating pool* M menggunakan seluruh populasi di P, dimana setiap nilai *fitness* dari setiap kandidat jaringan ditetapkan sebagai nilai probabilitas dari setiap individu.
3. Secara acak, pilih 2 individu sebagai *parent* dari M. Lakukan prosedur *crossover* yang menghasilkan 2 *offspring* dan letakkan di O. Prosedur diulang sampai jumlah *offspring* di O sama dengan di P.
4. Mutasikan setiap bit dari setiap *offspring* di O dengan tingkat mutasi tertentu, hitung nilai *fitness* setiap mutan sesuai dengan langkah ke-2.
5. Pilih n individu dari P dan O yang mempunyai nilai *fitness* terbaik.
6. Kosongkan P, dan set P dengan individu yang telah dipilih pada langkat ke-5, set nomor generasi menjadi i = i + 1, dan kosongkan O.
7. Langkah 2-6 diulang sampai mencapai nomor generasi yang telah ditentukan.

4. EKSPERIMEN

Dalam eksperimen ini, percobaan dilakukan untuk mengetahui perbandingan antara kinerja PNN dengan PNN-T. Data yang digunakan adalah data aroma 2 campuran dari Sistem Penciuman Elektronik [6][7]. Dalam melakukan perbandingan kinerja, digunakan 2 tolak ukur yaitu tingkat pengenalan dan biaya komputasi. Karena biaya komputasi sebanding dengan jumlah neuron pada lapisan pola [1], maka digunakan jumlah neuron pada lapisan pola sebagai acuan. Semakin banyak jumlah neuron maka biaya komputasi akan semakin tinggi dan sebaliknya.

Aroma campuran yang diujikan dalam percobaan ini ada 3 jenis dan setiap jenis terdiri dari 6 macam aroma, sehingga total aroma yang diujikan adalah 18 macam.

Tabel 1. Daftar aroma campuran yang akan diklasifikasikan

Jenis Aroma Campuran	Komposisi	Keterangan
Jenis 1	J0%, J15%, J25%, J35%, J45%, J70%	Komposis masing-masing zat 1:1 (10ml : 10ml)
Jenis 2	K0%, K15%, K25%, K35%, K45%, K70%	
Jenis 3	M0%, M15%, M25%, M35%, M45%, M70%	

Daftar seluruh aroma campuran bisa dilihat pada tabel 1. Inisial J25% berarti aroma jeruk + konsentrasi alkohol 25% dengan komposisi 1:1 (50%:50%). Pengujian dilakukan dalam beberapa tahap yaitu tahap pertama 6 aroma, tahap kedua 8 aroma, tahap ketiga 9 aroma, tahap keempat 12 aroma, dan tahap kelima 18 aroma. Tujuan dari tahapan ini adalah untuk membandingkan kinerja PNN dengan PNN-T dalam mengklasifikasikan aroma campuran dari mulai yang terendah (6 pola) sampai yang tertinggi (18 pola) dan jumlah neuron yang digunakan. Pada tahap pertama sampai keempat, dilakukan lebih dari satu percobaan yang menggunakan kombinasi aroma campuran yang berbeda.

Dalam eksperimen, untuk algoritma PNN digunakan 2 nilai parameter *smoothing*. Pertama adalah nilai yang dipilih secara sembarang yaitu 0,1, dan kedua adalah nilai optimal yang dipilih melalui pencarian manual dengan kisaran 0,003 dan 5 yaitu 0,003. Dimana nilai σ yang optimal adalah nilai yang menghasilkan tingkat pengenalan tertinggi dari data pelatihan. Sedangkan untuk PNN-T, parameter-parameter yang digunakan adalah toleransi tingkat kesalahan 0%, jumlah populasi tiap generasi 30, ukuran *mating pool* 30, jumlah generasi 30, dan probabilitas mutasi 0,1. Hasil eksperimen PNN dengan parameter *smoothing* 0,1 (PNN-0,1), PNN dengan parameter *smoothing* 0,003 (PNN-0,003), dan PNN-T terhadap data pelatihan dan pengujian dijelaskan pada pada tabel-tabel berikut.

Pada tabel 2 dapat dilihat bahwa rata-rata tingkat pengenalan oleh PNN-0,1 adalah 81,98%. Sedangkan PNN-0,003 dan PNN-T mempunyai tingkat pengenalan yang lebih baik yaitu 99,93% dan 100%. Untuk PNN-T, algoritma ini mampu memenuhi kriteria toleransi kesalahan 0% yang diberikan dalam parameter. Sedangkan tabel 4 memperlihatkan bahwa PNN-0,1 mempunyai rata-rata tingkat pengenalan 79,39%. Hasil tersebut lebih rendah daripada PNN-0,003 dan PNN-T yang mempunyai rata-rata tingkat pengenalan 98,27% dan 98,34%. Perlu diperhatikan bahwa nilai σ dalam PNN ditentukan secara manual. Sedangkan pada PNN-T, nilai σ ditentukan secara otomatis oleh algoritma.

Hasil lain yang diperlihat oleh tabel 2 dan 3 adalah tidak adanya pengaruh jumlah kelas yang harus diklasifikasikan terhadap tingkat klasifikasi. Hal ini berbeda dengan hasil yang diperoleh menggunakan algoritma *Backpropagation* dalam [6], dimana semakin banyak jumlah kelas maka tingkat pengenalan akan semakin menurun.

Tabel 2. Rata-rata tingkat pengenalan data pelatihan

Jumlah Kelas	Rata-rata tingkat pengenalan data pelatihan		
	PNN-0,1	PNN-0,003	PNN-T
6	79.43%	99.92%	100.00%
8	81.74%	99.88%	100.00%
9	90.93%	100.00%	100.00%
12	79.56%	99.91%	100.00%
18	78.25%	99.92%	100.00%
	81.98%	**99.93%**	**100.00%**

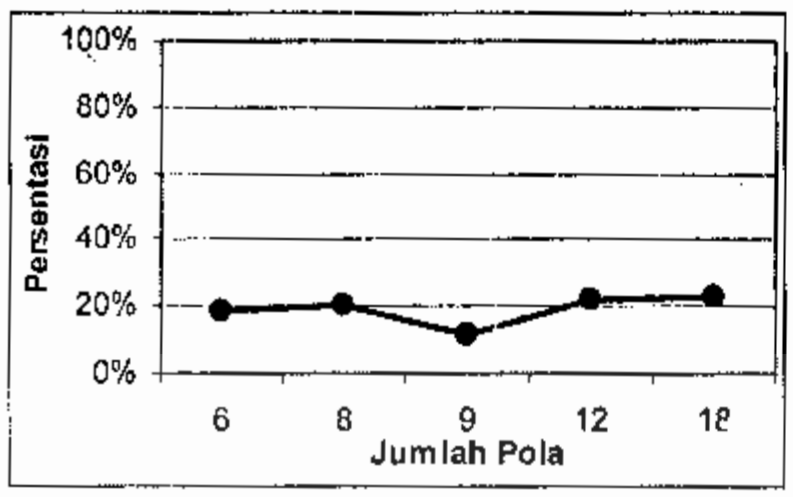
Tabel 3. Rata-rata tingkat pengenalan data pengujian

Jumlah Kelas	Rata-rata tingkat pengenalan data pengujian		
	PNN Standar (0,1)	PNN Standar (0,003)	PNN Teroptimasi
6	75.24%	98.06%	98.31%
8	78.61%	97.55%	97.67%
9	89.61%	99.53%	99.51%
12	77.12%	98.13%	98.02%
18	76.40%	98.07%	98.18%
	79.39%	**98.27%**	**98.34%**

Tabel 4. Rata-rata jumlah penggunaan neuron

Jumlah Kelas	Rata-Rata Jumlah Penggunaan Neuron		
	PNN Standar (0,1)	PNN Standar (0,003)	PNN Teroptimasi
6	583.33	583.33	108.31
8	766.67	766.67	156.47
9	875.00	875.00	97.75
12	1162.50	1162.50	252.75
18	1750.00	1750.00	404.08
	1027.50	**1027.50**	**203.87**

Perbandingan jumlah neuron yang digunakan dapat dilihat pada tabel 5. Dari tabel tersebut bisa dilihat bahwa PNN (PNN-0,1 dan PNN-0,003) mempunyai rata-rata jumlah penggunaan neuron yang sangat tinggi yaitu 1027,5 neuron. Hal ini sangat berbeda dengan PNN-T yang mempunyai rata-rata jumlah penggunaan neuron yang lebih rendah yaitu 203,87 neuron. Telah disebutkan sebelumnya bahwa biaya komputasi dari PNN sebanding dengan jumlah neuron yang digunakan [1] [6]. Sehingga bisa dikatakan bahwa PNN-T mempunyai biaya komputasi yang lebih kecil daripada PNN. Gambar 3 memperlihatkan bahwa jumlah penggunaan neuron oleh PNN-T dibawah 25% dari jumlah neuron yang digunakan oleh PNN.



Gambar 3. Persentasi rata-rata jumlah penggunaan neuron PNN Teroptimasi terhadap PNN Standar

5. KESIMPULAN

Dari penelitian ini dapat ditarik kesimpulan bahwa dalam aplikasi Sistem Penciuman Elektronik, algoritma PNN-T mempunyai tingkat pengenalan yang sama dengan algoritma PNN dengan nilai parameter *smoothing* optimal yang ditentukan melalui pencarian secara manual. Dan lebih baik daripada algoritma PNN dengan sembarang nilai parameter *smoothing*. Dari hasil percobaan juga bisa dilihat bahwa tidak ada pengaruh jumlah kelas terhadap tingkat klasifikasi dalam algoritma PNN baik yang belum dioptimasi maupun yang telah dioptimasi.

Untuk jumlah penggunaan neuron, PNN-T menggunakan neuron yang lebih sedikit daripada PNN. Dimana PNN-T menggunakan jumlah neuron dibawah 25% bila dibandingkan dengan PNN. Karena biaya komputasi sebanding dengan jumlah penggunaan neuron, maka PNN-T mempunyai biaya komputasi yang lebih kecil daripada PNN dalam Sistem Penciuman Elektronik.

Dalam penelitian ini, PNN dan PNN-T menggunakan nilai parameter *smoothing* yang sama untuk setiap kelas. Untuk beberapa kasus hal ini bukanlah pilihan yang baik. Misalkan ada permasalahan 2 kelas, dimana data pelatihan kelas A tersebar secara lebar, sedangkan data pelatihan kelas B terkonsentrasi, dan daerah A dan B berdekatan. Penggunaan nilai parameter *smoothing* yang besar, tidak dapat mencakup kelas secara baik. Tetapi bila digunakan nilai parameter *smoothing* yang kecil, dapat mencakup kedua kelas tetapi memerlukan jumlah data pelatihan yang besar. Hal ini bisa dihindari dengan menggunakan lebih dari satu nilai parameter *smoothing*. Dimana masing-masing kelas menggunakan nilai parameter *smoothing* yang berbeda sehingga dapat mencakup kedua kelas dan data pelatihan yang digunakan bisa lebih kecil. Selain itu, penggunaan *preprocessing* terhadap data yang akan diklasifikasi, seperti *Principal Component Analysis* dan *Fisher Discriminant Analysis* bisa digunakan agar representasi data menjadi lebih baik sehingga bisa meningkatkan tingkat pengenalan.

REFERENSI

[1]Mao, K.Z., K.-C. Tan, and W. Ser, "Probabilistic Neural-Network Structure Determination for Pattern Classification", *IEEE Transaction on Neural Networks*, vol. 11, no. 4, July 2000.

[2]Fausett, Laurene, *Fundamentals of Neural Networks Architecture, Algorithms, and Applications*, Prentice Hall, 1994.

[3]Goldberg, D. E., *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison Wesley, 1989.

[4]Masters, Timothy, *Advanced Algorithms For Neural Networks: A C++ Sourcebook*, John Wiley & Sons, 1995.

[5]Raghu, P. P., and B. Yegnanarayana, "Supervised Texture Classification Using a Probabilistic Neural Network and Constraint Satisfaction Model", *IEEE Transaction On Neural Networks*, vol. 9, no. 3, May 1998.

[6]Jatmiko, Wisnu, Benyamin Kusumoputro, "Pengenalan Aroma Campuran Dengan Probabilistic Neural Pada Sistem Penciuman Elektronik", *Jurnal Ilmu Komputasi dan Teknologi Informasi*, vol. 1, no.1, Mei 2001.

[7] Jatmiko, Wisnu, Benyamin Kusumoputro, "Karakteristik Sistem Penciuman Elektronik Dalam Mengenal Aroma Campuran dan Aroma Komposisi Menggunakan Algoritma Jaringan Neural Buatan", *Jurnal Ilmu Komputasi dan Teknologi Informasi*, vol. 1, no.2, Oktober 2001.