BAB 2

TINJAUAN PUSTAKA

2.1 Tinjauan Pustaka

Segmentasi Citra Teks Sastra Jawa, menggunakan metode profil proyeksi telah dilakukan (Widiarti, 2007) dan tingkat kesuksesan 86,78% mensegmentasi dokumen teks Sastra Jawa. Pengenalan pola huruf Jawa dengan metode *Learning Vector Quantization* (LVQ) dilakukan (Mafrur, et al., 2012). Presentase ketepatan yang diperoleh dari percobaan pada penelitian ini adalah 56,61 %. Pengenalan pola karakter huruf jawa dengan metode *Backpropagation Neural Network* dilakukan oleh (Nurmila, et al., 2010) dan hasilnya rata-rata keakuratan *Backpropagation Neural Network* dalam mengenali pola karakter huruf Jawa adalah sebesar 99.563% untuk data sampel berupa data pelatihan, 61.359% untuk data sampel diluar data pelatihan, dan 75% untuk data sampel data pelatihan dan di luar data pelatihan.

Upaya melestarikan dan mengekstrak pengetahuan implisit naskah Jawa kuno oleh (Karundeng, et al., 2012) berhasil mengevaluasi algoritma ekstraksi fitur antara lain arah jalur lokal, mesh dan pendekatan yang berbeda lainnya dalam hal tingkat klasifikasi *Machine Support Vector* (MSV). Algoritma Widiarti-Winarko (Widiarti & Winarko, 2012) berhasil menerjemahkan dokumen kuno yang ditulis dalam karakter Jawa dengan hasil 62,96%, dan 75% menemukan kata yang benar dalam bahasa Jawa.

Particle Swarm Optimization telah digunakan untuk mengatasi berbagai masalah masalah pada kajian segmentasi dan pengenalan pola (Merwe & Engelbrecht, 2003) (Cheo & Ye, 2004) (Omran, 2004) (Omran, et al., 2005) (Cui, et al., 2005) (Cui & Potok, 2005) (Abraham, et al., 2008) (Murugesan & Palaniswami, 2010) (Mohsen, et al., 2011) (Yih, et al., 2007) (Yih, et al., 2008) (Ye & Chen, 2005) (Lin, et al., 2009) (Santosa & Ningrum, 2009). *Particle Swarm Optimization* bisa melakukan segmentasi citra dengan hasil yang memuaskan (LAI , 2006). PSO memberikan kinerja yang menjanjikan dan perilaku yang stabil dalam mengenali angka tulisan tangan (Ba-Karait & Shamsuddin, Handwritten Digits Recognition using Particle Swarm Optimization, 2008). Penggabungan algoritma *Particle Swarm Optimization* dan *Backpropagation* telah dilakukan (Zhang, et al., 2007). Pengabungan algoritma *Particle Swarm Optimization* dan *Backpropagation* juga telah dilakukan (Lagudu & Sarma, 2013) untuk pengenalan pola tulisan tangan.

Penggunaan teknologi *GPU* banyak diterapkan juga pada algoritma *Particle Swarm Optimization* (Mussi, et al., 2009) (Zhou & Tan, 2009) (Zhou & Tan, 2011). Jika menggunakan NVIDIA CUDA untuk mempararelkan komputasi, PSO berjalan lebih cepat 170% pada *GPU*, daripada berjalan pada *CPU* dengan jumlah partikel 100. (Kristiadi, Pranowo, & Mudjihartono, PARALLEL PARTICLE SWARM OPTIMIZATION FOR IMAGE SEGMENTATION, 2013). Pada permasalahan konvergensi untuk optimalisasi fungsi *Rastrigin* dan fungsi *Ackley* di ruang pencarian 30-dimensi dengan menggunakan *General Purpose Graphics Processing Unit* (*GPGPU*) untuk

pemrosesan paralel dari PSO, fungsi *Rastrigin* berjalan lebih cepat 16 kali dari komputer *cluster* dan untuk masalah *Ackley* itu lebih cepat 8 kali daripada komputer *cluster*. (Liera, et al., 2011). Teknik yang didasarkan pada metode *Sauvola-Pietkinen* untuk mendeteksi tepi dengan kesinambungan menunjukkan bahwa algoritma PSO yang memanfaatkan teknik *thresholding* lokal baru, lebih baik daripada yang menggunakan metode *Otsu* (Setayesh, et al., 2009). Masalah estimasi parameter yang realistis di mana setiap prosesor melakukan perhitungan yang signifikan dengan metode PSO berhasil menunjukkan peningkatan percepatan ditandai dengan ukuran populasi (Wachowiak & Foster, 2012). Implementasi *clustering K-Means* secara pararel telah dilakukan (Farivar, et al., 2008) menghasilkan peningkatan kinerja 13kali.

Telah dilakukan implementasi dari algoritma *backpropagation* pada CUDA. Implementasi diuji dengan dua set patokan data standar dan hasilnya menunjukkan bahwa algoritma pelatihan paralel berjalan 63 kali lebih cepat. (Sierra-Canto, et al., 2010). Hasil komputasi algoritma pelatihan untuk jaringan saraf yang mengandung operasi matriks matematika, sangat baik jika jika menggunakan *GPU* (Kajan & Slačka, 2010). Teknik untuk pelatihan jaringan saraf yang rumit pada *GPU* (tersedia dalam *PC low end*) memungkinkan pelatihan yang dilakukan dengan waktu utilisasi *CPU* yang minimal. (Kharbanda & Campbell, 2011). Penelitian (Wojtera, 2009) pada *neural network convolutional* menggunakan *GPGPU* menghasilkan emulasi yang lebih buruk daripada program yang ditulis *native* untuk *CPU*. Paralelisasi telah dilakukan pada algoritma pelatihan jaringan syaraf pada arsitektur heterogen (Krpan & Jakobovi´ c, 2012).

(Hofmann, 2011) melakukan penelitian menggunakan *GPU* pada model perilaku pasar keuangan menggunakan *Evolving Neural Networks.* CUDA meningkatkan performansi hingga 80% dibandingkan dengan implementasi pada *CPU* pada Simulasi *paralel neural network* (Pendlebury, et al., 2012).

Graphics Processing Unit (*GPU*) bisa mempercepat proses sistem *Optical Character Recognition* (OCR) sekitar 20 kali sampai 30 kali daripada secara serial ketika berjalan pada *GPU GeForce 9500 GT* yang memiliki 32 core. (Singh, et al., 2011). *Graphics Processing Unit* (*GPU*) dan algoritma paralel didasarkan pada pengolahan setiap *pixel* sebagai *thread* yang berbeda mencapai keuntungan kinerja yang signifikan, berjalan sampai 6,8 kali lebih cepat dibandingkan dengan pendekatan sekuensial (Happ, et al., 2012). Segmentasi citra dengan *General purpose computing on the GPU* oleh (Backer, et al., 2013) memberikan keuntungan waktu komputasi. Permasalahan *Robust recognition of arbitrary object classes* menggunakan NVIDIA CUDA framework implementasinya hingga 82 kali lebih cepat dari sistem versi *CPU single-core* dan mencapai tingkat pengujian kesalahan masing-masing sebesar 0,76% dan 2,87% (Uetz & Behnke, 2009).

Pengenalan karakter Aksara Bangla dibuat oleh (Omee, et al., 2011) dengan menyajikan langkah langkah alur kerjanya dan (Bhowmik, et al., 2005) mengunakan jaringan syaraf tiruan. Sistem *Automated Road Sign Recognition* menggunakan *Artificial Neural Network* untuk papan informasi menggunakan Aksara Bengali, OCR menggunakan *Multi Layer Perceptron* menghasilkan akurasi sebesar 91,48%. (Rahman, et al., 2009). (Roy, et al., 2005) menggunakan

algoritma baru pada deteksi *skew* dan cukup berhasil. Ekstrasi teks *devanagari* dan *bangla* pada gambar gambar yang dihasilkan kamera digital menggunakan *Robust filtering binarization* dan duji cobakan pada 100 citra kamera digital. (Bhattacharya, et al., 2009). Segmentasi dan pengenalan aksara cetak bangla (Hasan, et al., 2005) dengan metode *Characteristic functions Hamming network* bisa dilakukan pada karakter cetak *isolated* dan *continuous size*. Pengenalan pola Aksara Bangla dengan metode ANN berhasil baik pada aksara yang tidak terkontaminasi (Upadhyay, et al., 2011).

2.2 Landasan Teori

2.2.1. Pengenalan Karakter Optik (Optical Character Recognition)

Optical Character Recognition (OCR) adalah sebuah aplikasi untuk mengubah dokumen dalam bentuk citra digital ke dalam bentuk dokumen dengan format teks (ASCII) yang dapat dengan mudah diubah isinya. Beberapa keuntungan dari penggunaan OCR, yaitu :

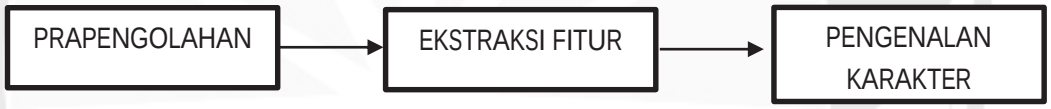
1. Dokumen hasil OCR dapat dengan mudah diubah isinya.
2. Pencarian terhadap isi dokumen menjadi sangat mudah
3. Biaya penyimpanan dokumen menjadi sangat rendah

Berdasarkan cara mendapatkan *input* , *Optical Character Recognition* dapat dikategorikan dalam dua bagian yaitu *offline* dan *online recognition*. Pada *offline recognition*, *input* yang digunakan dipindai menggunakan *scanner* sebelum diproses. Sedangkan dalam *online* r*ecognition, input* langsung didapatkan pada saat *input* tersebut dituliskan diatas layar sentuh (*touch screen*). Kualitas *input*

akan menentukan kinerja dari sistem *Optical Character Recognition* yang dibangun.

2.2.2. Deskripsi Model

Sistem yang diusulkan mengandung tiga subsistem. Subsistem prapengolahan, subsistem ekstraksi fitur dan subsistem pengenalan karakter. Pada subsistem prapengolahan, sistem akan melakukan segmentasi pada citra menggunakan metode *Particle Swarm Optimization*. Pada subsistem ekstraksi fitur, setiap karakter Aksara Jawa akan dikonversi menjadi nilai vektornya. Pada subsistem pengenalan karakter, akan digunakan metode jaringan syaraf tiruan *backpropagation* untuk mengenali karakter Aksara Jawa.



Gambar 2. 1. Alur Proses Pengenalan Karakter

2.2.3. Aksara Jawa

Masyarakat Jawa sudah mempunyai bentuk penulisan aksara, yang dianggap adiluhung leluhur bangsa Jawa hingga kini. Aksara Jawa yang menjadi bagian tak terpisahkan dari bahasa Jawa dan merupakan salah satu unsur kebudayaan dari masyarakat Jawa. Perkembangan Aksara Jawa juga ada kaitan dengan perkembangan bahasa Jawa yang lahir sebagai alat komunikasi masyarakat. Namun, Aksara Jawa dipercayai muncul setelah berlaku pertemuan antara peradaban Jawa dengan India. Sebelum itu, tidak terdapat bukti yang menunjukkan bangsa Jawa mempunyai aksaranya sendiri. Aksara Jawa yang ada

pada masa kini adalah hasil daripada pembentukan kembali bentuk dan gaya Aksara Jawa Kuna (kuno). (Mohamed, 2001)

Aksara Jawa merupakan turunan Aksara Brahmi sebagaimana Aksara Nusantara lainnya, memiliki kedekatan dengan Aksara Bali. Aksara Jawa memiliki bentuk yang lebih rumit dibandingkan dengan Aksara Latin biasa. Aksara Jawa terdiri atas aksara dasar yang disebut dengan Aksara *Carakan* atau *nglegeno* atau Aksara Jawa tanpa *sandangan*, Aksara Pasangan, Aksara Swara, Aksara Rekan, Aksara Murda, Wilangan atau Angka, dan Sandangan atau tanda baca (Nugraha, 2009).

Aksara Jawa sudah mendapat pengakuan resmi dari Unicode, lembaga di bawah naungan UNESCO di Unicode Standard versi 5.2 dengan titik kode A980 sampai A9DF. Pengakuan ini diberikan pada 2 Oktober 2009, bersamaan dengan penetapan batik sebagai warisan budaya tak benda Indonesia oleh Organisasi Pendidikan, Ilmu Pengetahuan, dan Kebudayaan Perserikatan Bangsa-Bangsa (UNESCO). Dengan pengakuan itu, kini aksara Jawa bisa dipakai untuk komputer yang setara dengan huruf lain di dunia yang telah digunakan dalam komputer, seperti Latin, Cina, Arab, dan Jepang. (Nugraha, 2009)

HA	NA	CA	RA	KA
DA	TA	SA	WA	LA
PA	DHA	JA	YA	NYA
MA	GA	BA	THA	NGA

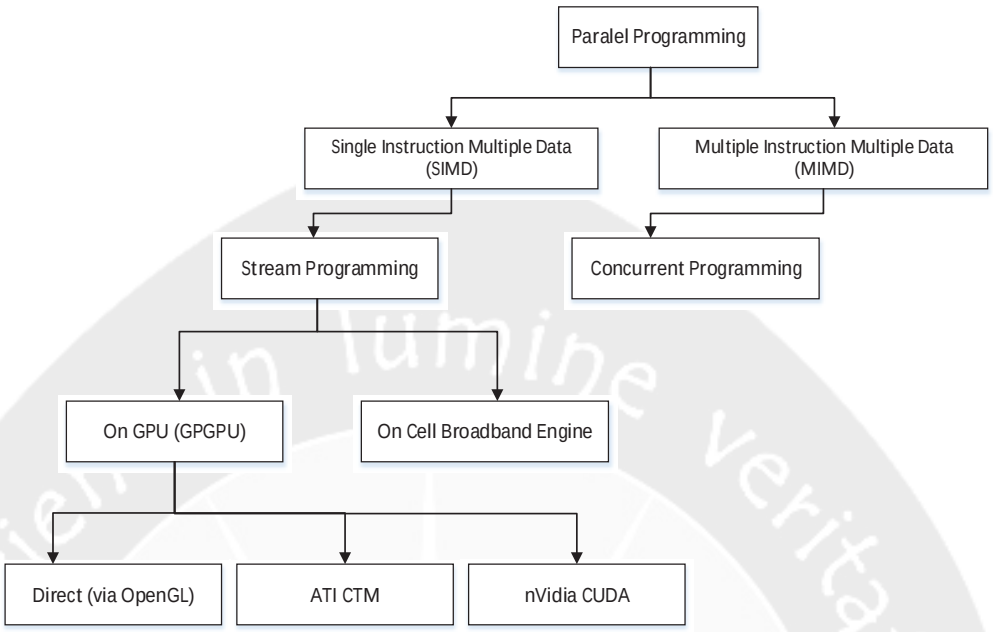
Gambar 2. 2. Duapuluh Aksara Jawa Dasar (Aksara Nglegéna) Dan Pengucapannya

2.2.4. Parallel computing

Komputasi paralel merupakan salah satu teknik melakukan komputasi secara bersamaan dengan memanfaatkan beberapa komputer juga secara bersamaan. Pada komputasi paralel dibutuhkan kapasitas yang sangat besar untuk memproses komputasi yang banyak. Di samping itu pemakai harus membuat pemrograman paralel untuk dapat merealisasikan komputasi. (Sanders & Kandrot, 2011)

Pemrograman paralel memiki tujuan utama yaitu untuk meningkatkan performa komputasi karena semakin banyak hal yang bisa dilakukan secara bersamaan dalam waktu yang sama, maka semakin banyak pula pekerjaan yang bisa diselesaikan (Kirk & Hwu, 2010).

Stream Programming adalah sebuah teknik pemrograman konkruen yang dapat digunakan untuk melakukan komputasi paralel. Dalam *stream programming*, data diorganisasi menjadi sekumpulan *stream*, dan sebuah program yang disebut *kernel* diaplikasikan pada setiap elemen *stream.* Dari gambar 2.2 dapat dilihat bahwa s*tream programming* menggunakan paradigma *Single Instruction Multiple Data* (SIMD). Artinya, program yang sama dieksekusi pada sekumpulan data secara paralel. (Kirk & Hwu, 2010)



Gambar 2. 3. *Stream Programming* Dalam Taksonomi *Parallel Programming*

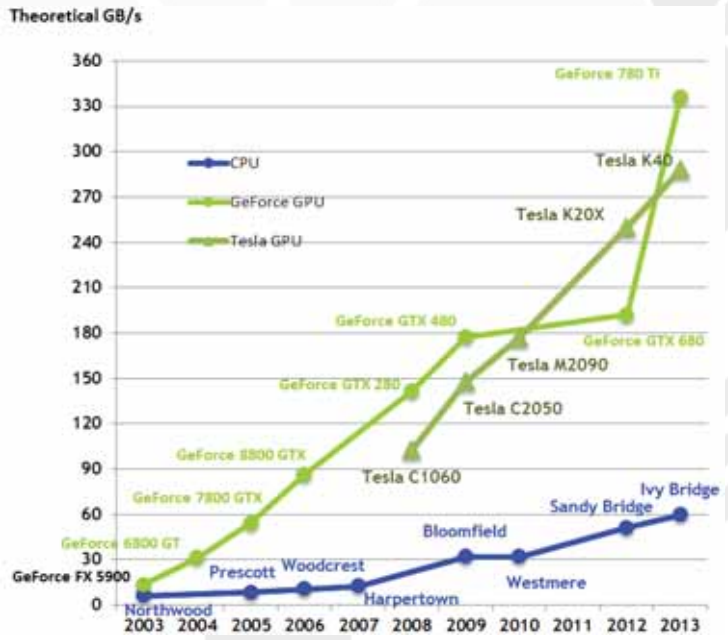
Salah satu perangkat keras yang dapat digunakan untuk melakukan stream programming adalah *Graphics Processing Unit* (GPU). GPU sangat cocok untuk diprogram dengan teknik ini, karena GPU dirancang dari awal untuk melakukan proses *rendering* 3D, dimana proses tersebut membutuhkan *hardware* yang dirancang secara khusus untuk memproses data secara SIMD. Untuk mengimplementasikan stream programming pada GPU, ada banyak framework yang dapat dipakai. Salah satunya adalah *Compute Unified Device Architecture* (CUDA). (CUDA™, 2012)

2.2.5. Kemampuan Komputasi GPU

Graphic Processing Unit (GPU) adalah salah satu *peripheral* komputer yang bertugas untuk melakukan proses *rendering* objek 3D ke layar. Karena *rendering* merupakan proses yang sangat paralel, maka evolusi arsitektur GPU

pun mengikutinya: GPU berevolusi menjadi sebuah multiprosesor yang mampu menjalankan tugas secara paralel.

GPU dapat disebut unit komputasi yang terpisah dari CPU karena GPU memiliki prosesor dan memori sendiri. GPU memiliki jumlah *Arithmetic Logic Unit* (ALU) yang jauh lebih besar dibandingkan komputer biasa. Oleh karena itu, kemampuan GPU dalam mengolah data yang besar dapat diandalkan. Pada gambar 2.3, dapat dilihat bahwa pertumbuhan kemampuan GPU jika diukur dengan *Floating Point Operations per Second* (FLOPS) jauh meninggalkan CPU, bahkan meninggalkan hukum Moore.



Gambar 2. 4. Peningkatan Kecepatan GPU

2.2.6. *Compute Unified Device Architecture* (CUDA)

CUDA™ adalah *platform* komputasi paralel dan model pemrograman yang memungkinkan peningkatan dramatis dalam kinerja komputasi dengan

memanfaatkan kekuatan dari *graphics processing unit* (*GPU*). (Kirk & Hwu, 2010)

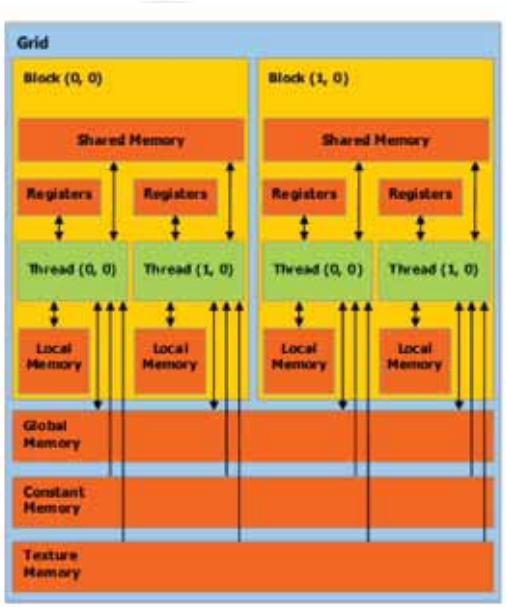
Sejak diperkenalkan pada 2006, CUDA telah banyak disebarkan melalui ribuan aplikasi dan diterbitkan makalah penelitian, dan terinstal lebih dari 300 juta CUDA-*enabled GPU* di *notebook, workstation*, komputer *cluster* dan super-komputer. *GPU* diaplikasikan juga untuk astronomi, biologi, kimia, fisika, *data mining*, manufaktur, keuangan. (Sanders & Kandrot, 2011)

Gambar 2.4 menerangkan tentang hirarki memori dalam CUDA. Setiap *stream processor*, yang dianalogikan dengan thread(x,y) dikelompokan dalam *block block*. Setiap *block* memiliki memori bersama yang dinamakan *shared memory*. *Shared memory* adalah *memory* berukuran kecil berkecepatan tinggi yang hanya dapat diakses *stream processor* dalam *block* yang sama. *Bandwidth* antara *stream processor* dan *shared memory* sangat besar, mencapai 70,7 GB/s. Jenis memory lain adalah *device memory*. *Memory* ini dapat diakses oleh seluruh *multiprocessor*, namun dengan *bandwidth* yang lebih lambat. Adapun *memory* komputer diistilahkan dengan *host memory*.

Implementasi konsep stream programming pada CUDA sebagai berikut :

1. *Stream*: Dalam CUDA, *stream* disimpan dalam sebuah memori milik GPU yang bernama *global memory*. *Stream* diorganisasi berdasarkan jumlah *thread* yang akan mengaksesnya secara paralel. Setelah itu, *thread* tersebut harus diorganisasikan menjadi *thread block*.

2. *Kernel*: Dalam CUDA, *kernel* tetap merupakan sebuah program yang beroperasi pada data yang terdapat pada *global memory*.



Gambar 2. 5. Hirarki Memori Pada CUDA

Langkah langkah umum untuk melakukan komputasi pada CUDA adalah sebagai berikut:

1. Tentukan jumlah *thread* yang kan beroperasi pada *stream*. Kemudian tentukan jumlah *thread per block* dan jumlah *thread block* yang efisien pada hardware CUDA. Jumlah tersebut dinamakan *execution configuration*.
2. Pindahkan semua data dari *host memory* ke *device memory*. Ini harus dilakukan karena GPU tidak bisa membaca *host memory*.
3. Lakukan komputasi pada *device memory* beserta *execution configuration* nya. Data dalam *device memory* ini akan menjadi *stream*.
4. Pindahkan hasil komputasi pada *device memory* ke *host memory*.

2.2.7. C / C++

Tahun 1969, laboratorium Bell AT&T di Muray Hill, New Jersey menggunakan bahasa *assembly* ini untuk mengembangkan sistem operasi UNIX. Maksudnya adalah untuk membuat sistem operasi yang dapat bersifat *'programmer-friendly'*. Setelah UNIX berjalan, Ken Thomson, seorang pengembang sistem di laboratorium tersebut mengembangkan bahasa baru dengan nama bahasa B. Huruf B ini diambil dari BCPL. Bahasa B ini kemudian digunakan untuk menulis ulang atau merevisi sistem operasi UNIX. Oleh karena bahasa B ini masih bersifat interpret dan lambat, maka pada tahun 1971, sistem operasi UNIX kemudian ditulis ulang dengan menggunakan bahasa C, yaitu bahasa pemrograman yang dikembangkan oleh Dennis Ritchie, seorang pengembang sistem di laboratorium yang sama.

Sampai sekarang bahasa C masih digunakan untuk melakukan pengembangan-pengembangan program dan sistem-sistem operasi, diantaranya sistem operasi Windows. Alasan itulah yang menjadikan bahasa C sangat populer di dunia pemrograman khususnya untuk industri perangkat lunak. Namun sayangnya bahasa C merupakan bahasa yang masih tergolong susah untuk dipelajari karena masih bersifat prosedural murni. Untuk membentur satu objek, kita harus melakukan banyak sekali penulisan kode. Hal ini tentu dapat dikatakan sebagai sebuah kelemahan. Untuk mengatasi masalah ini, pada tahun 1983, seorang doktor bernama Bjane Stroustrup yang juga bekerja di laboratorium yang sama menciptakan bahasa baru yaitu bahasa C++ yang meripakan *hybrid* dari bahasa C. Bahasa C++ didasarkan atas bahasa C sehingga kita dapat melakukan

kompilasi program-program yang ditulis dalam bahasa C dengan menggunakan kompiler C++. Keistimewaan dari bahasa C++ adalah karena bahasa ini mendukung pemrograman berarah objek atau lebih sering dikenal dengan istilah *Object Oriented Programming* (OOP).

2.2.8. Citra

Citra merupakan hasil keseluruhan dari suatu sistem perekaman data. Citra secara teoritis dapat dikelompokan menjadi dua jenis, yaitu citra kontinu dan citra diskrit (*digital images*). Citra kontinu dihasilkan dari dari suatu sistem optik yang menerima sinyal analog. Contoh hasil citra kontinu adalah bayangan objek pada retina mata manusia dan citra yang dihasilkan oleh kamera analog. Citra diskrit dihasilkan dari digitalisasi citra kontinu. Contoh citra diskrit adalah citra hasil kamera digital dan citra hasil *scanning scanner*. (Gonzalez & Woods, 2008)

2.2.8.1. Pengertian Citra Digital

Menurut (Solomon & Breckon, 2011) sebuah citra digital merupakan citra yang dinyatakan secara diskrit, baik untuk posisi koordinatnya maupun warnanya. Citra digital dapat direpresentasikan oleh sebuah matrik, dengan elemen matrik menyatakan titik sebuah warna. Elemen elemen matrik menyatakan titik sebuah warna. Elemen elemen matrik pada citra digital disebut dengan istilah *pixel* yang berasal dari kata *picture element*.

Pixel pada sebuah citra dapat direpresentasikan dengan sebuah titik $l(m, n)$ dimana m dan n merupakan ukuran citra digital pada koordinat spasial (*cartesian coodinat 2D*).

Gambar 2. 6. 2D Koordinat Spasial Sebuah Citra Digital Dengan Ukuran M×N *Pixel*

Selain pemilihan format citra, kualitas sebuah citra digital juga ditentukan oleh tipe data citra (*image data types*). Tipe data citra berpengaruh pada resolusi dan tempat yang dibutuhkan untuk menyimpan citra tersebut (*storage*).

2.2.8.2. Citra Monokrom (*Binary Image*)

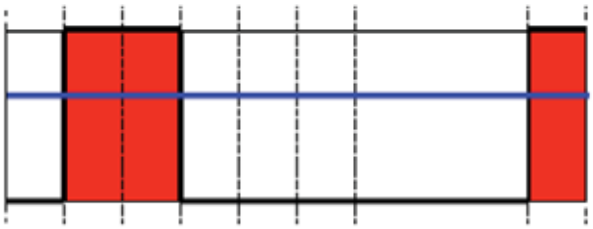
Citra biner merupakan citra yang didefinisikan oleh dua level warna yakni hitam dan putih atau 0 (*false*) dan 1 (*true*). Untuk mengkonfersi citra yang memiliki warna keabuan ke warna biner bisa menggunakan proses *binary thresholding*. Secara sederhana, *binary thresholding* digunakan untuk menghilangkan informasi citra yang memiliki derajat keabuan dibawah *threshold* yang ditentukan. *Binary thresholding* dibagi menjadi dua tipe, yaitu :

a. *Threshold Binary*

Operasi ini dapat diekspesikan sebagai berikut :

$$dst(x,y) = \begin{cases} 1, src(x,y) > T \\ 0, src(x,y) \leq T \end{cases}$$

Sehingga jika nilai pixel $src(x,y)$ lebih besar dari nilai threshold, pixel baru diinisialisasi dengan nilai 1. Sedangkan untuk nilai lainnya diinisialisasi dengan nilai 0.



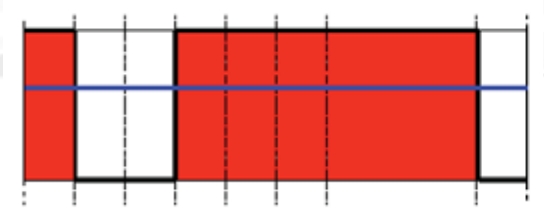
Gambar 2. 7. *Threshold Binary*

b. *Threshold Binary, Inverted*

Operasi ini dapat diekspesikan sebagai berikut :

$$dst(x,y) = \begin{cases} 0, src(x,y) > T \\ 1, src(x,y) \leq T \end{cases}$$

Sehingga jika nilai pixel $src(x,y)$ lebih besar dari nilai threshold, pixel baru diinisialisasi dengan nilai 0. Sedangkan untuk nilai lainnya diinisialisasi dengan nilai 1.



Gambar 2. 8. *Threshold Binary Inverted*

2.2.8.3. Citra Abu – abu atau *Intensity*

Citra dengan derajat keabuan berbeda dengan citra RGB, citra ini didefinisikan oleh satu nilai derajat warna. Umumnya bernilai 8 bit sehingga intensitas kecerahan warna sampai 256 level dan kombinasi warnanya 256 varian. Tingkat kecerahan paling rendah yaitu 0 untuk warna hitam dan putih bernilai 255.

Untuk mengkonfersikan citra yang memiliki warna RGB ke derajat keabuan bisa menggunakan :

$$Gray = \frac{R+G+B}{3} \quad (1)$$

atau

$$Gray = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B \quad (2)$$

2.2.8.4. Citra berwarna

Citra RGB dibangun oleh 3 *channel* matrik warna berukuran $m \times n$, yaitu *red channel*, *green channel* dan *blue channel*. Citra RGB mempunyai kualitas grafik tingkat tinggi (*high quality raster graphic*) dengan mode penyimpanan 24 bit untuk setiap *pixel*. Setiap *channel* warna merah, hijau dan biru masing masing mendapat mendapatkan alokasi 8 bit untuk menghasilkan warna. Pada sistem warna RGB, tiap *pixel* akan dinyatakan dengan tiga parameter bukan nomor warna, setiap warna mempunyai *range* nilai 0 sampai 255 atau mempunyai nilai derajat keabuan $256 = 2^8$. dengan demikian, *range* warna yang digunakan adalah $(2^8)(2^8)(2^8) = 2^{24}$.

2.2.8.5. Format File Citra

Sebuah format citra harus dapat menyatukan kualitas citra, ukuran file dan kompatibilitas dengan berbagai aplikasi. Saat ini tersedia berbagai format grafis, diantaranya yang terkenal adalah BMP, JPEG dan GIF. Setiap format file citra mempunyai kelebihan dan kekurangan dibandingkan satu sama lain.

2.2.9. Segmentasi Karakter

Segmentasi karakter adalah proses yang penting dalam langkah *preprocessing* untuk sistem pengenalan karakter (Bhowmik, et al., 2005). Segmentasi adalah proses pemecahan citra ke dalam obyek-obyek yang terkandung di dalamnya (Widiarti, 2007). Salah satu metode untuk melakukan segmentasi adalah *clustering* (Kristiadi, Pranowo, & Mudjihartono, PARALLEL PARTICLE SWARM OPTIMIZATION FOR IMAGE SEGMENTATION, 2013).

2.2.9.1. *Clustering*

Clustering merupakan proses untuk mengidentifikasi pengelompokan di dalam multidimensi berdasarkan suatu kemiripan tertentu. *Clustering* banyak digunakan untuk penambangan data, analisis data statistik, analisis citra, pengenalan pola, *information retrieval*, dan bioinformatika (Merwe & Engelbrecht, 2003).

Kualitas suatu algoritma *clustering* ditentukan oleh dua faktor, yaitu kemiripan suatu data yang ada di dalam suatu *cluster* yang sama dan ketidakmiripan suatu data dengan data lain yang ada pada *cluster* yang berbeda. Semakin mirip data yang ada dari dalam suatu *cluster* yang sama, maka kualitas algoritma tersebut semakin baik. Semakin tidak mirip suatu data dengan data lain yang ada di *cluster* yang berbeda, maka kualitas algoritma tersebut semakin baik. Ukuran kualitas algoritma *clustering* yang biasa digunakan adalah *Quantization Error* (Omran, et al., 2005).

2.2.9.2. *Particle Swarm Optimization*

Particle Swarm Optimization adalah algoritma optimasi sederhana namun *powerful*, diperkenalkan oleh Kennedy dan Eberhart pada tahun 1995 (Mussi, et al., 2009). PSO merupakan metode optimasi yang memanfaatkan kecerdasan kawanan dan terinspirasi dari perilaku dan gerakan kawanan hewan, contohnya perilaku kawanan burung saat mencari makanan. Setiap objek hewan disederhanakan menjadi sebuah partikel. Setiap partikel berperilaku secara terdistribusi dengan cara menggunakan kecerdasannya sendiri dan dipengaruhi perilaku kawanannya. Dengan demikian, jika satu partikel atau seekor burung menemukan jalan yang tepat atau pendek menuju kesumber makanan, sisa kawanan yang lain juga dapat segera mengikuti jalan tersebut meski lokasi mereka jauh dari kawanan tersebut.

Setiap partikel bergerak dengan kecepatan v. Jika suatu partikel memiliki kecepatan yang konstan maka jika jejak posisi suatu partikel divisualisasikan akan membentuk garis lurus. Dengan adanya faktor eksternal yang membelokkan garis tersebut yang kemudian menggerakkan partikel dalam ruang pencarian maka diharapkan partikel dapat mengarah, mendekati dan pada akhirnya mencapai titik optimal. Faktor eksternal yang dimaksud antara lain posisi terbaik yang pernah dikunjungi suatu partikel, posisi terbaik seluruh partikel (diasumsikan setiap partikel mengetahui posisi terbaik setiap partikel yang lainnya), serta faktor kreativitas untuk melakukan eksplorasi.

Dalam konteks optimasi multivariabel, kawanan diasumsikan mempunyai ukuran tertentu atau tetap dengan setiap partikel posisi awalnya terletak di suatu

lokasi yang acak dalam ruang multidimensi. Setiap partikel bergerak dalam ruang/*space* tertentu dan mengingat posisi terbaik yang pernah dilalui atu ditemukan terhadap sumber makanan atau fungsi nilai objektif. Setiap partikel menyampaikan informasi atau posisi terbaiknya kepada partikel yang lain dan menyesuaikan posisi dan kecepatan masing masing berdasarkan informasi yang diterima mengenai posisi terbaik tersebut.

Perilaku burung burung seperti bisa dilihat di gambar 2.6. mengikuti kebiasaan sebagai berikut:

a. Seekor burung tidak berada terlalu dekat dengan burung yang lain.
b. Burung tersebut akan mengarahkan terbangnya ke arah rata rata keseluruhan burung.
c. Akan memposisikan diri dengan rata rata posisi burung yang lain dengan menjaga sehingga jarak antar burung dalam kawanan itu tidak terlalu jauh.

Gambar 2. 9. Perilaku Kerumunan Kawanan Burung
(sumber : http://www.thepistrophy.com/bird-ballet-swarming-video-by-neels-castillon/)

Dengan begitu perilaku kawanan burung akan didasarkan pada kombinasi dari 3 faktor sederhana berikut :

1. Kohesi – terbang bersama;
2. Separasi – jangan terlalu dekat;
3. Penyesuaian (*alignment*) – mengikuti arah bersama.

Jadi PSO dikebangkan dengan berdasarkan model berikut :

1. Ketika seekor burung mendekati target atau makanan (atau bisa *minimum* atau *maximum* suatu fungsi tujuan) secara cepat mengirimkan informasi kepada burung burung yang lain dalam kawanan tertentu;
2. Burung yang lain akan mengikuti arah menuju ke makanan tetapi tidak secara langsung;
3. Ada komponen yang tergantung pada pikiran setiap burung, yaitu memorinya tentang apa yang sudah dilewati pada waktu sebelumnya.

Pencarian PSO untuk optima dari suatu fungsi, disebut fungsi *fitness*. Posisi terbaik dari suatu individu dalam *swarm* tersebut disimpan, dan dinamakan dengan pengalaman (*experience*) dari pertikel tersebut. Pengalaman pengalaman dari suatu partikel pada *swarm* dikomunikasikan ke semua partikel yang ada, sehingga gerakan gerakan partikel pada *swarm* akan cenderung menuju ke suatu arah berdasarkan pengalaman pengalaman yang ada.

Algoritma ini dimulai dengan membuat himpunan partikel partikel secara acak yang dinamakan dengan *swarm*. Kemudian setiap partikel tersebut mengkalkulasi kecepatan dan posisi barunya untuk setiap dimensi yang ada pada ruang pencarian. Kecepatan partikel dipengaruhi oleh 3 faktor, yaitu kecepatan

saat itu, pengalaman partikel tersebut, dan posisi terbaik dari semua partikel yang ada pada *swarm*. Sehingga untuk partikel *i*, pada dimensi *d*, kecepatannya dapat dikalkulasi dengan persamaan berikut :

$$V_{id} = W * V_{id} + c_1 * r_1 * (P_{id} - X_{id}) + c_2 * r_2 * \left(P_{gd} - X_{id}\right) \text{ ……...…(3)}$$

Dimana W adalah *inertia weight* yang digunakan untuk menyeimbangkan kemampuan partikel dalam mencari solusi optimal global dan lokal, c_1 dan c_2 merupakan konstanta non negatif. r_1 dan r_2 merupakan bilangan acak diantara 0 dan 1. P_{id} merupakan posisi terbaik partikel tersebut berdasarkan pengalamannya, P_{gd} merupakan posisi terbaik dari semua partikel yang ada (*global*) pada dimensi d, dan X_{id} merupakan posisi partikel saat itu.

Kecepatan tersebut akan digunakan untuk menentukan posisi baru dari partikel tersebut. Posisi baru dari partikel adalah posisi partikel pada saat itu ditambahkan dengan kecepatan geraknya. Sehingga dapat dikalkulasi dengan persamaan berikut :

$$X_{id} = X_{id} + V_{id} \text{ ……..(4)}$$

Dimana X_{id} merupakan posisi partikel, dan V_{id} merupakan kecepatan partikel (Kristiadi, Pranowo, & Mudjihartono, PARALLEL PARTICLE SWARM OPTIMIZATION FOR IMAGE SEGMENTATION, 2013).

2.2.9.3. *PSO Clustering*

Menurut (Merwe & Engelbrecht, 2003) dalam konteks *clustering*, suatu partikel tunggal merepresentasikan *centroid cluster* sebanyak N_c. Sehingga untuk setiap partikel X_1, dapat direpresentasikan sebagai berikut :

$$X_1 = (m_{i1}, \dots\dots, m_{ij}, \dots\dots m_{iN_c}) \quad \text{.....(5)}$$

dimana m_{ij} merupakan *centroid cluster* ke-*j* pada partikel ke-*I*, pada cluster C_{ij}. Dari representasi artikel di atas, maka dapat disimpulkan bahwa kumpulan (*swarm*) dari *p centroid* merepresentasikan sejumlah *clustering*, sebanyak jumlah partikel, pada data tertentu.

Fungsi *fitness* dari suatu partikel dapat dihitung dengan menggunakan quantization error yang dinyatakan sebagai berikut :

$$f = \frac{\sum_{j=1}^{N_c} \frac{\sum_{\forall Z_p \in Z} d(Z_p, m_j)}{|Z|}}{N_c} \quad \text{.....(6)}$$

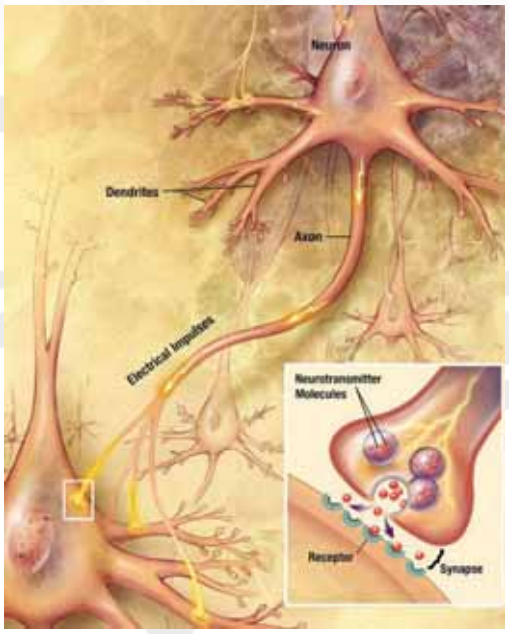
dimana, *Z* merupakan kumpulan data yang di *cluster*, dan *d* merupakan fungsi jarak *euclidean* yang dinyatakan dalam persamaan berikut :

$$d(Z_p, m_j) = \sqrt{\sum_{i=1}^{d_m} (Z_{pi} - m_{ji})^2} \quad \text{.....(7)}$$

dimana d_m merupakan dimensi dari data yang di *cluster*.

2.2.10. Jaringan Syaraf Tiruan (JST)

Aktivitas berpikir dan bertingkah laku diyakini dikendalikan oleh otak dan seluruh sistem syaraf. Kemampuan untuk belajar dan bereaksi terhadap suatu perubahan di dalam lingkungan, tentunya memerlukan suatu yang dinamakan kecerdasan. Jaringan Syaraf Tiruan (JST), merupakan model atau pola dalam pemrosesan informasi. Model ini terinspirasi dari sistem biologi syaraf mahluk hidup seperti pemrosesan informasi pada otak. Struktur pemrosesan informasi ini terbentuk dari sejumlah *neuron* yang saling terhubung satu sama lain dan memecahkan masalah secara bersamaan. JST pertama kali diperkenalkan oleh seorang neurolog Warren S. McCulloch dan matematikawan Walter Pitts pada tahun 1943 yang kemudian dikenal dengan *McCulloch-Pitts Neuron*.



Gambar 2. 10. Model Biologi Neuron (sumber: http://en.wikipedia.org/wiki/Neuron)

Node dalam JST didasari oleh representasi matematis yang disederhanakan dari *neuron* yang sebenarnya. Komputasi *neural* menggunakan konsep dari sistem

syaraf biologis untuk mengimplemantasikan simulasi *software* dari proses paralel yang besar, yang melibatkan elemen elemen proses (disebut *neuron* atau *neurodes*) yang terhubung satu sama lain dalam arsitektur jaringan. *Neurode* adalah analogi dari *neuron* biologis, menerima *input* yang merepresentasikan impuls impuls elektris, yang diterima oleh *neuron* biologis dari *neuron-neuron* lainnya. *Output* dari *neurode* dapat disamakan dengan sinyal yang dikirim dari neuron biologis melalui *axon*. *Axon* pada neuron biologis bercabang menuju *dendrite neuron* lain, dan impuls impuls ditransmissikan melalui *synaps*. Sebuah *synaps* dapat meningkatkan atau menurunkan kekuatannya, dengan demikian, mempengaruhi perambatan level sinyal. Hal tersebut mengakibatkan pembangkitan sinyal (*excitation)* atau penghalangan sinyal (*inhibition*) ke *neuron* berikutnya.

Kemampuan untuk belajar berdasarkan adaptasi adalah faktor utama yang membedakan sistem syaraf tiruan dari aplikasi sistem pakar. Sistem pakar diprogram untuk membuat kesimpulan (*inference*) berdasarkan data yang menjelaskan lingkungan. Permasalahan sistem syaraf tiruan, dipihak lain, dapat menyesuaikan bobot *node* sebagai tanggapan *input* dan mungkin pada *output* yang diinginkan.

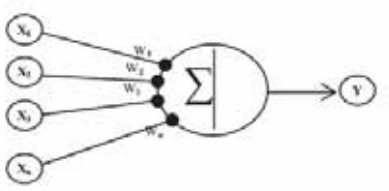
Penelitian dan pengembangan JST menghasilkan sistem yang menarik dan bermanfaat yang menggunakan beberapa fitur dari sistem biologis, meskipun sebenarnya kita masih jauh dari pembuatan mesin seperti otak (*brain like machine*). Bidang penerapan komputasi neural terus berkembang, dengan lebih banyak penelitian dan pengembangan yang dibutuhkan untuk meniru otak

manusia. Tetapi waluapun demikian, banyak teknik-teknik yang bermanfaat, yang terinspirasi dari sistem biologis, yang digunakan dalam aplikasi aplikasi nyata. Tabel menunjukan perbandingan antara BNN dan JST.

Tabel 2. *Biological Neural Network* vs Jaringan Syaraf Tiruan

Fakta fakta tentang Neuron	
BNN	JST
Soma	*Node*
Dendrit	*Input*
Axon	*Output*
Synaps	*Weight*
Kecepatan rendah	Kecepatan tinggi
Neuron banyak	*Neuron* beberapa

Pada gambar 2.9 diperlihatkan bahwa Jaringan syaraf tiruan terdiri atas satuan-satuan pemroses berupa *neuron*. Y sebagai output menerima input dari *neuron* $X_1, X_2, X_3, \ldots, X_n$ dengan bobot $W_1, W_2, W_3, \ldots, W_n$ Hasil penjumlahan seluruh *impuls neuron* dibandingkan dengan nilai ambang tertentu melalui fungsi aktivasi f setiap *neuron*. Fungsi aktivasi digunakan sebagai penentu keluaran suatu *neuron*.



Gambar 2. 11. Proses Komunikasi Antar Neuron

Salah satu metode pelatihan dalam Jaringan Syaraf Tiruan adalah pelatihan terbimbing (*supervised learning*). Pada pelatihan terbimbing diperlukan sejumlah

masukan dan target yang berfungsi untuk melatih jaringan hingga diperoleh bobot yang diinginkan. (Nurmila, et al., 2010)

2.2.10.1. Dasar dasar Komputasi Neural

2.2.10.1.1. Neurode

Sebuah JST terdiri dari unit unit dasar yang disebut *artificial neurons* atau *neurodes*, yang merupakan elemen pemroses dalam sebuah jaringan. Setiap *neurode* menerima input data, memprosesnya, kemudian mengeluarkan sebuah *output* tunggal. *Input* bisa berasal dari data mentah maupun output *neuron* lainnya. *Output* bisa merupakan produk final, ataupun menjadi input bagi *neuron* lainnya.

2.2.10.1.2. Jaringan

Sebuah JST terdiri dari kumpulan *neuron neuron* yang terhubung, yang sering dikelompokkan dalam lapisan lapisan (*layers*). Pada umumnya tidak ada asumsi yang spesifik mengenai arsitektur jaringannya. Bermacam macam topologi jaringan syaraf merupakan pokok persoalan dari penelitian dan pengembangan JST. Dalam hal arsitektur berlapis, ada dua struktur dasar. Dua struktur dasar tersebut adalah:

a. Struktur dua lapisan : *input* dan *output.*
b. Stuktur tiga lapisan : *input, intermediate* (disebut juga *hidden*) dan *output.*

Lapisan *input* (*input layer*) menerima data dari luar dan mengirimkan ke sinyal ke lapisan berikutnya. Lapisan terluar mengintepretasikan sinyal dari

lapisan sebelumnya, untuk kemudian menghasilkan hasil yang ditransmisikan keluar sebagai pemahaman jaringan terhadap *input* data yang diterimanya.

2.2.10.1.3. Input

Sebuah *input* dapat disamakan dengan sebuah atribut tunggal dari suatu pola atau data lainnya dari dunia luar. Jaringan dapat dirancang untuk menerima sekumpulan nilai *input* yang berupa nilai *biner* atau kontinu. Perlu dicatat, bahwa dalam komputasi *neural*, *input* hanya berupa bilangan. Jadi apabila terdapat masalah yang bersifat kualitatif atau berupa grafik, maka informasi tersebut harus diproses terlebih dahulu untuk menghasilkan suatu nilai numerik yang ekivalen sebelum dapat diintrepretasikan oleh JST.

2.2.10.1.4. Output

Output dari sebuah jaringan adalah solusi dari masalah. Tujuan dari sebuah jaringan adalah untuk menghitung nilai *output*. Dalam tipe JST *supervised*, *output* awal dari jaringan biasanya tidak tepat, dan jaringan harus disesuaikan sampai didapatkan *output* yang benar.

2.2.10.1.5. Hidden Layer

Dalam arsitektur banyak lapisan (*multi-layered*), *hidden layers* tidak berinteraksi secara langsung dengan dunia luar, tetapi menambah tingkat kompleksitas agar JST dapat beroperasi dalam masalah yang lebih kompleks. *Hidden layer* menambah sebuah representasi internal dari suatu masalah, yang dapat menjadikan jaringan memiliki kemampuan untuk berurusan secara *robust* dengan masalah yang bersifat komplek dan *non linear*.

2.2.10.1.6. Bobot (*weight*)

Weight atau bobot dalam JST mengungkapakan kekuatan relatif (atau nilai matematis) dari berbagai koneksi yang mentransfer data dari *layer* ke *layer*. Dengan kata lain, bobot mengungkapakan kepentingan relatif dari setiap *input* ke dalam elemen proses (*neuron*). Bobot sangat penting untuk JST, karena dengan bobot ini jaringan disesuaikan secara berulang untuk menghasilkan *output* yang diinginkan, dengan bobot, demikian juga membuat jaringan untuk belajar. Tujuan dari melatih JST adalah untuk menemukan himpunan bobot yang akan dapat menginterpretasikan data *input* dari suatu masalah dengan tepat.

2.2.10.1.7. Fungsi Penjumlahan (*Summation Function*)

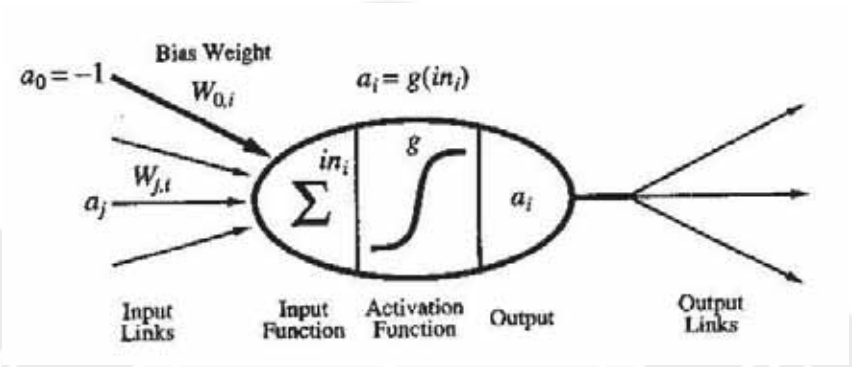
Summation function menghitung rata rata bobot dari semua elemen input. Sebuah *summation function* sederhana akan mengalikan setiap nilai *input* dengan bobotnya dan menjumlahkannya untuk *weighted sum*.

Dengan demikian *neurodes* dalam sebuah JST memiliki kebutuhan pemrosesan yang sederhana. Intinya, mereka harus memantau sinyal yang datang dari *neurodes* lain, menghitung *weighted sum*, dan kemudian mennetukan sinyal yang cocok untuk dikirimkan ke *nurodes* yang lainnya.

2.2.10.1.8. Fungsi Transfer atau Aktivasi

Fungsi transfer yang dipakai dalam metode belajar *backpropagation* harus memiliki sifat kontinu dan dapat diturunkan. Terdapat banyak fungsi transfer yang digunakan pemakaiannya tergantung dari aplikasi yang dirancang, yang penting fungsi transfer tersebut turunannya mudah dihitung untuk dapat mengimplementasikan algoritma *backpropagation* ini. Fungsi transfer yang biasa

dipakai oleh metode belajar *backpropagation* adalah fungsi *sigmod biner*. Sistem kerja fungsi aktivasi ditunjuka melalui gambar 2.10.



Gambar 2. 12. Fungsi Aktivasi

Fungsi aktivasi / transfer *sigmoid bipolar* $(-1 \ldots 1)$:

$$g(x) = \frac{1-e^{-x}}{1+e^{-x}} \quad \text{.....(8)}$$

2.2.10.1.9. Arsitektur Jaringan Syaraf Tiruan

Model jaringan syaraf tiruan yag telah dipelajari dan diimplementasikan banyak sekali ragamnya. Tiga arsitektur representatif adalah sebagai berikut :

a. *Associative Memory System*

Sistem ini mengkorelasi data *input* dengan informasi yang disimpan di dalam memori. Informasi ini dapat dipanggol kembali dari *input* yang tidak lengkap atau *input* yang terganggu. *Associative memory systems* ini dapat mendeteksi kesamaan (*similiarity*) antara *input* baru dan pola yang telah disimpan. Sebagian besar arsitektur jaringan syaraf dapat digunakan sebagai *associative memory*, dan contoh utamanya adalah jaringan Hopfiled.

b. *Multiple Layer Structure*

Associative Memory System dapat memiliki satu atau lebih lapisan tengah (*hidden layers*). Algoritma belajar yang paling lazim dipakai untuk arsitektur ini adalah *backpropagation*, dimana pendekatannya adalah dengan mengkoreksi jaringan ketika disesuaikan dengan data yang diberikan kepadanya. Tipe lain dari *supervised learning* adalah *competitive filter associative memory*. Tipe ini dapat belajar dengan cara mengubah bobotnya dalam pengenalan kategori data input, tanpa disediakan contoh oleh e*xternal trainer*. Contoh utama dari tipe ini adalah jaringan Kohonen.

c. *Double Layer Structure*

Struktur ini dicontohkan oleh pendekatan *Adaptive Resonance Theory*, tidak memerlukan pengetahuan tentang banyaknya kelas dalam *training data*, tapi menggunakan *feed forward* dan *feedback* untuk menyelesaikan parameter. Pada tipe ini data dianalisis untuk membangun banyaknya kategori yang dapat berubah ubah, yang merepresentasikan *input* data yang diberikan.

2.2.10.1.10. Proses Belajar

Belajar adalah suatu proses dimana parameter parameter bebas JST diadaptasikan melalui suatu proses perangsangan bekelanjutan oleh lingkungan dimana jaringan itu berada. Jenis belajar ditentukan oleh pola dimana pengubahan parameter dilakukan. Sehingga dalam proses belajar terdapat kejadian kejadian sebagai berikut :

a. JST dirangsang oleh lingkungan

b. JST mengubah dirinya sebagai hasil rangsangan ini.

c. JST memberikan respon dengan cara yang baru kepada lingkungan, disebabkan perubahan yang terjadi dalam struktur internalnya sendiri.

2.2.10.1.11. Paradigma Belajar

Pertimbangan yang penting dalam JST adalah penggunaan algoritma yang sesuai untuk belajar. JST telah didesain untuk tipe belajar yang berbeda. Tipe tipe belajar itu adalah :

a. *Heteroassociation*

Memetakan satu set data ke data yang lainnya. Tipe ini akan menghasilkan *output* yang pada umumnya berbeda dengan pola *input*nya. Tipe ini digunakan, sebagi contoh, dalam prediksi *stock market*.

b. *Autoassociation*

Menyimpan pola untuk toleransi *error*. Tipe ini menghasilkan pola *output* yang mirip atau tepat sama dengan pola *input*nya. Contoh penggunaan tipe ini adalah dalam sistem pengenalan pola secara optik.

c. *Regularly detection*

Mencari fitur yang berguna dalam data (*feature extraction*). Tipe ini digunakan dalam identifikasi sinyal sonar.

d. *Reinforce learning*

Beraksi terhadap *feedback*. Tipe ini digunakan pada pengendali dalam pesawat ruang angkasa.

Ada dua pendekatan dasar belajar dalam JST, yaitu *supervised learning* dan *unsupervised learning*.

2.2.10.1.12. Supervised Learning (Belajar Dengan Pengawasan)

Dalam pendekatan *supervised learning* kita menggunakan sekumpulan *input* dimana *output* yang sesuai telah diketahui. Perbedaan antar *output* aktual dan *output* yang diinginkan digunakan untuk mengkalkulasi koreksi pada bobot jaringan syaraf (disebut *learning with teacher*).

2.2.10.1.13. *Unsupervised Learning* (Belajar Tanpa Pengawasan)

Dalam *unsupervised learning*. JST mengorganisasikan dirinya untuk menghasilkan kategori dimana kumpulan *input* akan termasuk ke dalamnya. Tidak ada pegetahuan tentang apakah klasifikasi yang dibuat itu benar atau tidak, dan darimana jaringan berasal bisa berarti atau tidak untuk orang yang menggunakan jaringan. Tetapi, banyaknya kategori dapat dikontrol dengan cara mengubah ubah parameter tertentu dalam model. Dalam kasus *unsupervised learning* ini, harus ada pemeriksaan manusia dalam kategori final untuk memberikan arti dan menentukan kegunaan dari *output* yang dihasilkan.

2.2.10.1.14. JST propagasi balik (Backpropagation Neural Network)

Metode *backpropagation* sering juga disebut dengan *generalized delta rule*. *Backpropagation* adalah metode turunan gradien (*gradient descent method*) untuk meminimalkan *total square error* dari *output* yang dikeluarkan oleh jaringan.

JST propagasi balik merupakan algoritma pelatihan terbimbing yang mempunyai banyak lapisan. JST propagasi balik 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. Syarat fungsi aktivasi dalam JST propagasi balik adalah bersifat kontinu, terdifferensial dengan mudah, dan merupakan fungsi yang tidak turun. Fungsi aktivasi yang dapat memenuhi ketiga syarat tersebut adalah *logsig*, *tansig*, dan *purelin.*

Metode pengenalan merupakan proses inisialisasi data yang akan diolah selanjutnya oleh JST propagasi balik. Data yang akan dikenali disajikan dalam bentuk vektor. Masing-masing data mempunyai target yang disajikan juga dalam bentuk vektor. Target atau keluaran acuan merupakan suatu peta karakter yang menunjukkan lokasi dari vektor masukan. Sedangkan metode pelatihan merupakan proses latihan mengenali data dan menyimpan pengetahuan atau informasi yang didapat ke dalam bobot-bobot (Sierra-Canto, et al., 2010).

Terdapat 3 fase dalam pelatihan (*training*) JST propagasi balik, yaitu fase maju (*feed forward*), fase mundur (*back propagation*), dan fase modifikasi bobot. Dalam fase *feed forward*, pola masukan dihitung maju dimulai dari lapisan input hingga lapisan *output*. Dalam fase *back propagation*, tiap-tiap unit *output* menerima target pola yang berhubungan dengan pola input untuk dihitung nilai kesalahan. Kesalahan tersebut akan dipropagasikan mundur. Sedangkan fase modifikasi bobot bertujuan untuk menurunkan kesalahan yang terjadi. Ketiga fase tersebut diulang secara terus menerus hingga kondisi penghentian dipenuhi. (Nurmila, et al., 2010)