DISERTASI – TE143597

ANALISA CITRA WAJAH BAYI UNTUK DETEKSI NYERI DAN TANGIS MENGGUNAKAN MULTI STAGE CLASSIFICATION DAN DEEP LEARNING

YOSI KRISTIAN
07111360010008

DOSEN PEMBIMBING
Prof. Dr. Ir. Mauridhi Hery Purnomo, M.Eng.
Dr. I Ketut Eddy Purnama, ST., MT.

PROGRAM DOKTOR
DEPARTEMEN TEKNIK ELEKTRO
FAKULTAS TEKNOLOGI ELEKTRO
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2018

DISERTASI – TE143597

ANALISA CITRA WAJAH BAYI UNTUK DETEKSI NYERI DAN TANGIS MENGGUNAKAN MULTI STAGE CLASSIFICATION DAN DEEP LEARNING

YOSI KRISTIAN
07111360010008

DOSEN PEMBIMBING
Prof. Dr. Ir. Mauridhi Hery Purnomo, M.Eng.
Dr. I Ketut Eddy Purnama, ST., MT.

PROGRAM DOKTOR
DEPARTEMEN TEKNIK ELEKTRO
FAKULTAS TEKNOLOGI ELEKTRO
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2018

LEMBAR PENGESAHAN

Disertasi disusun untuk memenuhi salah satu syarat memperoleh gelar
Doktor (Dr.)
di
Institut Teknologi Sepuluh Nopember

oleh:

Yosi Kristian
NRP. 07111360010008

Tanggal Ujian : 28 Agustus 2018
Periode Wisuda : September 2018

Disetujui oleh:

1. Prof. Dr. Ir. Mauridhi Hery Purnomo, M.Eng. (Pembimbing I)
   NIP: 19580916 1986 01 1001

2. Dr. I Ketut Eddy Purnama, ST., MT. (Pembimbing II)
   NIP: 19580916 1986 01 1001

3. Prof. Sri Hartati, M.Sc, Ph.D (Penguji)
   NIP: 19610921 198803 2001

4. Dr. Adhi Dharma Wibawa, ST., MT (Penguji)
   NIP: 19760505 200812 1003

5. Dr. Eko Mulyanto Yuniarno, ST., MT. (Penguji)
   NIP: 19680601 199512 1009

Dekan Fakultas Teknologi Elektro

Dr. Tri Arief Sardjono, S.T., M.T.
NIP. 197002121995121001

Halaman ini sengaja dikosongkan

PERNYATAAN KEASLIAN TESIS

Dengan ini saya menyatakan bahwa isi keseluruhan Tesis saya dengan judul "**ANALISA CITRA WAJAH BAYI UNTUK DETEKSI NYERI DAN TANGIS MENGGUNAKAN MULTI STAGE CLASSIFICATION DAN DEEP LEARNING**" adalah benar-benar hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diijinkan dan bukan merupakan karya pihak lain yang saya akui sebagai karya sendiri.

Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap pada daftar pustaka. Apabila ternyata pernyataan ini tidak benar, saya bersedia menerima sanksi sesuai peraturan yang berlaku.

Surabaya, 9 Agustus 2018

Yosi Kristian

NRP. 07111360010008

Halaman ini sengaja dikosongkan

ANALISA CITRA WAJAH BAYI UNTUK DETEKSI NYERI DAN TANGIS MENGGUNAKAN *MULTI STAGE CLASSIFICATION* DAN *DEEP LEARNING*

Nama mahasiswa : Yosi Kristian
NRP : 07111360010008
Pembimbing : 1. Prof. Dr. Ir. Mauridhi Hery Purnomo, M.Eng.
2. Dr. I Ketut Eddy Purnama, ST., MT.

ABSTRAK

Bayi belum dapat menginformasikan rasa nyeri yang mereka alami, karena itu bayi menangis saat mereka mengalami nyeri. Dengan semakin berkembangnya teknologi visi komputer, beberapa tahun terakhir muncul beberapa penelitian yang mencoba mengenali nyeri pada tangis bayi memanfaatkan *machine learning* dan pengolahan citra.

Dalam disertasi ini diteliti tentang analisa pemanfaatan *handcrafted feature* dan *feature learning* dengan bantuan berbagai *machine learning* untuk mengatasi masalah klasifikasi nyeri pada wajah bayi. Pada penelitian menggunakan *handcrafted feature* dicari kombinasi fitur geometri dan tekstur yang paling optimal untuk dapat mengklasifikasikan nyeri dengan bantuan *multi stage classification* menggunakan SVM. Sedangkan pada penelitian dengan *feature learning* digunakan arsitektur *deep learning* dengan tipe *Deep Convolution Neural Network* (DCNN) *Autoencoder* dan *Long-Short Term Memory* (LSTM) *Network*.

Dari penelitian yang dilakukan performa *handcrafted feature* ditambah dua tahap klasifikasi cukup seimbang jika dibandingkan dengan performa *deep learning*. Keunggulan penelitian *deep learning* dengan memanfaatkan *autoencoder* dan LSTM adalah kemampuan mengolah video atau *sequence gambar*.

Kata kunci: Deteksi tangis bayi, klasifikasi nyeri pada bayi, fitur geometri, fitur tekstur, *feature learning*, *deep learning*

Halaman ini sengaja dikosongkan

ANALYSIS INFANT FACIAL IMAGE FOR PAIN AND CRY DETECTION USING MULTI STAGE CLASSIFICATION AND DEEP LEARNING

By : Yosi Kristian
Student Identity Number : 07111360010008
Supervisor(s) : 1. Prof. Dr. Ir. Mauridhi Hery Purnomo, M.Eng.
2. Dr. I Ketut Eddy Purnama, ST., MT.

ABSTRACT

Babies are yet able to inform the pain they are experiencing, so babies cry when they experience pain. With the development of computer vision technology, the last few years have emerged several studies that try to recognize the pain in babies faces using machine learning and image processing.

In this dissertation we try to analyze the use of handcrafted features and feature learning with the help of machine learning to tackle the infant facial pain classification problem. In the research using handcrafted features we found the most optimal combination for geometrical and textural features to classify pain in infant with the help of multi stage classification using SVM. While in the research with feature learning we use deep learning architecture with Deep Convolution Neural Network (DCNN) Autoencoder and Long-Short Term Memory (LSTM) Network.

From the research conducted, the performance of handcrafted feature plus two stages of classification is quite balanced when compared with the performance of the one using deep learning. The superiority of our deep learning research is the ability to process video or picture sequences because the usage of LSTM.

Key words: Infant facial cry detection, Infant facial pain classification, geometrical feature, textural feature, feature learning, deep learning.

Halaman ini sengaja dikosongkan

KATA PENGANTAR

Puji syukur kami panjatkan kepada Tuhan yang Maha Kuasa karena berkat rahmat dan bimbingan-Nya akhirnya kami dapat menyelesaikan disertasi program doktor teknik elektro ini. Kami menyadari sepenuhnya bahwa selama penyusunan disertasi ini banyak pihak yang telah memberikan bantuan, sehingga pada kesempatan ini kami ingin menyampaikan terima kasih yang tulus kepada:

1. Bapak Prof. Dr. Ir. Mauridhi Hery Purnomo, M.Eng. dan Bapak Dr. I Ketut Eddy Purnama, ST., MT. selaku Promotor dan Co-Promotor yang telah memberikan bimbingan, saran, semangat, bantuan dan dengan penuh kesabaran terus mendorong agar penulis dapat menyelesaikan studi S3 di Teknik Elektro ITS.
2. Bapak Dr. Adhi Dharma Wibawa, ST., MT., Bapak Dr. Eko Mulyanto Yuniarno, ST., MT., Ibu, Prof. Sri Hartati, M.Sc, Ph.D, dan Dr. Surya Sumpeno, ST., M.Sc., sebagai penguji disertasi dan proposal disertasi yang telah memberikan banyak masukan, kritik dan saran dalam penyelesaian disertasi ini.
3. Prof. Hideya Takahashi, dan Dr. Kayo Yoshimoto yang telah memberikan kesempatan dan masukan dalam penyusunan jurnal saat mengikuti program PKPI di Osaka City University.
4. Bapak Dr. Eng. Ardyono Priyadi, ST., M.Eng dan Bapak Dr.Ir. Djoko Purwanto M.Eng, Bapak Dr.Ir. Wirawan DEA, Bapak Dr. Ronny Seto Wibowo, ST, MT. sebagai pejabat Teknik Elektro ITS yang telah membantu kelancaran studi di ITS.
5. Bapak Arya Tandy Hermawan sebagai ketua STTS yang selalu memberi semangat dan mensupport disetiap kesempatan.
6. Bapak Gunawan, Bapak Herman Budianto dan Bapak FX. Ferdinandus sebagai pembantu ketua di STTS yang telah banyak direpoti dalam berbagai urusan dalam penyelesaian disertasi ini.

7. Ibu Endang Setyati yang telah merekomendasikan penulis untuk mengambil S3 di ITS serta terus memberi semangat untuk dapat menyelesaikan disertasi ini.
8. Bapak Reddy Alexandro dan Ibu Esther Irawati Setiawan yang telah banyak membantu penyelesaian disertasi.
9. Bapak Edwin Pramana, Bapak Lukman Zaman, Bapak Andreas dan Pak Joan Santoso yang terus memberi semangat dalam penyelesaian disertasi ini.
10. Effendy Hadi Sutanto dan Stefanie Tanujaya yang telah banyak membantu dalam penyusunan dataset dan hal-hal teknis.
11. Pak Hartono, Bu Ranny dan Pak Sapari sebagai karyawan Pps Teknik Elektro yang telah membantu kelancaran proses administrasi.
12. Rekan-rekan di STTS yang tidak dapat disebutkan satu persatu yang selalu memberikan bantuan dan semangat.
13. Dokter Elizeus Hanindito yang telah banyak membantu, dan menyediakan data.
14. Papa (Alm.), Mama, dan Adik yang selalu mensupport dalam penyelesaian disertasi ini.
15. Istri tercinta, Maria Bernadette Ria Ardhani, dan putra tersayang, Nick Frederic Kristian yang setia mendampingi, mendukung, dan menyemangati dengan penuh kesabaran, terutama ketika penulis harus mengikuti joint research ke Osaka.

Akhirnya penulis sadar bahwa karya ini masih jauh dari sempurna, oleh karena itu kritik dan saran yang membangun selalu kami harapkan.

Surabaya, 28 Agustus 2018
Penulis

DAFTAR ISI

LEMBAR PENGESAHAN v
PERNYATAAN KEASLIAN TESIS vii
ABSTRAK ix
ABSTRACT xi
KATA PENGANTAR xiii
DAFTAR ISI xv
DAFTAR GAMBAR xix
DAFTAR TABEL xxii
BAB 1 PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Rumusan Masalah 2
1.3 Tujuan 3
1.4 Posisi, Tahapan dan *Roadmap* Penelitian 3
1.5 Kontribusi dan Orisinalitas Penelitian 6
1.6 Penelitian Sebelumnya 6
1.7 Sistematika Pembahasan 12
BAB 2 KAJIAN PUSTAKA 15
2.1 Penunjang Praproses 15
2.1.1 Face Detection 15
2.1.2 Landmark detection menggunakan Active Shape Model 21
2.2 Ekstraksi Fitur 27
2.2.1 Fitur Geometri 28
2.2.2 Fitur Tekstur 30
2.3 Machine Learning 31
2.3.1 Adachi Chaotic Neural Network 31
2.3.2 Convolutional Neural Network (ConvNet) 34
2.3.3 Recurrent Neural Network (RNN) 43
2.3.4 Long Short Term Memory Network (LSTM) 45
2.3.5 Support Vector Machine 51

BAB 3 DETEKSI TANGIS PADA WAJAH BAYI MEMANFAATKAN FITUR GEOMETRI DAN ADACHI *CHAOTIC NEURAL NETWORK* 55
3.1 Praproses dan Ekstraksi Fitur 56
3.2 *Chaotic NN Classifier* 58
3.3 Hasil Penelitian 59
3.4 *Concluding Remarks* 59
BAB 4 KLASIFIKASI NYERI PADA WAJAH BAYI MEMANFAATKAN FITUR TEKSTUR DAN FITUR GEOMETRI DAN SKENARIO KLASIFIKASI DUA TAHAP 61
4.1 Teori Nyeri 61
4.2 Dataset 62
4.3 Arsitektur Sistem 64
4.4 Ekstraksi Fitur Geometri 64
4.5 Ekstraksi Fitur Tekstur 65
4.6 Klasifikasi dengan SVM dan Variasi Kernelnya 66
4.7 *Metric* dan Skenario Ujicoba 67
4.8 Ujicoba Deteksi Tangis 68
4.9 Ujicoba Klasifikasi Nyeri 68
4.10 *Concluding Remarks* 70
BAB 5 PEMANFAATAN DCNN AUTOENCODER DAN LSTM UNTUK KLASIFIKASI TANGIS DAN NYERI PADA VIDEO WAJAH BAYI 73
5.1 Dataset 73
5.2 Convolutional Neural Network 74
5.3 *Convolutional Autoencoder* 76
5.4 Long Short Term Memory Network 77
5.5 Klasifikasi Nyeri dan Tangis Pada Wajah Bayi 79
5.5.1 Tahap Praproses 80
5.5.2 Tahap Convolution Autoencoder 81
5.5.3 Tahap Klasifikasi Menggunakan LSTM 83
5.6 Ujicoba Sistem 85
5.7 *Concluding Remarks* 93
BAB 6 PENUTUP 95

6.1 Kesimpulan Hasil Penelitian 95
6.2 Usulan Penelitian Selanjutnya 96
DAFTAR PUSTAKA 99
LAMPIRAN A 105
BIODATA PENULIS 107

Halaman ini sengaja dikosongkan

DAFTAR GAMBAR

Gambar 1.1	Gambar bayi menangis a. Bayi menangis karena alasan lain, b. Bayi menangis karena kesakitan	2
Gambar 1.2	Diagram *Fishbone* Penelitian	4
Gambar 1.3	Posisi Penelitian	5
Gambar 1.4	Bagan Kontribusi dan Orisinalitas Penelitian	6
Gambar 2.1	Contoh Haar-like Features	16
Gambar 2.2	Contoh Hasil Perhitungan Integral *Image*	17
Gambar 2.3	Contoh Integral Image Sebagian	17
Gambar 2.4	Proses Klasifikasi dengan Cascade Classifier	20
Gambar 2.5	Data training dengan titik keterangannya	22
Gambar 2.6	Pengukuran Akurasi Model	26
Gambar 2.7	Contoh Proses Fitting ASM	26
Gambar 2.8	Rujukan titik-titik penting wajah (Facial Landmark)	27
Gambar 2.9	Jarak antara titik titik acuan pada wajah bayi yang digunakan untuk menentukan nyeri	29
Gambar 2.10	Ilustrasi Tetangga (N) dan Radius (R) pada LBP	30
Gambar 2.11	Contoh perhitungan LBP	31
Gambar 2.12	Ilustrasi Convolutional Neural Networks	35
Gambar 2.13	Susunan Neuron pada Layer Input	36
Gambar 2.14	Local Receptive Field atau Kernel Berukuran 5x5	37
Gambar 2.15	Kernel Setelah Satu Kali *Slide*	37
Gambar 2.16	Ilustrasi Zero-padding	39
Gambar 2.17	Convolutional Layer	40
Gambar 2.18	Ilustrasi Maxpooling Layer	42
Gambar 2.19	Recurrent Neural Network	44
Gambar 2.20	Beberapa Jenis Recurrent Neural Network	46
Gambar 2.21	Notasi Penjelas Unit	46
Gambar 2.22	RNN Unit	46
Gambar 2.23	LSTM Unit	47

Gambar 2.24 LSTM Cell State 47
Gambar 2.25 LSTM Gate 47
Gambar 2.26 Forget Gate 49
Gambar 2.27 Input Gate dan Candidate Gate 49
Gambar 2.28 Update Cell State 49
Gambar 2.29 Output Gate 50
Gambar 2.30 Margin pada SVM 51
Gambar 2.31 Paralel Hyperplane pada SVM 52
Gambar 3.1 Blok diagram Deteksi Tangis dengan Adachi Chaotic NN 55
Gambar 3.2 Contoh deteksi wajah bayi dengan metode Viola Jones 56
Gambar 3.3 Pemberian Landmark Manual Pada Wajah Bayi 57
Gambar 3.4 Proses Fitting ASM pada wajah baru 57
Gambar 3.5 Fitur Geometri untuk Deteksi Tangis Bayi 58
Gambar 4.1 Contoh AU pada wajah yang mengalami nyeri. 62
Gambar 4.2 a. Proses seleksi frame. b. Contoh citra dalam dataset. 63
Gambar 4.3 Arsitektur Sistem 64
Gambar 4.4 Garis-garis putih pada baris pertama menunjukkan fitur NPPN, dan pada baris kedua menunjukkan fitur ENPPD. 65
Gambar 4.5 Proses TBLBP81 66
Gambar 4.6 Proses MPLBP81 67
Gambar 4.7 Visualisasi Nilai Area Under ROC Curve Untuk Deteksi Tangis Memanfaatkan Variasi Fitur dan Kernel SVM 69
Gambar 4.8 Grafik Nilai F1 Score Untuk Deteksi Tangis Memanfaatkan Variasi Fitur dan Kernel SVM Visualisasi 69
Gambar 4.9 Contoh Ujicoba dari input, intermediate result, sampai hasil saat ujicoba 70
Gambar 5.1 Proses pembentukan dataset dari video. 74
Gambar 5.2 Contoh Arsitektur Convolutional Autoencoder 77
Gambar 5.3 Ilustrasi LSTM *Unit* 79
Gambar 5.4 Tahap *Preprocessing* 80
Gambar 5.5 Ilustrasi Rotasi Berdasarkan Posisi Mata 81
Gambar 5.6 Grafik ReLU dibandingkan dengan Leaky ReLU 82

Gambar 5.7 Performa Autoencoder. (a) Citra Input, (b) Hasil Autoencoder dengan Loss MSE, dan (c) Hasil Autoencoder dengan Loss SSIM. 83
Gambar 5.8 Ilustrasi Sistem Klasifikasi Tangis dan Nyeri dari Video Wajah Bayi 84
Gambar 5.9 Contoh Potongan Video Output dari Sistem 84
Gambar 5.10 Grafik Loss pada ujicoba CAE skenario pertama 86
Gambar 5.11 Citra Input (a) Beserta Citra Hasil Rekonstruksi Dari Model Skenario Pertama (b) 86
Gambar 5.12 Grafik Hasil Loss Pada CAE Skenario Kedua 87
Gambar 5.13 Citra Input (a) Beserta Citra Hasil Rekonstruksi Dari Model Skenario Kedua (b) 87
Gambar 5.14 Arsitektur Convolutional Autoencoder Pada Skenario Ketiga 88
Gambar 5.15 Grafik Hasil Loss Skenario Ketiga 88
Gambar 5.16 Citra Input (a) Beserta Citra Hasil Rekonstruksi Dari Model Skenario Ketiga (b) 88
Gambar 5.17 Grafik Hasil Loss Skenario Keempat 89
Gambar 5.18 Citra Input (a) Beserta Citra Hasil Rekonstruksi Dari Model Skenario Keempat (b) 89
Gambar 5.19 Visualisasi Hasil Uji Coba LSTM 91
Gambar 5.20 Visualisasi Hasil Uji Coba Dropout Pada LSTM 92
Gambar 6.1 Dua representasi citra wajah, (kiri) Matrix Pixel RGB, (kanan) titik-titik landmark wajah. 96
Gambar 6.2 Dua representasi citra wajah bayi diumpankan ke dalam DCNN dengan target derajat nyeri yang dialami. 98

Halaman ini sengaja dikosongkan

DAFTAR TABEL

Tabel 1.1	Penelitian Sebelumnya Tentang Wajah dan Suara Bayi	7
Tabel 1.2	Penelitian Sebelumnya Yang Menunjang Disertasi	9
Tabel 2.1	Fitur Geometri Penelitian Terdahulu	28
Tabel 3.1	Matriks confusion untuk Ideal-M-AdNN Classifier	59
Tabel 3.2	Perbandingan Akurasi dengan Classifier Lain	59
Tabel 4.1	Persebaran kelas dalam dataset.	63
Tabel 4.2	Hasil Deteksi Tangis	68
Tabel 4.3	Hasil Deteksi Nyeri untuk Satu Tahap	68
Tabel 4.4	Hasil Deteksi Nyeri untuk Dua Tahap	70
Tabel 5.1	Hasil Uji Coba Autoencoder	85
Tabel 5.2	Uji Coba Long Short Term Memory	91
Tabel 5.3	Uji Coba Dropout Pada Long Short Term Memory Network	91
Tabel 5.4	Confusion Matrix dari Deteksi Tangis	91
Tabel 5.5	Confusion Matrix dari Klasifikasi Nyeri	92
Tabel 5.6	Perbandingan Dengan Penelitian Sebelumnya	92
Tabel 6.1	Dua macam arsitektur DCNN yang akan dicoba.	96

Halaman ini sengaja dikosongkan

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Bayi sering menangis untuk berkomunikasi dengan orang lain karena mereka belum bisa mengekspresikan kebutuhan dan keinginannya melalui bahasa. Walaupun mereka bisa memberikan tanda lainnya seperti menendang kaki, melambaikan tangan, menggerakkan kepala, tapi cara paling efektif bagi bayi untuk mendapatkan perhatian adalah melalui tangisan. Ada berbagai makna tangisan bayi, bayi akan menangis jika merasa tidak nyaman seperti lapar, popok basah, terlalu panas atau dingin, pakaian yang terlalu ketat, kesakitan, dan masih banyak sebab lainnya.

Tangisan bayi memiliki banyak makna, namun yang paling berbahaya adalah tangisan karena kesakitan (*pain*), terutama setelah bayi menjalani operasi. Pada ruang pemulihan bayi di rumah sakit jumlah suster juga sangat terbatas sehingga untuk mengawasi masing-masing bayi secara kontinu sangat tidak mungkin. Manusia dewasa dapat menunjukkan atau memperlihatkan rasa sakitnya dengan berbagai macam cara [1]. Salah satu cara yang paling sering digunakan untuk mengukur seberapa sakit pasien adalah dengan laporan pasien itu sendiri (*self reporting*) sehingga tidak diperlukan keahlian khusus maupun teknologi lanjut untuk mengetahui seberapa sakit yang dirasakan oleh pasien tersebut [2].

Pada penelitian-penelitian sebelumnya sudah banyak diteliti tentang makna tangis bayi melalui suara [3]. Kondisi rumah sakit yang menempatkan beberapa bayi dalam satu ruang akan sangat sulit untuk mengisolasi suara tangis bayi, saat ada beberapa bayi yang menangis. Kondisi lain yang juga harus diperhitungkan adalah bagaimana menangani bayi yang memiliki kelainan pita suara (bisu) sehingga saat menangis tidak mengeluarkan suara. Karena itu, kita mengandalkan ekspresi wajah mereka sebagai salah satu sumber informasi. Dibandingkan dengan *response* secara tingkah laku maupun psikis, ekspresi wajah untuk rasa sakit lebih spesifik dan konsisten dalam menunjukkan rasa sakit [4]. Inilah mengapa pada beberapa instrumen untuk mengukur nyeri pada bayi seperti

(a) (b)

Gambar 1.1 Gambar bayi menangis a. Bayi menangis karena alasan lain, b. Bayi menangis karena kesakitan

CRIES [5], FLACC (*Face, Legs, Activity, Cry, and Consolability*) [4], dan MIPS (*Modified Infant Pain Scale*) [6] lebih fokus pada ekspresi wajah bayi.

Pada banyak rumah sakit di Indonesia, observasi dan penilaian FLACC merupakan metode pengukuran yang sangat umum digunakan untuk mengukur rasa sakit pada bayi [7]. Meskipun begitu, dengan banyaknya jumlah bayi dan terbatasnya jumlah dokter dan perawat pada rumah sakit, maka sangat sulit bagi mereka untuk menjaga dan mengawasi para bayi secara terus menerus. Inilah mengapa, dibutuhkan sebuah sistem cerdas yang dapat mendeteksi rasa sakit pada bayi secara otomatis dan memiliki tingkat akurasi yang tinggi.

Dalam disertasi ini diteliti tentang bagaimana sebuah sistem dapat mendeteksi bayi yang menangis hanya melalui citra wajahnya, kemudian sistem akan mengklasifikasikan mana bayi yang menangis karena kesakitan atau karena alasan yang lain. Pada gambar 1.1 tampak contoh bayi yang menangis karena kesakitan dan karena alasan lain.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dijelaskan di atas, Gap penelitian yang ada adalah belum adanya penelitian tentang *pain detection* yang berbasis pada wajah bayi yang menangis.

Sedangkan permasalahan yang dihadapi dalam penyelesaian penelitian ini adalah sebagai berikut:

- Diperlukan sebuah sistem deteksi wajah bayi yang memiliki akurasi baik.
- Diperlukan model wajah yang dapat melakukan ekstraksi komponen-komponen wajah bayi, baik yang sedang menangis atau tidak.
- Diperlukan sebuah *classifier* untuk mendeteksi apakah wajah bayi sedang menangis atau tidak.
- Dibutuhkan *classifier* untuk mendeteksi apakah bayi sedang menangis karena kesakitan atau karena hal lain.
- Menerapkan sistem ini untuk video *stream*.

1.3 Tujuan

Tujuan yang akan dicapai dari penelitian ini adalah:

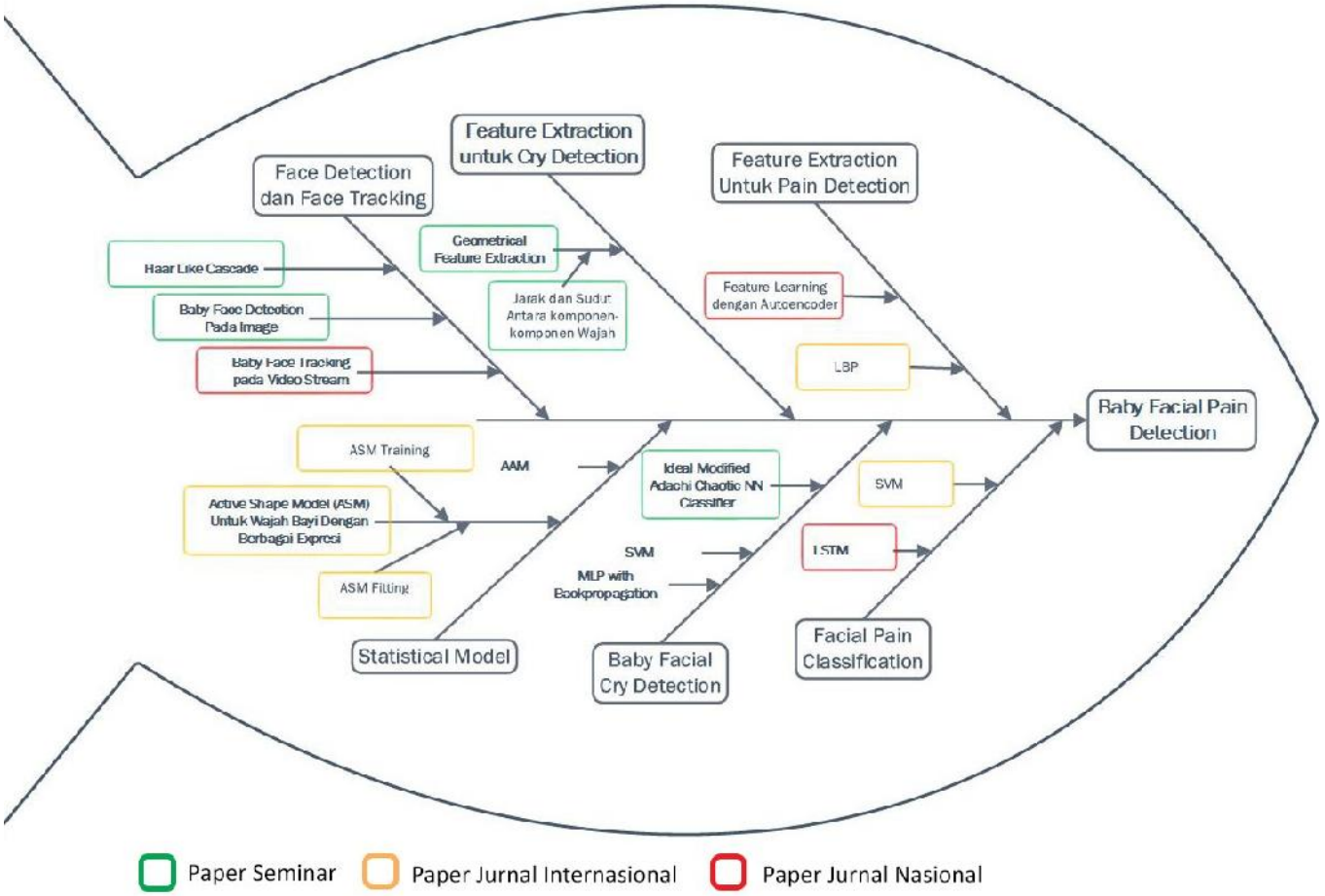
- Membangun sistem *classifier* untuk mendeteksi apakah bayi yang menangis disebabkan karena kesakitan bukan karena sebab lain.
- Untuk mencapai tujuan di atas maka perlu
    - Membangun sistem *face detection* untuk bayi.
    - Membangun model *facial landmarking* untuk bayi dengan berbagai ekspresi.
    - Membangun sistem untuk ekstraksi fitur wajah bayi.
    - Membangun sistem *classifier* untuk mengklasifikasikan bayi sedang menangis atau tidak dengan memanfaatkan *machine learning*.

Sedangkan manfaat yang diharapkan dari penelitian ini adalah:

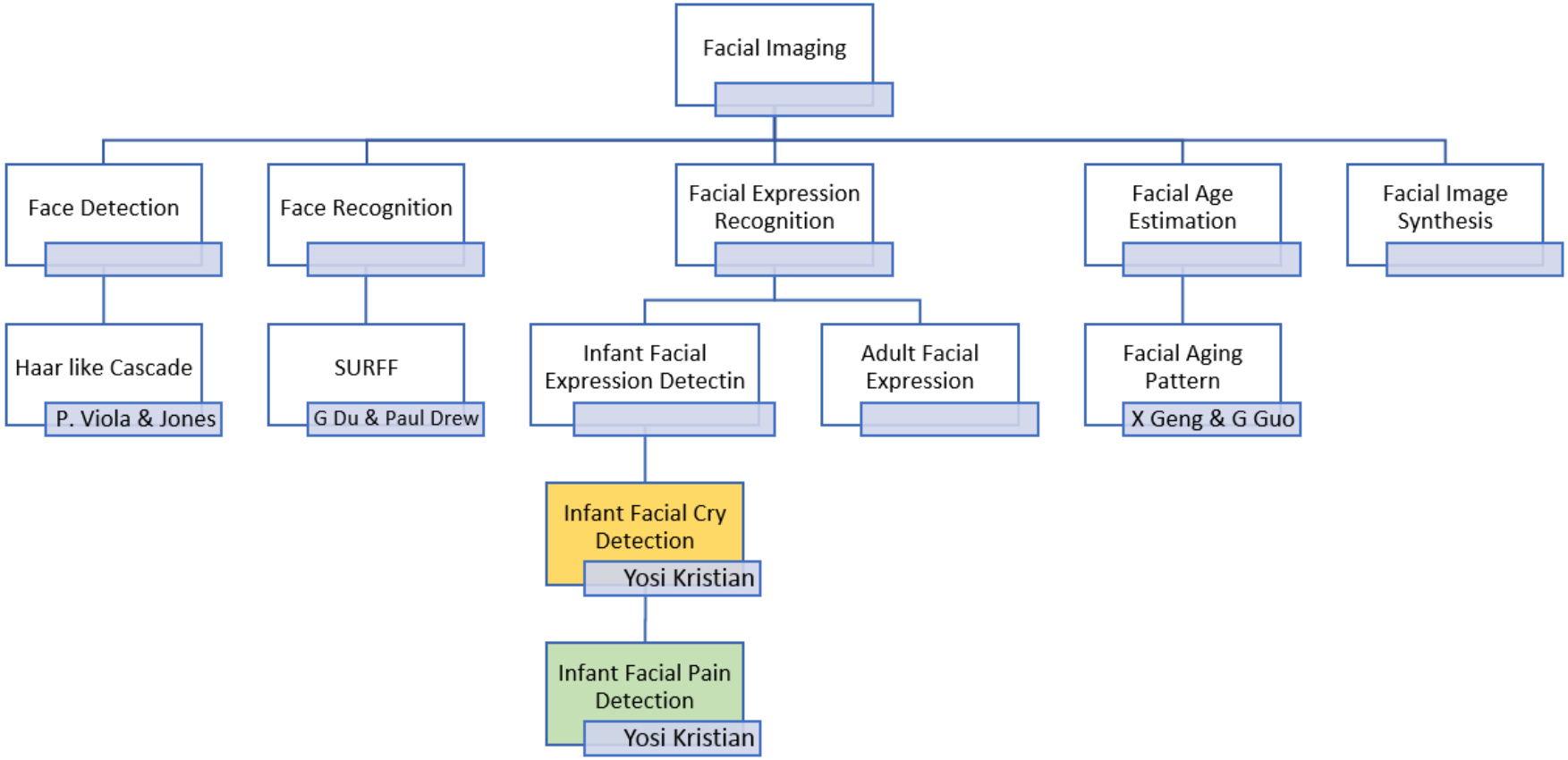
- Dihasilkan sebuah sistem yang menerima input citra/video wajah bayi dan dapat melakukan *cry detection* dan *pain detection*.

1.4 Posisi, Tahapan dan *Roadmap* Penelitian

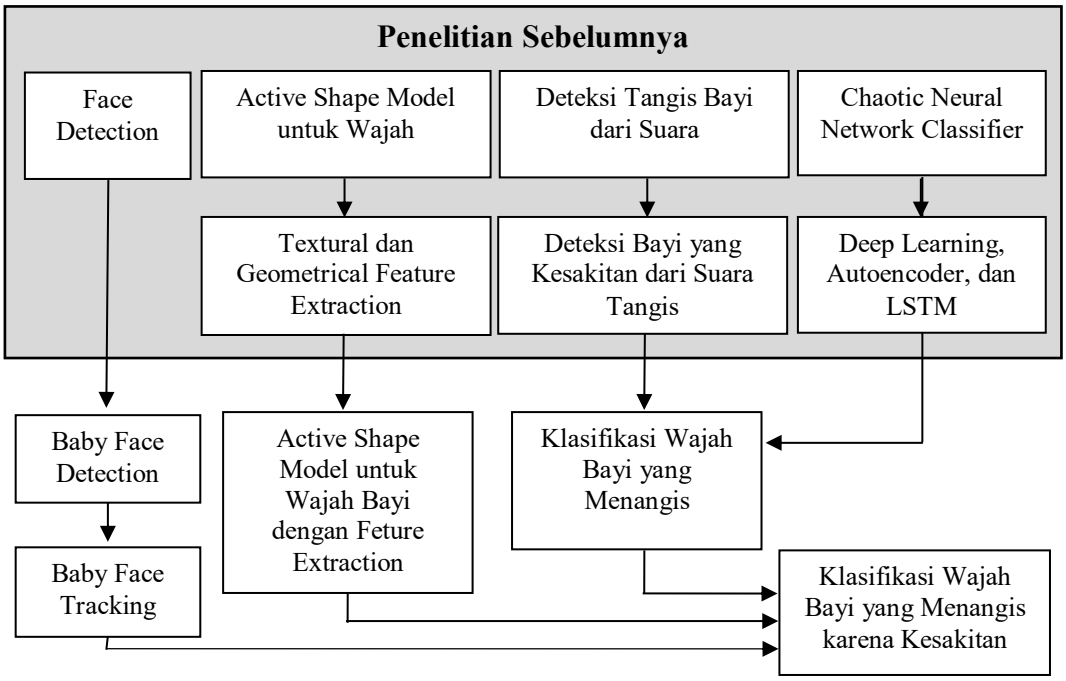
Tahapan penelitian dalam disertasi ini dibagi menjadi 3 tahap. Pada gambar 1.2 tampak tahapan dan *roadmap* penelitian dalam bentuk *fishbone* diagram untuk penyelesaian disertasi ini. Untuk posisi penelitian pada distertasi ini dapat dilihat pada gambar 1.3



Gambar 1.2 Diagram *Fishbone* Penelitian



Gambar 1.3 Posisi Penelitian



Gambar 1.4 Bagan Kontribusi dan Orisinalitas Penelitian

1.5 Kontribusi dan Orisinalitas Penelitian

Kontribusi dari hasil penelitian disertasi ini adalah dibangunnya sebuah sistem visi komputer yang dapat mengklasifikasikan nyeri yang dialami bayi berdasarkan citra ekspresi wajahnya. Sedangkan untuk Orisinalitas dapat dilihat pada Gambar 1.4

1.6 Penelitian Sebelumnya

Penelitian-penelitian sebelumnya terkait ekspresi wajah dan suara bayi dapat dilihat pada tabel 1.1. Sedangkan penelitian-penelitian sebelumnya yang menunjang disertasi ini tampak pada tabel 1.2.

Tabel 1.1 Penelitian Sebelumnya Tentang Wajah dan Suara Bayi

Tahun	Pengarang	Judul Penelitian	Publikasi	Wajah / Suara	Keterangan
2013	Elizeus Hanindito	Validasi Dinamika Pola Akustik Tangis Sebagai Indikator Nyeri Akut Pasca bedah pada Bayi [1]	Disertasi Doktor Fakultas Kedokteran Univ Airlangga Surabaya.	Suara	Penulis membangun sebuah dataset video bayi kemudian melakukan penilaian berdasarkan FLACC untuk menilai tinkat kesakitan bayi. Penulis juga memanfaatkan hormone kortisol untuk validasi tingkat nyeri yang dalami bayi. Penulis kemudian memanfaatkan dataset untuk mencoba mendeteksi nyeri dari suara tangis bayi
2010	Fang, Chiung-Yao, H. W. Lin, and Sei-Wang Chen	An Infant Facial Expression Recognition System based on Moment Feature Extraction [19]	Journal VISAPP	Wajah	Penulis mencoba mengenali expresi bayi seperti marah, sedih, lapar, takut dari wajah. Fitur yang dimanfaatkan adalah Zernike momen. Penulis juga memanfaatkan face detection namun menggunkan color component.
2008	Martin Schiavenato, Jacquie F. Byers, Paul Scovanner, James M. McMahon, Yinglin Xia, Naiji Lu, Hua He.	Neonatal pain facial expression: Evaluating the primal face of pain	Elsevier	Wajah	Pada penelitian ini penulis mengamati beberapa landmark point pada wajah kemudian mengukur perpindahannya saat bayi mengalami kesakitan. Peneliti juga meneliti perpindahan landmark point ini untuk berbagai ras dan gender kemudian menemukan bahwa rata rata perpindahan memang berbeda untuk ras dan gender yang berbeda.

Tahun	Pengarang	Judul Penelitian	Publikasi	Wajah / Suara	Keterangan
2012	Naufal Mansor, M. Jamil, S.H.S.M. Rejab, M.N. Jamil.	Early choke infant monitoring scheme	Prosiding IEEE-IMSNA	Wajah	Pada penelitian ini penulis mencoba mendeteksi saat bayi sedang tersedak dari fitur wajahnya. Penulis memanfaatkan PCA untuk meringkas dimensi wajah, dan SVM sebagai classifier.
2010	D. Wen, C. Fang, X. Ding, and T. Zhang	Development Of Recognition Engine For Baby Faces	Prosiding IEEE-ICPR 2010	Wajah	Pembuatan system rekognisi yang spesifik untuk wajah bayi. Penulis memanfaatkan gabor dan PCA. Penulis memanfaatkan 300 wajah bayi.
2012	M. Mansor, M. Rejab, S.-F. Jamil, A.-F. Jamil, A. Junoh, and J. Ahmad	Fast Infant Pain Detection Method	Prosiding IEEE-ICCCE 2012	Wajah	Penulis memanfaatkan Face detection dan PCA untuk pain detection, tanpa memanfaatkan ASM dan tanpa melalui cry detection, penulis juga hanya memanfaatkan still image.
2013	W.-L.Ou, S. M.Yu, J.-W.Chang, and C.-P.Fan	Video Based Vomit and Facial Foreign Object Detections for Baby Watch and Safety	Prosioding ICOT 2013	Wajah	Penulis meneliti tentang pembuatan system untuk mendeteksi benda asing atau muntahan yang ada pada wajah bayi. Sistem ini dilengkapi dengan eye dan mouth detection, penelitian ini menghasilkan akusrasi sampai dengan 88%

Tabel 1.2 Penelitian Sebelumnya Yang Menunjang Disertasi

Tahun	Pengarang	Judul Penelitian	Publikasi	Keterangan
2001	P. Viola and M. Jones	Rapid object detection using a boosted cascade of simple features	Proceedings of the 2001 IEEE Computer Society Conference	Dasar pembuatan algoritma Haar like cascade untuk Face Detection.
2004	P. Viola and M. Jones	Robust real-time face detection	International journal of computer vision, vol. 57, no. 2	Algoritma Face Detection yang paling banyak dipakai. Memanfaatkan Haar-like cascade, dengan Ada Boost dan Integral Image. Algoritma ini dipakai karena popular dengan kecepatannya. Dalam penelitian ini juga akan memanfaatkan algoritma karya Viola & Jones ini.
1995	T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham	Active shape models-their training and application	Computer vision and image understanding, vol. 61, no. 1	Tentang penelitian statistical model baru yang dikenal dengan nama Active Shape Model. ASM merupakan model yang akan dimanfaatkan dalam penelitian ini untuk mengekstraksi komponen komponen wajah.
1990	K. Aihara, T. Takabe, and M. Toyoda	Chaotic neural networks	Physics letters A, vol. 144, no. 6	Penemuan Chaotic Neural Network Pertama kali.
1997	M. Adachi and K. Aihara	Associative dynamics in a chaotic neural network	Neural Networks, vol. 10, no. 1	Penyempurnaan Chaotic Neural Network awal dan ditemukannya Adachi Neural Network (AdNN). Masih banyak kelemahan dan belum dimanfaatkan untuk pattern recognition.
2007	D. Calitoiu, B. J. Oommen, and D. Nussbaum	Periodicity and stability issues of a chaotic pattern recognition neural network	Pattern analysis and applications, vol. 10, no. 3	Modifikasi AdNN sehingga dapat dipakai untuk pattern recognition. Modifikasi ini dikenal dengan nama M-AdNN,

Tahun	Pengarang	Judul Penelitian	Publikasi	Keterangan
2012	K. Qin and B. Oommen	Ideal chaotic pattern recognition using the modified adachi neural network	Prosiding Chaotic Modeling and Simulation (CMSIM) 2012	Fine Tuning dari M-AdNN sehingga performa pattern recognitionnya lebih baik. Modifikasi ini dikenal dengan nama Ideal-M-AdNN.
1993	Ekman	Facial Expression and Emotion,	American Psychologist Vol.48, No. 4, April 1993.	Penelitian tentang kaitan emosi dengan ekspesi wajahnya. Komponen wajah dianalisa dan digunakan untuk membentuk action unit.
1998	John Platt	Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines	Advances in kernel methods 1998	Tentang algoritma training SVM yang diberi nama Squential Minimal Optimization atau SMO.
2015	Yosi Kristian, Mochamad Hariadi, Mauridhi Hery Purnomo	Ideal Modified Adachi Chaotic Neural Networks and active shape model for infant facial cry detection on still image	WCCI – IJCNN 2015 Beijing	Tentang klasifikasi tangis berdasarkan wajah bayi menggunakan fitur geometri dan Chaotic Neural Network.
2016	Yosi Kristian, Hideya Takahashi. I Ketut Eddy Purnama, Kayo Yoshimoto, Esther Irawati Setiawan, Elizeus Hanindito. Mauridhi Hery Purnomo.	A novel approach on infant facial pain classification using multi stage classifier and geometrical-textural features combination	IAENG IJCS Vol 44 Feb 2017	Tentang klasifikasi tangis dan nyeri pada wajah bayi menggunakan skenario dua tahap dan kombinasi fitur Geometri dan Tekstur. Dalam penelitian ini juga diteliti pengaruh variasi kernel SVM pada klasifikasi tangis dan nyeri.

Tahun	Pengarang	Judul Penelitian	Publikasi	Keterangan
2018	Yosi Kristian, I Ketut Eddy Purnama, Effendy Hadi Sutanto, Lukman Zaman, Esther Irawati Setiawan, Mauridhi Hery Purnomo.	Klasifikasi Nyeri Dan Tangis Pada Bayi Berdasarkan Video Expresi Wajah Menggunakan DCNN Autoencoder dan LSTM	JNTETI Vol. 7, No. 3, Bulan Agustus 2018	Tentang klasifikasi tangis dan nyeri pada video wajah bayi menggunakan deep learning. Digunakan DCNN Autoencoder untuk feature learning, dan LSTM untuk pemrosesan sequence frame.

1.7 Sistematika Pembahasan

Pembahasan inti disertasi ini dimulai dengan deteksi tangis pada wajah bayi memanfaatkan fitur geometri saja pada Bab 3. Kemudian dilanjutkan pada Bab 4 dengan pemanfaatan fitur geometri dan fitur tekstur untuk klasifikasi tangis dan nyeri pada bayi. Pada bab 5 dibahas tentang pemanfaatan *deep learning* untuk deteksi nyeri pada video wajah bayi. Penulisan disertasi ini terdiri atas 6 bab, dengan rincian tiap bab sebagai berikut:

Bab 1 Pendahuluan

Pada bagian ini peneliti menyajikan ikhtisar topik utama yang dibahas dengan perumusan masalah, tujuan dan manfaat, kontribusi dan kebaruan, serta posisi dan *roadmap* penelitian.

Bab 2 Tinjauan Pustaka

Bab ini menjelaskan beberapa teori yang mendukung *framework* penelitian. Bab ini terbagi dalam 3 bagian, pertama, terkait pra proses sebelum ekstraksi fitur. Kedua, menyajikan tentang ekstraksi fitur geometri dan fitur tekstur. Ketiga, *machine learning* dan *deep leaning*.

Bab 3 Deteksi Tangis Pada Wajah Bayi memanfaatkan Fitur Geometri dan Adachi *Chaotic NN*

Berisi tentang riset awal deteksi tangis pada wajah bayi yang memanfaatkan ekstraksi fitur geometri yang dibantu *active shape model* dan menggunakan Adachi *chaotic NN* sebagai *classifier*.

Bab 4 Klasifikasi Nyeri Pada Wajah Bayi memanfaatkan Fitur Tekstur dan Fitur Geometri dan skenario Klasifikasi dua Tahap

Bab ini berisi tiga bagian besar: Pertama tentang pemanfaatan klasifikasi dua tahap untuk deteksi nyeri pada wajah bayi. Bagian kedua berisi tentang kombinasi fitur geometri dan tekstur yang performanya cukup baik dalam deteksi nyeri pada bayi. Bagian ketiga berisi pengujian sistem yang dirancang.

Bab 5 Klasifikasi Nyeri Pada Video Wajah Bayi memanfaatkan DCNN Autoencoder dan LSTM Network

Bab ini berisi tiga bagian besar: Pertama tentang penjelasan dataset dan beberapa teori penunjang. Bagian kedua berisi tentang klasifikasi nyeri

pada video wajah bayi menggunakan autoencoder dan LSTM. Bagian ketiga berisi pengujian sistem yang dirancang.

Bab 6 Penutup

Bab ini adalah bab terakhir yang berisi 2 bagian. Pertama, kesimpulan hasil pengujian dan komparasi antara *handcrafted feature* dan *feature learning.* Kedua, penelitian selanjutnya yang direncanakan.

Halaman ini sengaja dikosongkan

BAB 2
KAJIAN PUSTAKA

Kajian pustaka ini dibagi menjadi tiga bagian: penunjang praproses, ekstraksi fitur, dan *machine learning*. Pada masing-masing bagian dijelaskan teknik-teknik yang dipakai untuk menunjang disertasi ini.

2.1 Penunjang Praproses

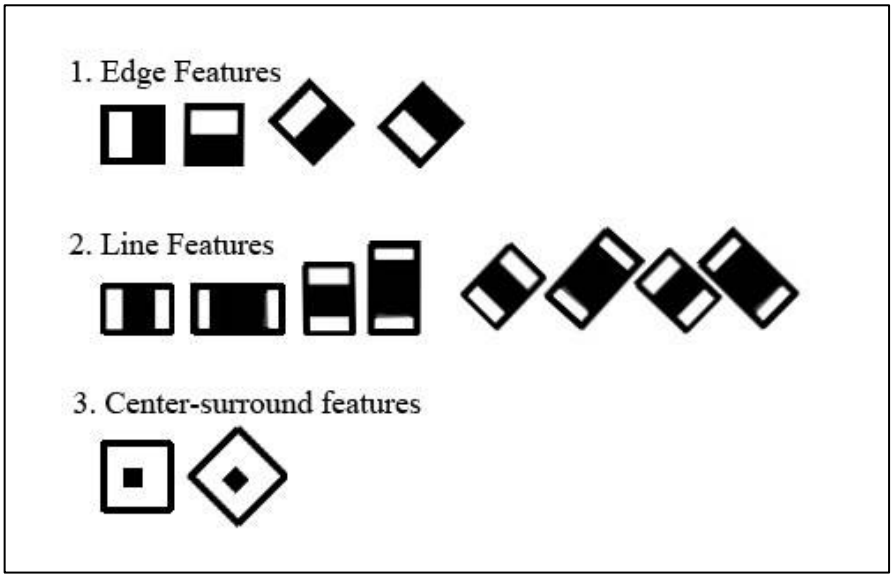
Pada subbab penunjang praproses ini dibahas tentang teknik *face detection,* dan *teknik face landmark detection*.

2.1.1 Face Detection

Masalah deteksi wajah atau *face detection* sebenarnya telah terpecahkan sejak adanya metode Viola-Jones yang memanfaatkan *Haar-like feature* dan *Adaboost* [8]. Metode ini terbukti cepat dan akurat, sehingga saat ini banyak dipakai dipakai di berbagai aplikasi.

Algoritma Viola-Jones digagas oleh Paul Viola dan Michael Jones pada tahun 2001. Algoritma ini merupakan metode untuk mendeteksi objek dalam gambar, tidak hanya wajah saja, namun metode ini banyak digunakan untuk mendeteksi wajah. Pada algoritma ini terdapat beberapa konsep utama yaitu *Haar-like feature*, *integral image* agar proses pendeteksian berjalan dengan cepat, metode *machine-learning AdaBoost*, dan *cascaded classifier* untuk mengkombinasikan banyak fitur dengan efektif dan efisien.

Fitur yang digunakan pada algoritma Viola-Jones merupakan fitur yang berbasis Haar *wavelets.* Haar *wavelets* adalah gelombang tunggal yang berbentuk segiempat yang hanya mempunyai interval tinggi dan interval rendah. Jika digambarkan dalam dua dimensi, sebuah gelombang segi empat digambarkan dengan sepasang segi empat yang berdekatan, satu berwarna terang dan satunya lagi berwarna gelap. Sesungguhnya kombinasi bidang segi empat yang digunakan untuk deteksi objek visual bukanlah Haar *wavelets* yang sebenarnya, namun lebih cocok untuk digunakan sebagai deteksi objek visual. Oleh karena itu, fitur-fitur ini



Gambar 2.1 Contoh Haar-like Features [8]

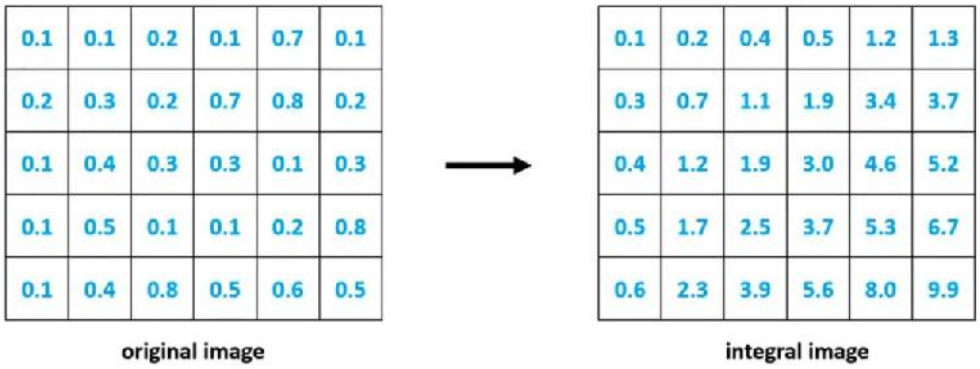
disebut *Haarlike features*, dan bukan Haar *wavelets*. Contoh fitur-fitur Haar tersebut dapat dilihat pada Gambar 2.1.

Nilai dari Haar-like features dihitung dengan cara mengurangkan nilai rata-rata piksel daerah gelap dengan nilai rata-rata piksel daerah terang. Jika perbedaan nilai berada di atas *threshold* yang ditentukan, maka fitur tersebut dianggap ada.

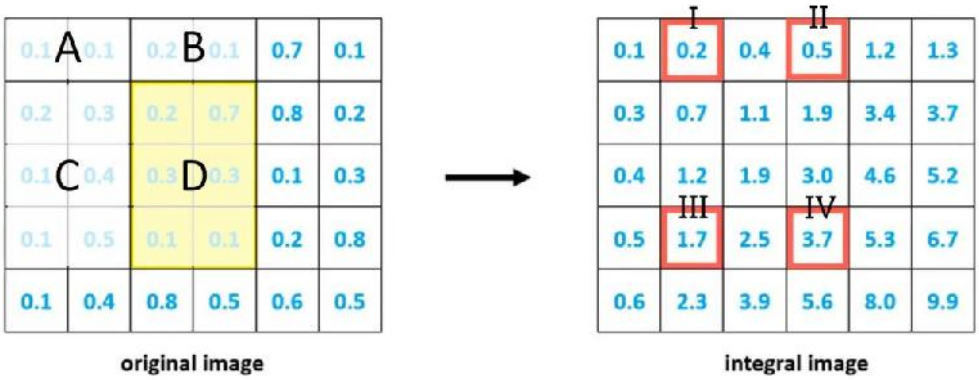
Untuk menentukan ada atau tidaknya Haar-like features pada setiap bagian citra yang diproses dan pada beberapa ukuran secara efisien, Viola-Jones menggunakan sebuah teknik yang disebut *Integral Image*.

Secara umum integrasi berarti menggabungkan beberapa unit kecil menjadi satu. Pada kasus ini, unit-unit kecil tersebut adalah nilai piksel dari citra yang sedang diproses. Nilai integral untuk setiap piksel adalah jumlah dari semua piksel yang berada di atas dan di bawah piksel tersebut sampai piksel itu sendiri. Dimulai dari pojok kiri atas kemudian ke kanan dan bawah, seluruh citra dapat diintegrasikan hanya dengan beberapa operasi saja untuk setiap pikselnya.

Seperti ditunjukkan pada Gambar 2.2, hasil integral image dari setiap piksel pada lokasi (x, y) merupakan penjumlahan dari semua piksel yang terdapat pada daerah berbentuk persegi dari pojok kiri atas citra sampai ke lokasi (x, y). Jadi formula integral image dapat dilihat pada Persamaan 2.1. Dimana ii adalah integral *image* dan i adalah citra asli. Nilai rata-rata pada persegi tersebut dapat diperoleh



Gambar 2.2 Contoh Hasil Perhitungan Integral *Image*



Gambar 2.3 Contoh Integral Image Sebagian

dengan melakukan pembagian nilai pada lokasi (x, y) dengan jumlah area pada persegi. Contoh hasil perhitungan integral image dapat dilihat pada Gambar 2.2

$$ii(x,y) = \sum_{x' \le x, y' \le y} i(x', y') \tag{2.1}$$

Namun cara di atas tidak dapat dilakukan jika ujung kiri atas dari area tersebut merupakan ujung kiri atas dari citra asli. Diperlukan cara untuk menghitung nilai penjumlahan untuk area yang tidak pada ujung kiri atas dari citra aslinya. Cara untuk memecahkan masalah tersebut digambarkan pada gambar 2.3.

Jumlah pixel pada area D dapat dihitung menggunakan 4 area lain yang berada di sekitar area D. Sehingga nilai D merupakan jumlah dari semua pixel pada persegi A+B+C+D, dikurangkan dengan jumlah persegi A+B dan A+C, kemudian

ditambahkan jumlah pada persegi A, sehingga secara singkat menjadi seperti Persamaan 2.2.

$$D = (A + B + C + D) - (A + B) - (A + C) + A \tag{2.2}$$

dimana (A+B+C+D) merupakan nilai integral image pada lokasi IV, (A+B) merupakan nilai pada lokasi II, (A+C) merupakan nilai pada lokasi III, dan A merupakan nilai pada lokasi I. Sehingga dengan integral image, dapat diperoleh nilai jumlah piksel untuk area persegi manapun dari citra asli dengan operasi matematika seperti pada Persamaan 2.3. Untuk luas area D pada Gambar 2.3 dihitung menggunakan Persamaan 2.2 adalah: $3{,}7 - 0{,}5 - 1{,}7 + 0{,}2 = 1{,}7$.

$$(X_4, Y_4) - (X_2, Y_2) - (X_3, Y_3) + (X_1, Y_1) \tag{2.3}$$

Untuk menentukan spesifikasi Haar-like feature yang akan digunakan dan mengatur tingkat threshold, Viola dan Jones menggunakan sebuah algoritma *machine learning* yang disebut *Adaptive Boosting.* Adaptive Boosting atau yang lebih dikenal dengan nama *AdaBoost* dipublikasikan pertama kali oleh Yoav Freund dan Robert Schapire yang memenangkan "Gödel Prize" pada tahun 2003. Algoritma ini dapat digabungkan dengan algoritma machine learning lainnya untuk meningkatkan kinerja algoritma tersebut. Konsep umum AdaBoost adalah mencari fitur-fitur yang memiliki tingkat pembeda yang tinggi. Hal ini dilakukan dengan mengevaluasi setiap fitur terhadap data training dengan menggunakan nilai dari fitur tersebut. Fitur yang memiliki batas perbedaan terbesar antara gambar yang mengandung objek dan gambar yang tidak mengandung objek dianggap sebagai fitur terbaik.

AdaBoost merupakan algoritma yang berfungsi untuk melakukan pemilihan dari fitur-fitur yang berjumlah sangat banyak dengan memilih fitur-fitur tertentu saja. AdaBoost mengkombinasikan banyak *weak classifier* agar dapat membentuk sebuah *strong classifier*. Classifier merupakan sebuah fungsi yang dapat mengelompokkan data input sesuai dengan kategorinya. Weak classifier merupakan classifier dengan tingkat akurasi yang tidak terlalu baik, sedangkan strong classifier merupakan classifier dengan tingkat akurasi yang sangat baik. Namun jika kita

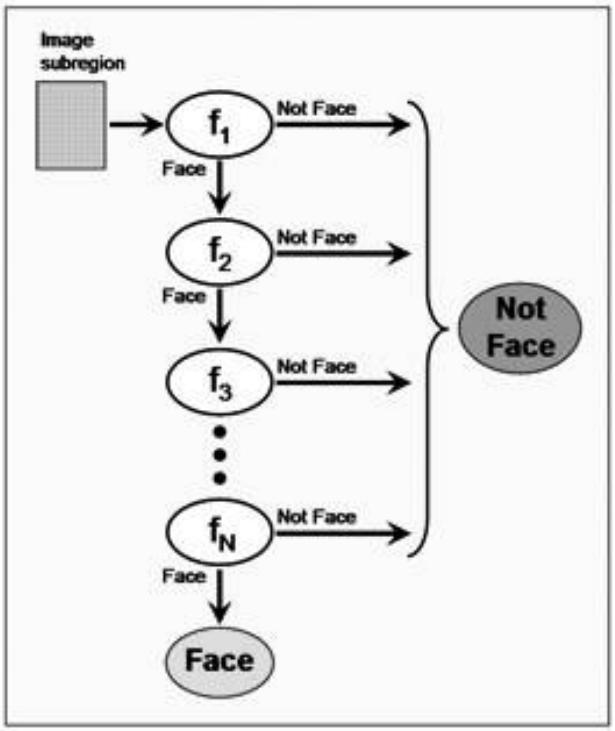
memiliki banyak weak classifier, dan hasil setiap classifier ini memberikan "sumbangan" terhadap hasil akhir, maka akan didapatkan sebuah strong classifier.

AdaBoost memilih beberapa weak classifier untuk digabungkan dan ditentukan masing-masing bobotnya. Kombinasi dari beberapa weak classifier tersebut akan membentuk sebuah strong classifier. Secara umum, boosting dilakukan secara berulang-ulang, secara bertahap menambahkan weak classifier ke dalam strong classifier. Pada setiap iterasi, sebuah classifier akan belajar dari suatu set data training lalu ditambahkan ke dalam strong classifier. Setelah ditambahkan, bobot classifier tersebut akan diubah.

Seperti yang dijelaskan di awal, selain Haar-like feature, integral image, dan algoritma AdaBoost, terdapat sebuah konsep utama lagi yaitu cascade classifier. Cascade classifier merupakan sebuah ide untuk melakukan klasifikasi secara bertingkat. Dengan mengkombinasikan proses klasifikasi secara bertingkat, maka kecepatan dari proses pendeteksian objek dapat lebih meningkat. Pencarian objek hanya akan dipusatkan kepada daerah-daerah pada citra yang terdapat peluang objek tersebut akan ditemukan.

Cascade classifier, merupakan tahap terakhir dalam algoritma Viola-Jones. Setiap filter yang terdapat pada cascade classifier ini merupakan fitur-fitur yang didapat dari proses sebelumnya, yaitu algoritma AdaBoost. Filter yang ada di setiap level di *training* agar dapat mengklasifikasikan data training yang disediakan.

Pada Gambar 2.4 ditunjukkan bahwa dalam prosesnya, jika salah satu tahapan filter menyatakan bahwa daerah citra tersebut tidak terdapat objek, maka daerah citra tersebut langsung akan dianggap bukan objek. Ketika sebuah daerah citra dapat lolos dan dianggap oleh filter tersebut terdapat objek, maka daerah citra ini akan dimasukkan ke tahapan filter yang selanjutnya. Jika sebuah daerah citra dapat melalui semua tahapan filter yang ada dalam cascade classifier ini, maka pada daerah citra tersebut terdapat objek yang dicari. Cara kerja klasifikasi bertingkat ini akan mengeliminasi banyak daerah citra yang negatif saat melalui filter yang pertama dengan proses yang sedikit. Dalam hal ini negatif berarti tidak terdapat objek yang dicari pada daerah tersebut. Kemudian, tahapan filter selanjutnya akan mengeliminasi beberapa daerah citra negatif dengan proses yang lebih besar.



Gambar 2.4 Proses Klasifikasi dengan Cascade Classifier

Setelah beberapa filter, daerah negatif yang akan dicari akan berkurang secara drastis, sehingga pencarian dapat dilakukan dengan optimal.

Dalam cascade classifier, diperlukan beberapa fitur yang merupakan hasil dari AdaBoost, fitur-fitur tersebut disusun secara urut berdasarkan bobot yang didapat dari proses AdaBoost. Semakin besar bobotnya, maka fitur tersebut akan diletakkan di depan. Hal ini disebabkan agar proses klasifikasi ini dapat mengeliminasi daerah citra negatif secepat dan seefisien mungkin. Selain itu juga digunakan untuk meminimalkan jumlah kesalahan klasifikasi.

Dengan penjelasan di atas dapat disimpulkan bahwa terdapat beberapa hal yang mempengaruhi kecepatan dan tingkat akurasi algoritma Viola-Jones dalam mendeteksi sebuah objek pada sebuah citra, yaitu jumlah tahapan filter dalam cascade classifier, jumlah fitur yang terdapat dalam setiap filter, dan threshold tiap filter tersebut. Threshold setiap level ditentukan serendah mungkin untuk dapat meloloskan daerah citra dengan objek yang dicari. Banyaknya operasi pada cascade classifier memerlukan waktu yang lebih banyak untuk dapat dijalankan.

2.1.2 Landmark detection menggunakan Active Shape Model

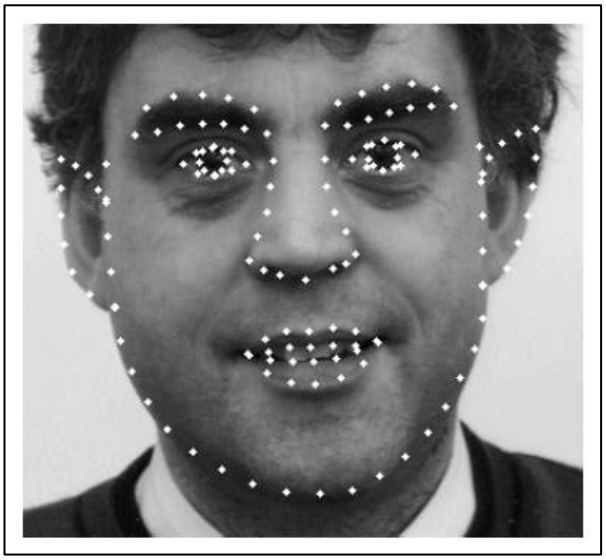
Active Shape Model (ASM) merupakan suatu model statistik yang dicetuskan pada tahun 1995 Cootes, Taylor, Cooper dan Graham [9]. Model ini perlu ditraining dan memiliki elastisitas sehingga dapat melakukan *morphing* sesuai bentuk targetnya. Sebagai contoh, ASM telah digunakan dalam bidang kedokteran untuk menganalisa hasil MRI (*Magnetic Resonance Imaging*) otak [10] dan mengidentifikasi tulang dari hasil *X-Ray* dari sendi pinggul [11]. Selain itu contoh penggunaan ASM yang lain adalah penggunaannya untuk pengenalan suara visual, dan mengklasifikasikan tanaman dari gambar.

Pada implementasinya terdapat dua buah tahapan dalam algoritma ASM, yaitu tahap *training* dan tahap *fitting*. Tahap training merupakan tahap pengekstraksian data dan pengetahuan dari data training, sementara tahap testing merupakan tahap yang akan menghasilkan informasi dari data input yang diberikan sesuai dengan hasil pembelajaran dari tahap training.

2.1.2.1 Training ASM

Tahap training membutuhkan sekumpulan citra yang relevan dengan objek yang akan dikenali. Sebagai contoh, jika tujuan objek yang akan dikenali adalah wajah manusia maka sekumpulan citra yang harus disediakan untuk proses training adalah citra wajah. Pada data training harus terdapat citra dengan tingkat keragaman yang relevan dengan objek yang akan dikenali, sehingga variasi bentuk objek dapat dimodelkan dengan baik. Sebagai contoh jika mengumpulkan data training untuk mengenali beberapa ekspresi wajah manusia dari beberapa sudut pemotretan, maka citra data training harus berisi wajah manusia dengan ekspresi yang bervariasi sesuai dengan ekspresi yang akan dapat dikenali dari beberapa sudut pemotretan yang berbeda-beda. Umumnya data training yang besar lebih baik, namun yang terpenting dari data training merupakan tingkat variasi. Model yang sederhana dapat dibentuk dari data training yang sedikit, hanya sekitar 10 sampai 20 citra saja.

Untuk membentuk data training, selain dibutuhkan citra yang relevan dengan objek yang akan dikenali, untuk setiap citra tersebut juga dibutuhkan keterangan mengenai titik-titik yang sesuai untuk merepresentasikan bentuk objek tersebut. Titik-titik ini akan digunakan untuk memperoleh shape yang sesuai untuk



Gambar 2.5 Data training dengan titik keterangannya [57]

objek yang akan dikenali. Proses membubuhi titik-titik pada data training biasanya dilakukan dengan cara manual, namun beberapa metode telah dikembangkan untuk membubuhi titik-titik tersebut secara otomatis [12]. Gambar 2.5 merupakan contoh data citra training berikut titik-titik keterangannya. Tidak terdapat aturan mengenai jumlah titik-titik harus dimiliki oleh setiap citra training, namun untuk setiap citra training harus memiliki jumlah titik-titik yang sama.

Karena data training berupa citra dua dimensi, maka sekumpulan titik-titik untuk data training tersebut dapat direpresentasikan dengan vektor 2n-dimensi, dimana n merupakan jumlah titik yang ditandai untuk setiap citra training. Sehingga untuk setiap titik yang direpresentasikan dengan (x_i, y_i) maka dapat vektor tersebut dapat ditulis dengan cara Persamaan 2.4.

$$x = (x_1, ..., x_n, y_1, ..., y_n) \tag{2.4}$$

Setiap kumpulan titik akan merepresentasikan sebuah bentuk. *Mean shape* atau bentuk rata-rata diperoleh dengan menghitung rata-rata dari N vektor, dimana N adalah jumlah citra pada data training, dan bentuk rata-rata ini akan digeser dan diletakkan pada pusat dengan melakukan translasi, serta diperbesar sehingga jumlah kuadrat dari setiap titik-titiknya akan berjumlah 1.

Setiap bentuk yang terdapat pada data training akan disejajarkan dengan bentuk rata-rata dengan menggunakan metode rotasi, translasi, dan penyekalaan.

Proses perhitungan ini dilakukan menggunakan *Procrustes analysis*. Proses menyelaraskan bentuk ini berarti meminimalkan jarak kuadrat diantara titik-titik setiap bentuk dengan titik-titik pada bentuk rata-rata. Dengan demikian jika D merupakan jumlah jarak kuadrat dari bentuk $(x_1,...,x_n,y_1,...,y_n)$ dan $(x'_1,...,x'_n,y'_1,...,y'_n)$ makan D dapat dijabarkan dengan Persamaan 2.5.

$$D = \sum_{i=1}^{n} (x_i - x'_i)^2 + (y_i - y'_i)^2 \tag{2.5}$$

Setelah proses penyelarasan ini, kumpulan bentuk dari data training akan digambarkan sebagai matriks dengan 2n baris serta n kolom. Matriks ini akan disebut sebagai matriks M. Matriks ini akan merepresentasi kisaran nilai dari bentuk yang terdapat dalam data training. Jika matriks yang ke- *j* pada data training direpresentasikan oleh vektor $(x_{1j},...,x_{nj},y_{1j},...,y_{nj})$ maka bentuk dari matriks M tampak pada Persamaan 2.6.

$$M = \begin{pmatrix} x_{11} & \cdots & x_{1N} \\ \cdots & & \cdots \\ x_{n1} & \cdots & x_{nN} \\ y_{11} & \cdots & y_{1N} \\ \cdots & & \cdots \\ y_{n1} & \cdots & y_{nN} \end{pmatrix} \tag{2.6}$$

Principal Component Analysis (PCA) akan diaplikasikan pada maktriks M untuk mengetahui komponen-komponen yang penting pada kumpulan bentuk - bentuk pada data training. Tahap pertama dalam PCA adalah menghitung kovarian matriks yang berukuran $2n \times 2n$ untuk matriks M. Kovarian matriks akan dihitung dengan Persamaan 2.7.

$$S = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})(x_i - \bar{x})^T \tag{2.7}$$

dimana $x_i = (x_{1j}, \dots, x_{nj}, y_{1j}, \dots, y_{nj})^T$ dan $\bar{x}$ merupakan bentuk rata-rata x yang didapat dari Persamaan 2.8.

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{2.8}$$

Tahap selanjutnya adalah menghitung *eigenvalue* dan *eigenvector*. Eigenvector akan diurutkan menurut nilai eigenvalue masing-masing. Eigenvector dengan eigenvalue terbesar komponen utama, dan eigenvector dengan eigenvalue kedua terbesar merupakan komponen utama kedua, dan seterusnya. Meskipun akan terdapat eigenvector berjumlah $2n$, namun mungkin saja dengan beberapa set t eigenvector yang pertama saja sudah mampu merepresentasikan sebagian besar variasi bentuk dari bentuk-bentuk yang terdapat pada data training, dimana t lebih kecil dari $2n$. Eigenvalue dari sebuah eigenvector menyatakan nilai variasi eigenvector terebut, dengan kata lain jumlah total eigenvalue setiap eigenvector merupakan total variasi dari data training. t akan dipilih sehinggan t eigenvector pertama akan merepresentasikan hampir semua, namun bukan seluruh, variasi yang ada. Jumlah t akan dipilih agar memenuhi Persamaan 2.9.

$$\sum_{i=1}^{t}\lambda_i \geq pV \tag{2.9}$$

dimana p merupakan proporsi variasi yang diinginkan agar dapat merepresentasikan variasi data training sebanyak t eigenvector. Dan V merupakan total variasi dari training data.

Maka t eigenvector akan diperoleh dan akan digunakan dalam ASM, sedangkan yang lain tidak akan digunakan. Bentuk baru akan dibangun menggunakan bentuk rata-rata dan menambahkan kombinasi linear dari t eigenvector. Jika A merupakan matriks dari t eigenvector, maka bentuk baru, x, yang akan dibangun oleh ASM akan berbentuk seperti pada Persamaan 2.10.

$$x = \bar{x} + Ab \tag{2.10}$$

dimana $\bar{x}$ merupakan bentuk rata-rata dan b adalah vektor t-dimensi yang mengandung parameter bentuk untuk bentuk yang baru.

2.1.2.2 Fitting ASM

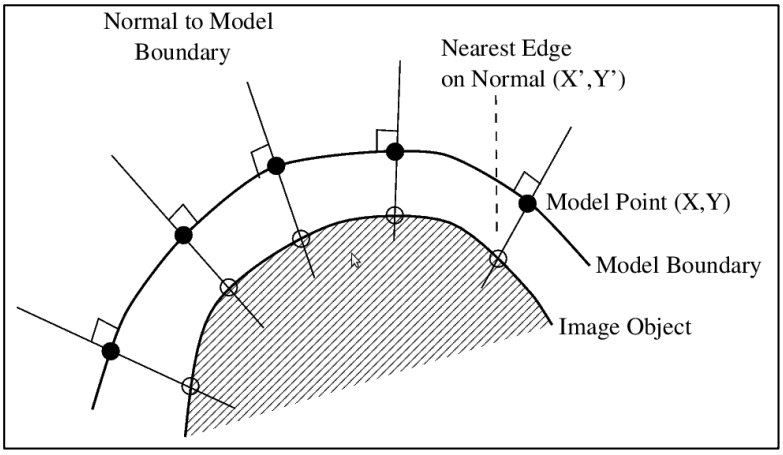
Ketika ASM telah dibentuk dari kumpulan data training, model dapat digunakan untuk menyesuaikan citra yang baru. Tujuan dari proses fitting adalah mencari parameter vektor, b, rotasi , θ, perbesaran, s dan vektor translasi, (x_t, y_t), agar bentuk x sedapat mungkin merepresentasikan object dalam citra. Tranformasi ini dapat dilihat pada Persamaan 2.11.

$$x = T_{xt,yt,s,\theta}(\bar{x} + Ab) \tag{2.11}$$

T merepresentasikan transformasi dengan translasi terhadap (x_t, y_t), perbesaraan sebesar s, dan rotasi sebesar θ. Untuk mengetahui ketepatan sebuah model, diperlukan fungsi untuk mengukurnya. Fungsi ini dapat dibentuk dengan mendeteksi tepi citra, dan menghitung jumlah jarak kuadrat antara setiap titik pada bentuk terhadap titik tepi terdekat dari citra tersebut.

Proses fitting akan dilakukan secara berulang-ulang. Untuk citra yang baru, bentuk yang pertama kali digunakan adalah bentuk rata-rata (mean shape). Jika fitting dilakukan pada video, maka bentuk yang digunakan sebagai bentuk awal untuk proses fitting adalah bentuk yang didapat setelah proses fitting pada frame sebelumnya. Bentuk awal ini akan dilambangkan dengan x. Akan ditarik sebuah garis yang tegak lurus dengan setiap titik model akan diperpanjang hingga menemukan tepi objek dari citra seperti yang terlihat pada Gambar 2.6.

Titik dimana garis perpanjangan tersebut bertemu dengan tepi akan dicatat. Kumpulan titik-titik ini disebut *image boundary points* (akan dilambangkan y). Selanjutnya, procrustes analysis akan digunakan untuk mencari nilai kombinasi $(\theta, (x_t, y_t), s)$ agar ketika di aplikasikan pada x akan meminimalkan jarak antara x



Gambar 2.6 Pengukuran Akurasi Model [58]

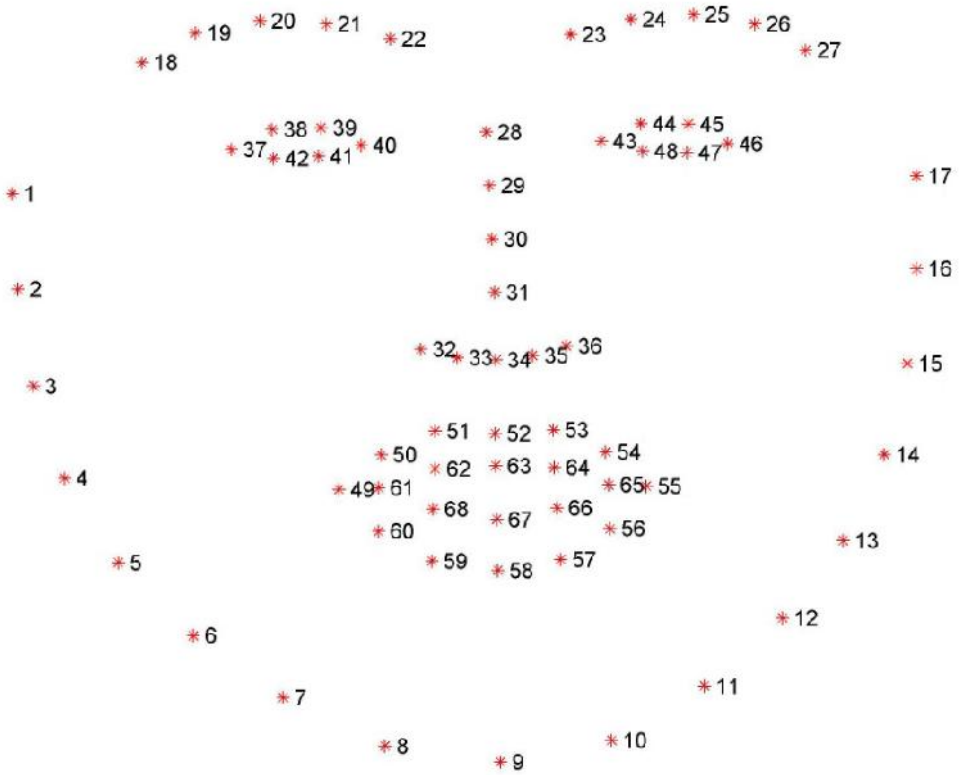
Gambar 2.7 Contoh Proses Fitting ASM [58]

dan y. Parameter bentuk model yang dilambangkan dengan b akan diperbarui menggunakan Persamaan 2.12.

$$b = A^T (y - \bar{x}) \tag{2.12}$$

Kemudian, jarak antara parameter bentuk awal dan parameter bentuk yang didapat setelah proses akan dihitung selisihnya. Jika selisih jarak berada di atas threshold, maka langkah fitting yang sama akan dilakukan menggunakan parameter yang telah diperbarui. Jika selisih jarak berada di bawah threshold, maka proses fitting telah selesai. Nilai threshold ditentukan oleh akurasi yang diinginkan dan batasan waktu untuk setiap proses fitting.

Pada Gambar 2.7 diperlihatkan hasil proses fitting setiap iterasi pada sebuah citra wajah. Hasil dari model yang terbentuk dapat digunakan untuk menentukan fitur-fitur wajah pada sebuah citra. Pada percobaan yang dilakukan oleh Cootes dan



Gambar 2.8 Rujukan titik-titik penting wajah (Facial Landmark)

Taylor, fitting sebuah wajah dilakukan dalam 18 iterasi dengan waktu fitting dibawah 1 detik.

Pada disertasi ini digunakan referensi landmark wajah dari Intelligent Behaviour Understanding Group (iBUG) yang dipublikasikan C. Sagonas pada web https://ibug.doc.ic.ac.uk/resources/facial-point-annotations [13]. Pada Gambar 2.8 tampak titik-titik acuan *landmark* wajah bayi ada 68 titik.

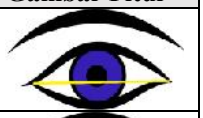
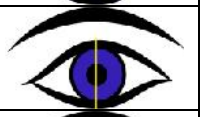
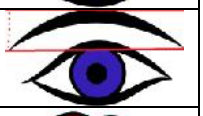
2.2 Ekstraksi Fitur

Ada dua kategori ekstraksi fitur yang biasa dilakukan pada citra wajah: fitur geometri, dan fitur tekstur. Fitur geometri yang banyak dipakai merupakan jarak antara dua titik penting wajah yang dinormalisasi [14]. Sedangkan fitur tekstur wajah cocok diekstrak menggunakan *Local Binary Pattern (LBP)*[15].

2.2.1 Fitur Geometri

Banyak sekali fitur yang bisa didapat dari landmark point yang ditemukan oleh ASM. Pada penelitian mengenai pengenalan expresi dan pengenalan usia manusia berdasarkan wajahnya banyak sekali memanfaatkan fitur geometri dan fitur texture. Sebagai contoh table 2.1 menunjukkan fitur geometri yang dipakai untuk pengenalan expresi wajah pada penelitian Zhang [14].

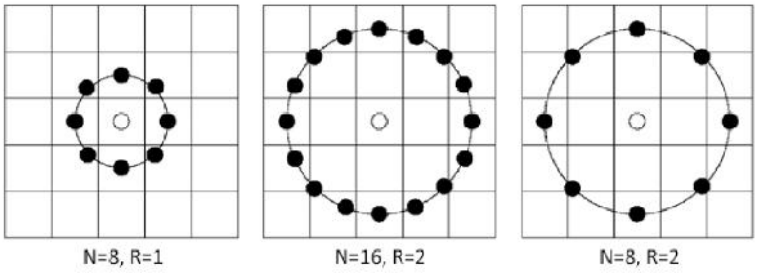
Tabel 2.1 Fitur Geometri Penelitian Terdahulu [14]

No.	Penjelasan Fitur	Gambar Fitur
1.	Panjang mata	
2.	Lebar mata	
3.	Jarak ujung kiri alis ke tengah mata	
4.	Jarak tengah kiri bawah alis ke tengah mata	
5.	Jarak tengah kanan bawah alis ke tengah mata	
6.	Jarak ujung kanan alis ke tengah mata	
7.	Sudut antara ujung kiri dengan tengah kiri atas alis	
8.	Sudut antara ujung kanan dengan tengah kanan atas alis	
9.	Lebar alis	
10.	Panjang mulut	
11.	Lebar mulut bagian dalam	

No.	Penjelasan Fitur	Gambar Fitur
12.	Sudut antara ujung kiri dengan tengah kiri atas luar mulut	
13.	Sudut antara ujung kanan dengan tengah kanan atas luar mulut	
14.	Sudut antara ujung kiri dengan tengah kiri bawah luar mulut	
15.	Sudut antara ujung kanan dengan tengah kanan bawah luar mulut	

Gambar 2.9 Jarak antara titik titik acuan pada wajah bayi yang digunakan untuk menentukan nyeri

Pada penelitian Martin Schiavenato et. al. yang berjudul "*Neonatal pain facial expression: Evaluating the primal face of pain*" [16] diteliti tentang jarak



Gambar 2.10 Ilustrasi Tetangga (N) dan Radius (R) pada LBP

titik-titik pada wajah yang dapat merepresentasikan nyeri. Gambar 2.9 menunjukkan jarak dari titik-titik landmark di wajah yang digunakan untuk mengetahui apakah bayi sedang mengalami kesakitan.

2.2.2 Fitur Tekstur

Pada disertasi ini, fitur tekstur diekstrak menggunakan *Local Binary Pattern* (LBP) yang dipublikasikan Timo Ojala [17] pada tahun 2002. LBP merupakan operator tekstur lokal yang *powerful*, dengan kompleksitas komputasi yang rendah dan lebih tidak sensitif terhadap pencahayaan.

Sudah banyak penelitian terkait ekspresi wajah yang menggunakan LBP dan dilaporkan memperoleh hasil yang baik [18], [19] dan [20]. LBP *descriptor* diperoleh dengan cara menghitung selisih nilai *graylevel* antara *pixel* P dengan nilai *graylevel* tetangganya N yang terletak pada radius R. Ilustrasi nilai tetangga dan radius dapat dilihat pada gambar 2.10. LBP melabeli sebuah pixel P dengan membandingkan nilai graylevelnya dengan nilai graylevel tetangga kemudian melakukan thresholding dan merangkumnya dalam bentuk biner.

$$LBP(X_pY_p) = \sum_{n=0}^{N-1} s(g_p - g_n)2^n \qquad 2.13$$

$$s(x) = \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases} \qquad 2.14$$

Formula perhitungan LBP dapat dilihat pada Persamaan 2.13 dan 2.14 dimana N adalah jumlah tetangga g_p adalah nilai *graylevel pixel* p dan s adalah sebuah fungsi *threshold*. Sedangkan contoh perhitungan LBP dengan R=1 dan N=8 pada sebuah pixel dapat dilihat pada gambar 2.11.

Pixel

7	6	5
7	7	4
7	9	8

Hasil fungsi s(x)

1	0	0
1		0
1	1	1

bobot pengali

1	2	4
128		8
64	32	16

Gambar 2.11 Contoh perhitungan LBP

Dari contoh pada gambar 2.10 nilai LBP *descriptor* untuk pixel yang dilingkari adalah desimal dari 11110001 atau 128+64+32+16+1 = 241. Dengan cara menambahkan zero padding di sekeliling citra maka akan didapatkan LBP descriptor yang memiliki dimensi yang sama dengan citra input. Langkah berikutnya adalah memperoleh LBP *Histogram*, dimana akan dilakukan pembentukan *histogram* (*counting*) sesuai nilai LBP *descriptor*. Dengan ini didapatkan LBP histogram sepanjang 256 (0-255).

Untuk mereduksi jumlah LBP histogram yang berjumlah 256 dilakukan *summary rotation invariant* sesuai paper Timo Ojala [17]. Dengan cara menggabungkan nilai histogram LBP yang sebenarnya memiliki ciri yang sama namun hanya di rotate. Contoh penggabungannya adalah nilai LBP 8 (0000 0100) ditambah dengan nilai LBP 16 (0000 1000), 32, 64, 128, 1, 2, dan 4. Dimana pixel-pixel tersebut hanya memiliki 1 tetangga yang lebih terang dari dirinya. Setelah mengalami *summary rotation invariant* jumlah histogram LBP menjadi tinggal 59.

2.3 Machine Learning

Pada sub bab ini dijelaskan beberapa machine learning yang dipakai dalam disertasi ini, termasuk Adachi Chaotic NN, Convolusional NN, dan LSTM.

2.3.1 Adachi Chaotic Neural Network

Model pertama dari Chaotic Neural Network dikenalkan oleh Aihara pada tahun 1989. Kemudian pada tahun 1997 Adachi and Aihara mengusulkan sebuah model *associative network* yang tersusun dari *chaotic neuron* yang saling terhubung

melalui sebuah auto-associative matriks. Neural Network ini kemudian dikenal dengan nama Adachi Neural Network (AdNN) [21]. Di tahun 2007, berdasarkan AdNN, Calitoiu mengenalkan sebuah modifikasi dari basic AdNN dan menyebutnyait Modified AdNN (M-AdNN) [22]. Kemudian di tahun 2012-2013 Qin dan Oommen mengusulkan sebuah *tuning up* terhadap M-AdNN untuk meningkatkan kemampuan M-AdNN dalam menagani *pattern recognition*, metode yang telah di- improve ini kemudian mereka sebut Ideal-M-AdNN [23].

Adachi's Model of Chaotic Neural Networks atau biasa disebut AdNN, sebenarnya adalah semuah model neural network yang menyerupai model Hopfield, dimana arsitektur ini terdiri dari N neuron, dan disusun sebagai completely connected graph . Setiap neuron berkomunikasi dengan setiap neuron lain termasuk dirinya sendiri. AdNN dimodelkan sebagai dynamical associative memory, yang mengikuti Persamaan 2.15, 2.16, dan 2.17.

$$x_i(t+1) = f(\eta_i(t+1) + \xi_i(t+1) \tag{2.15}$$

$$\eta_i(t+1) = k_f \eta_i(t) + \sum_{j=1}^{N} w_{ij} x_j(t) \tag{2.16}$$

$$\xi_i(t+1) = k_r \xi_i(t) - \alpha x_i(t) + a_i \tag{2.17}$$

- $x_i(t)$ adalah output dari neuron ke-i pada waktu ke-t.
- f adalah fungsi logistic dengan parameter kemiringan yang memenuhi Persamaan 2.18.

$$f(y) = \frac{1}{1 + \exp(-y/\varepsilon)} \tag{2.18}$$

- k_f adalah parameters pembusukan (*decay*) dan k_r adalah parameter ketahanan (*refractoriness*).
- w_{ij} adalah bobot neural network dari neuron ke-j menuju neuron ke-i.
- a_i adalah bias.
- Nilai bobot w_{ij} ditentukan dengan Persamaan 2.19

$$w_{ij} = \frac{1}{p} \sum_{s=1}^{p} (2x_i^s - 1)(2x_j^s - 1) \tag{2.19}$$

- Dimana x_s^i adalah komponen ke-i dari pattern ke-s

- Sedangkan nilai bias ditentukan dengan Persamaan 2.20

$$a_i = 2 + 6x_i^s \tag{2.20}$$

2.3.1.1 Modified Adachi Neural Network (M-AdNN)

Calitoiu et al pada paper [22] mengusulkan modifikasi terhadap Adachi's chaotic neural networks agar dapat melakukan pattern recognition. M-AdNN mengenalkan dua buah internal state yang akan digunakan oleh semua neuron, yaitu $\eta(t)$ and $\xi(t)$ yang memenuhi Persamaan 2.21 dan 2.22.

$$\eta_i(t+1) = k_f \eta\ (t) + \sum_{j=1}^{N} w_{ij} x_j(t) \tag{2.21}$$

$$\xi_i(t+1) = k_r \xi\ (t) - \alpha x_i(t) + a_i \tag{2.22}$$

Setiap step t + 1, dua global state ini di update dengan nilai $\eta_N(t + 1)$ dan $\xi_N(t + 1)$. Pada M-AdNN, setiap kali kita menghitung internal state yang baru, hanya akan menggunakan memori dari internal state neuron ke-N. Hal ini sangat berbeda dengan AdNN yang menggunakan nilai internal state dari semua neuron pada waktu ke-t.

Perubahan kedua adalah pada aturan pemberian bobot. Pemberian nilai inisial untuk bobot w dikembalikan ke Persamaan 2.23. Pada AdNN, jika nilai $x_i^s = x_j^s = 0$ untuk semua s(1..p), maka w_{ij} akan bernilai 1. Namun untuk kasus yang sama pada M-AdNN akan menghasilkan nilai 0. Hal ini menyebabkan M-AdNN memiliki nilai feedback yang lebih kecil dibandingkan AdNN dan akan mempercepat proses.

$$w_{ij} = \frac{1}{p} \sum_{s=1}^{p} (x_i^s)(x_j^s) \tag{2.23}$$

2.3.1.2 Ideal Modified Adachi Neural Network (Ideal-M-AdNN)

Tujuan utama dari penelitian Ideal-M-AdNN ini adalah untuk menunjukkan bahwa M-AdNN bila disetel dengan tepat akan dapat melakukan pattern recognition dengan baik. Sebuah sistem chaotic pattern recognition tidak sama seperti pattern reconition pada umumnya yang melakukan "proklamasi" atau

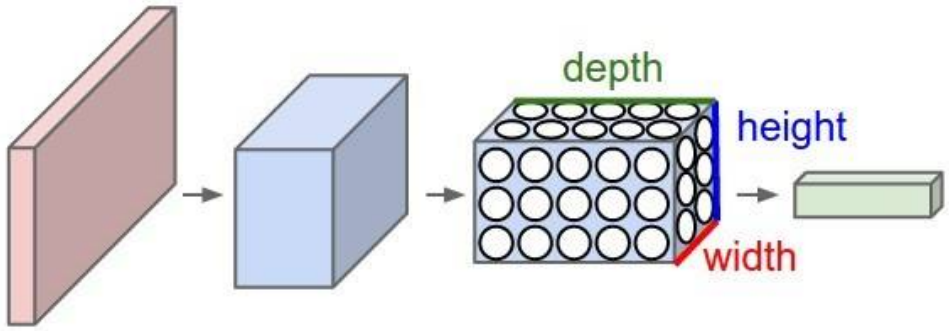
dengan jelas menyebutkan bahwa pattern dikenali. Sistem chaotic pattern recognition akan mengeluarkan sinyal periodic saat sebuah pattern dikenali. Pada sisi lain bila pattern yang dimasukkan bukan merupakan pattern yang pernah ditrainingkan maka sistem akan mengeluarkan sinyal chaotic.

Pada penelitian Calitoiu M-AdNN sudah dapat mengolah pattern recognition, namun performa M-AdNN masih belum seperti yang diharapkan. Ada dua modifikasi yang di usulkan dalam penelitian Ideal_M-AdNN oleh Qin dan Oomen: Yang pertama adalah mengembalikan inisialisasi bobot ke algoritma Asal karya Adachi, yang kedua menentukan parameter kelandaian ε yang semula 0,00015 menjadi 0,015 [23].

2.3.2 Convolutional Neural Network (ConvNet)

Convolutional neural networks, atau ConvNet merupakan sebuah jenis *feed-forward* neural network dimana konektivitas antar neuronnya terinspirasi oleh visual cortex pada otak hewan [24]. Setiap neuron akan merespon stimulus dalam sebuah wilayah tertentu yang disebut dengan *receptive field*. Receptive field untuk setiap neuron saling tumpang tindih sebagian sehingga dapat membentuk sebuah *visual field*. Secara matematis, respon dari neuron terhadap stimulus dalam sebuah receptive field dapat diperkirakan menggunakan operasi konvolusi (convolution). ConvNet banyak digunakan untuk berbagai macam aplikasi diantarnya pengenalan gambar dan video, sistem rekomendasi, dan *natural language processing*. ConvNet juga dikenal dengan nama space invariant artificial neural network (SIANN) karena memiliki arsitektur yang bersifat *shared weights*.

ConvNet banyak digunakan untuk memproses gambar karena fully connected network biasa kurang efisien untuk digunakan memproses gambar dengan ukuran yang agak besar. Sebagai contoh apabila gambar input berukuran 32x32x3, maka pada hidden layer pertama akan terdapat 3072 weight. Jumlah ini masih dapat diatasi, namun apabila gambar input yang digunakan berukuran 200x200x3 maka pada layer pertama saja akan terdapat 120000 weight. Apabila fully connected memiliki banyak hidden layer, maka jumlah parameter akan bertambah dengan sangat cepat yang dapat mengakibatkan network cepat mengalami overfitting karena terlalu banyak parameter. Sedangkan pada ConvNet,



Gambar 2.12 Ilustrasi Convolutional Neural Networks

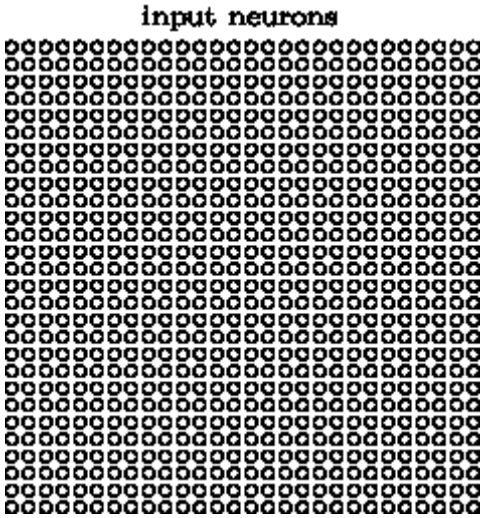
neuron akan tersusun ke dalam sebuah volume 3 dimensi, tidak seperti pada fully connected network yang hanya memiliki 1 dimensi. Kemudian, setiap neuron pada sebuah layer hanya akan terhubung dengan beberapa neuron pada layer selanjutnya sehingga akan mengurangi jumlah parameter yang diperlukan. Setiap layer pada ConvNet bertujuan untuk mengubah sebuah volume dengan dimensi tertentu menjadi sebuah volume dengan dimensi lain. Sebagai contoh dari 200x200x3 menjadi 100x100x32.

Dapat dilihat pada Gambar 2.12, ConvNet akan mengubah sebuah volume dengan sebuah dimensi ke dimensi lain yang berbeda. Pada akhir ConvNet, volume fitur biasanya berukuran sangat kecil yang merupakan representasi sederhana dari gambar input.

2.3.2.1 Convolutional Layer

Convolutional layer merupakan layer inti dari ConvNet yang dapat menjelaskan mengapa ConvNet dapat memiliki performa yang lebih baik dibanding dengan fully connected network biasa. Convolutional layer merupakan tempat dimana proses komputasi yang paling berat pada ConvNet dilakukan. Terdapat dua konsep penting pada convolutional layer yaitu local receptive field dan shared weights.

Kedua konsep tersebut merupakan alasan mengapa ConvNet memiliki sifat yang disebut *translation invariance* dan dapat berjalan lebih cepat dibandingkan *fully connected network* biasa. Local *receptive field* merupakan sebuah *pattern* konektivitas antar neuron yang dimodelkan berdasarkan *visual cortex* milik hewan.



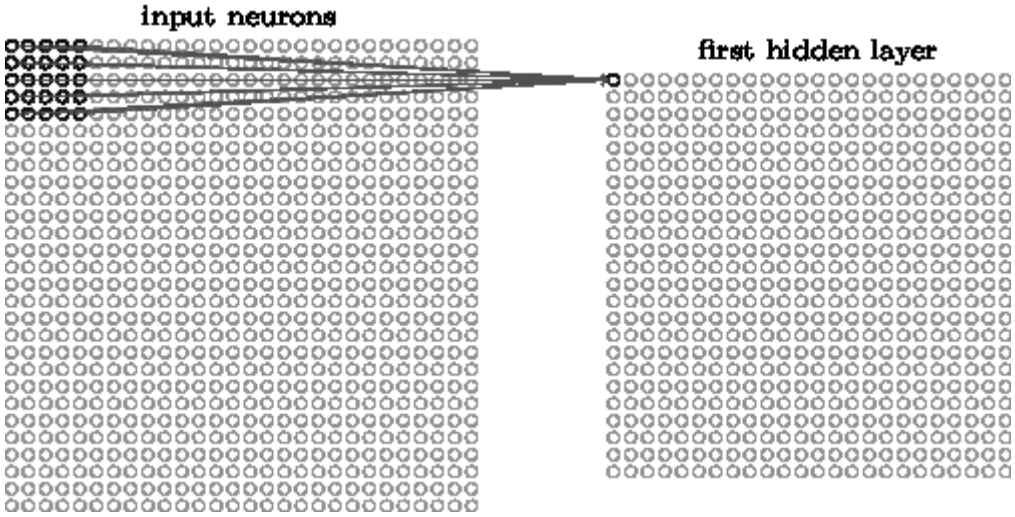
Gambar 2.13 Susunan Neuron pada Layer Input

Neuron-neuron pada *visual cortex* hewan akan merespon terhadap stimulus yang diberikan oleh neuron-neuron lain yang berada pada sebuah daerah tertentu dan terbatas, atau yang sering disebut sebagai receptive field. Setiap receptive field milik neuron-neuron yang berbeda memiliki sebuah daerah yang *overlap* sehingga apabila seluruh receptive field disambung, maka akan membentuk sebuah daerah penglihatan. Pada ConvNet, respon sebuah neuron terhadap receptive field dapat dilakukan dengan menggunakan operasi konvolusi.

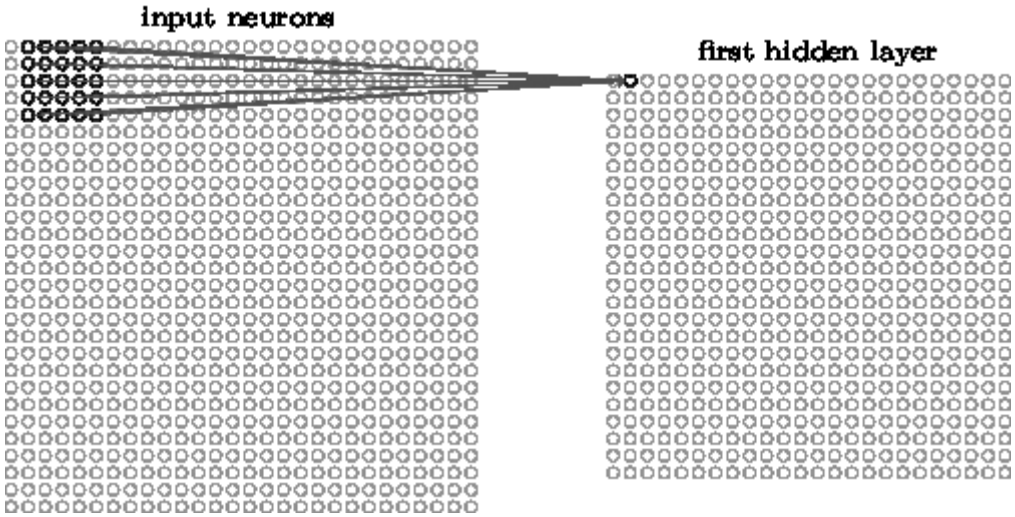
Seperti yang telah dijelaskan sebelumnya bahwa input ConvNet pada umumnya memiliki dimensi yang lebih dari 1. Sebagai contoh apabila input adalah sebuah gambar grayscale berukuran 28x28, maka susunan neuron pada layer input dapat digambarkan seperti contoh pada Gambar 2.13.

Pada ConvNet sebuah neuron pada sebuah layer akan mempunyai koneksi ke neuron pada layer selanjutnya seperti layaknya pada sebuah fully connected network. Namun tidak seperti yang dilakukan oleh fully connected network, sebuah neuron pada ConvNet tidak terhubung dengan seluruh hidden neuron pada sebuah hidden layer, melainkan hanya terhubung dengan beberapa neuron saja yang mencakup sebuah daerah kecil yang dinamakan dengan local receptive field.

Local receptive field sering disebut sebagai sebuah *filter* atau *kernel*. Setiap koneksi sebuah neuron dengan sebuah hidden neuron memiliki weight dan setiap hidden neuron memiliki nilai bias yang keduanya dapat dilatih, sama seperti sebuah



Gambar 2.14 Local Receptive Field atau Kernel Berukuran 5x5



Gambar 2.15 Kernel Setelah Satu Kali *Slide*

hidden neuron pada fully connected network. Sebagai contoh apabila sebuah neuron memiliki koneksi dengan 25 neuron yang berada di dalam sebuah daerah berukuran 5x5, maka koneksi neuron tersebut dapat dilihat pada Gambar 2.14. Daerah berukuran 5x5 inilah yang disebut dengan filter atau kernel. Ukuran sebuah kernel merupakan sebuah hyperparameter yang ditentukan pada saat pembuatan arsitektur ConvNet yang biasa disebut sebagai *kernel size*. Kernel size yang digunakan tidak dapat melebihi ukuran input gambar.

Sebuah kernel akan di-*slide* sepanjang dan selebar input gambar dengan sebuah jarak tertentu yang dinamakan dengan *stride*. Sebagai contoh apabila sebuah

kernel 5x5 di-slide pada gambar berukuran 28x8 dengan nilai stride 1, akan menghasilkan hidden layer berukuran 24x24 seperti contoh Gambar 2.15.

Sebuah kernel akan dislide secara terus menerus dengan sebuah nilai stride tertentu hingga mencakup seluruh gambar input. Nilai stride dapat diisi dengan nilai yang berbeda untuk jarak vertikal dan horizontal. Namun pada aplikasi-aplikasi yang populer digunakan nilai yang sama karena pada umumnya input gambar yang digunakan berbentuk persegi. Proses sliding secara terus menerus ini disebut sebagai proses konvolusi.

Sebuah kernel adalah tempat convolutional layer menyimpan parameter-parameternya. Pada proses konvolusi dapat dilihat bahwa ketika sebuah kernel dislide sepanjang input gambar, maka parameter-parameter yang ada akan dishare dengan neuron-neuron lain. Konsep inilah yang disebut dengan shared weights, dimana tidak seperti pada fully connected network dimana seluruh neuron memiliki sebuah parameter dengan seluruh neuron pada sebuah hidden layer, pada convolutional layer hanya terdapat beberapa parameter saja yang digunakan untuk menghubungkan seluruh neuron input dengan hidden layer. Hal ini yang menyebabkan ConvNet dapat berjalan dengan cepat dan lebih rendah dalam pemakaian memory karena jumlah parameternya tidak sebanyak sebuah fully connected network.

Sebuah kernel yang di-slide sepanjang dan selebar gambar input juga memberikan sebuah manfaat yaitu menjadikan ConvNet memiliki sifat translation invariance. Translation invariance merupakan kemampuan ConvNet untuk mendeteksi sebuah pattern walaupun gambar input telah ditranslasi. Hal ini dapat dilakukan karena seluruh parameter pada kernel dishare dengan seluruh neuron input sehingga seluruh neuron pada hidden layer dapat mendeteksi pattern yang sama hanya berbeda lokasi, sama seperti dengan yang terjadi pada visual cortex hewan. Sehingga neuron-neuron pada hidden layer dapat disebut sebagai sebuah daerah penglihatan dari convolutional layer.

Pada beberapa aplikasi proses konvolusi dapat mempertahankan dimensi volume agar sama dengan dimensi volume input. Hal ini dilakukan dengan cara menambahkan zero-padding dengan jumlah tertentu tergantung dari besar kernel yang digunakan dan menggunakan nilai stride = 1 pada gambar input. Zero-padding

0	0	0	0	0	0
0	35	19	25	6	0
0	13	22	16	53	0
0	4	3	7	10	0
0	9	8	1	3	0
0	0	0	0	0	0

Gambar 2.16 Ilustrasi Zero-padding

merupakan penambahan angka 0 di luar gambar input dengan jumlah tertentu. Pada Gambar 2.16 dapat dilihat sebuah gambar dengan ukuran 4x4 yang diberi zero-padding sebesar 1 yang kemudian mengubah ukuran gambar input menjadi 6x6.

Jumlah zero-padding yang perlu ditambahkan P berdasarkan sebuah ukuran kernel F dan stride $S = 1$ untuk mempertahankan dimensi gambar input setelah operasi konvolusi dapat dihitung menggunakan Persamaan 2.24.

$$P = (F - 1)/2 \tag{2.24}$$

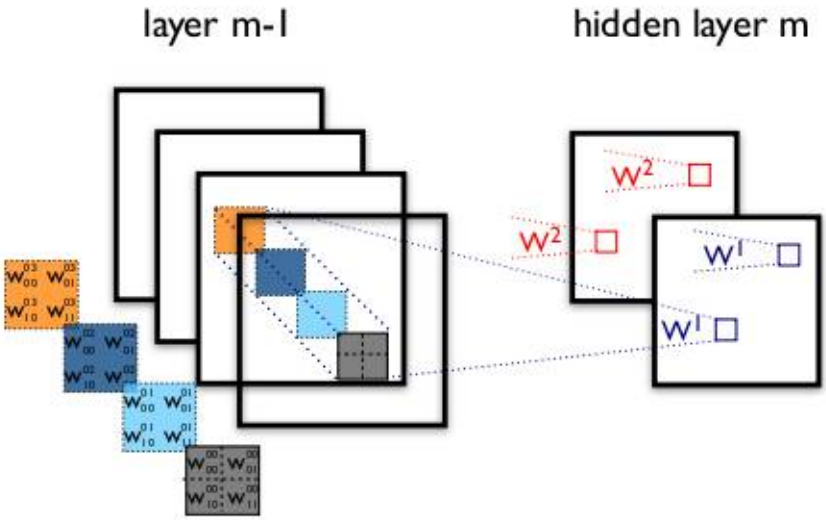
Sebagai contoh pada Gambar 2.16, zero-padding yang digunakan adalah sebesar 1, maka untuk mempertahankan dimensi, dapat digunakan filter berukuran 3x3. Kemudian untuk menghitung ukuran hidden layer atau *feature map* yang dihasilkan W'xH'xD' dari convolutional layer dapat digunakan Persamaan 2.25 sampai 2.27.

$$W' = \frac{W-F+2P}{S} + 1 \tag{2.25}$$

$$H' = \frac{H-F+2P}{S} + 1 \tag{2.26}$$

$$D' = \mathrm{K} \tag{2.27}$$

Pada Persamaan 2.25, W merupakan panjang gambar input, F merupakan ukuran kernel yang digunakan, P merupakan jumlah zero-padding yang ditambahkan, dan S merupakan nilai stride. Pada Persamaan 2.26, H adalah tinggi



Gambar 2.17 Convolutional Layer

gambar input. Pada Persamaan 2.27, K adalah jumlah kernel pada convolutional layer. Sebagai contoh pada Gambar 2.13 dan Gambar 2.14, input gambar diketahui memiliki ukuran $W = 28,\ H = 28$ dengan ukuran kernel $F = 5$, zero-padding $P = 0$, dan stride $S = 1$, maka ukuran output yang dihasilkan adalah $W' = H' = ((28 - 5 + 0)/1) + 1 = 24$, yang berarti ukuran output adalah 24x24. Karena jumlah kernel yang digunakan adalah 1, maka kedalaman dari output adalah 1 sehingga dimensi lengkap dari output adalah 24x24x1.

Sebuah kernel merupakan tempat convolutional layer menyimpan parameter-parameternya atau yang disebut dengan weight. Sebuah convolutional layer idealnya memiliki beberapa kernel dengan ukuran yang sama dengan nilai weight yang berbeda-beda antar kernel. Kernel yang berbeda berujuan untuk mendeteksi pattern yang berbeda pada gambar input. Semakin banyak kernel, maka convolutional layer dapat menghasilkan daerah penglihatan yang berbeda-beda sehingga semakin banyak pattern yang dapat dideteksi oleh ConvNet.

Gambar 2.17 merupakan contoh convolutional layer antara layer m dengan layer $m - 1$. Empat kotak kecil dengan ukuran 2x2 dan warna yang berbeda merupakan kernel dari convolutional layer $m - 1$. Sehingga convolutional layer pada layer $m - 1$ memiliki 4 buah kernel yang akan menghasilkan 4 buah feature map. Sedangkan pada hidden layer m dapat dilihat yang dimaksud dengan *local*

connectivity pada w^1 yang hanya terkoneksi dengan sebagian kecil neuron pada layer $m - 1$. Dengan hal ini, maka convolutional layer mampu untuk mengurangi jumlah parameter yang dibutuhkan dibandingkan dengan jumlah parameter pada fully connected layer. Jumlah weight pada sebuah convolutional layer dapat dihitung menggunakan Persamaan 2.28.

$$n = (F.F.D).K \tag{2.28}$$

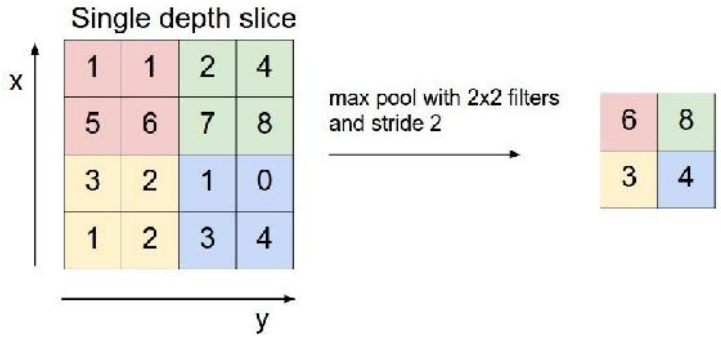
Sebagai contoh pada Gambar 2.13 dan 2.14, jumlah weight pada convolutional layer adalah $5.5.1.1 = 25$ untuk input gambar berukuran 28x28 dan hidden layer 24x24. Sedangkan pada sebuah fully connected layer, maka jumlah weight yang dibutuhkan adalah $28.28.1.24.24.1 = 451584$ yang merupakan jumlah parameter yang sangat banyak hanya untuk sebuah hidden layer.

Semakin banyak jumlah convolutional layer pada sebuah ConvNet, maka fitur-fitur atau pattern yang sangat kompleks dari gambar dapat dideteksi. Pada layer-layer awal, sebuah convolutional layer umumnya mendeteksi fitur-fitur dasar seperti garis dan bentuk. Seiring dengan bertambahnya convolutional layer, maka fitur-fitur yang dapat dideteksi semakin abstrak yang sangat sulit dimengerti oleh manusia.

2.3.2.2 Pooling Layer

Layer penyusun sebuah ConvNet berikutnya adalah pooling layer. Pooling layer biasanya diletakkan setelah convolutional layer dengan tujuan untuk menyederhankanan fitur yang diperoleh dari convolutional layer. Dengan disederhanakannya fitur, maka setelah melewati pooling layer, dimensi dari output convolutional layer akan berkurang.

Jenis pooling yang sering digunakan adalah maxpooling. Sama seperti pada convolutional layer, pada pooling layer juga terdapat sebuah kernel. Perbedaannya adalah jika pada convolutional layer jumlah kernel ada banyak, pada pooling layer hanya terdapat satu buah kernel saja. Ukuran kernel maxpooling yang paling sering digunakan adalah 2x2 dengan stride sebesar 2.



Gambar 2.18 Ilustrasi Maxpooling Layer

Seperti yang dilakukan pada convolutional layer, kernel pada maxpooling akan di-slide hingga mencakup seluruh neuron input sambil mengambil nilai aktivasi maksimum dari neuron yang berada pada cakupan kernel. Contoh operasi yang dilakukan oleh maxpooling layer dapat dilihat pada Gambar 2.18.

Pada contoh Gambar 2.18, maxpooling layer berisi sebuah kernel berukuran 2x2 dan stride sebesar 2. Maka kernel akan di-slide pada input neuron dengan jarak 2 horizontal dan vertikal sambil mengambil nilai maksimum dari seluruh neuron yang berada dalam cakupan kernel masing-masing. Dengan konfigurasi kernel seperti ini, maka maxpooling mampu menghilangkan sebesar 75% aktivasi dan melakukan *downsampling* pada input sebesar 50% yang mengakibatkan ukuran output menjadi setengah dari input. Pada pooling layer tidak perlu ditambahkan zero-padding karena tujuan dari pooling layer adalah untuk menyederhanakan input.

Maxpooling hanya akan mengurangi panjang dan lebar dari input. Besar kedalaman atau feature map dari input akan tetap setelah dilakukan maxpooling. Pada dasarnya yang dilakukan oleh maxpooling adalah mendapatkan sebuah informasi penting dari daerah-daerah kecil. Maxpooling baik untuk digunakan pada masalah-masalah dimana letak suatu fitur tidak penting selama fitur tersebut ditemukan seperti pada masalah-masalah deteksi. Karena yang dilakukan oleh maxpooling adalah menghilangkan informasi dari lokasi fitur pada sebuah gambar. Sedangkan pada masalah-masalah seperti *face recognition* dimana posisi objek seperti mata, hidung, dan mulut menjadi penting, maka maxpooling dapat menyebabkan masalah. Untuk masalah-masalah seperti ini, pada umumnya yang

dilakukan adalah dengan menggunakan convolutional layer yang memiliki stride yang agak besar yang juga dapat mengurangi aktivasi dari neuron-neuron yang dijadikan input. Namun terlepas dari hal tersebut, dengan mengurangi jumlah aktivasi atau parameter, maka pelatihan ConvNet dapat berjalan lebih cepat dan menggunakan lebih sedikit memori untuk komputasi.

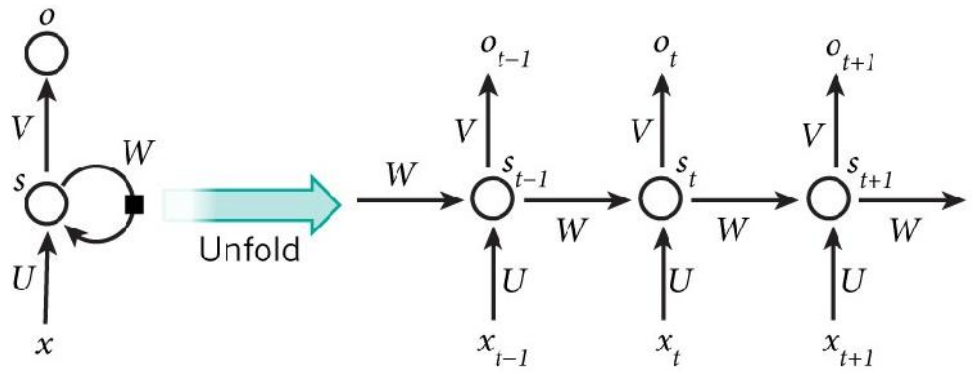
2.3.2.3 Arsitektur ConvNet

Sebuah *Convolutional Neural Networ*k yang paling dasar adalah sebuah *neural network* yang memiliki kombinasi layer yang terdiri dari *convolutional layer, pooling layer,* dan *fully connected layer*. Pada umumnya sebuah ConvNet memiliki beberapa convolutional layer, setelah beberapa convolutional layer kemudian disambung dengan pooling layer untuk menyederhanakan aktivasi. Sedangkan *fully connected layer* dan *softmax* biasanya diletakkan pada akhir ConvNet yang berguna untuk mengambil keputusan output. Jenis fungsi aktivasi yang biasa digunakan pada ConvNet adalah ReLu, baik untuk *convolutional layer* maupun *fully connected layer*-nya.

2.3.3 ***Recurrent Neural Network* (RNN)**

Recurrent Neural Network atau yang selanjutnya disebut dengan RNN adalah suatu jenis Neural Network yang memiliki arsitektur yang sedikit berbeda bila dibandingan *Feed Forward Neural Network*. Pada Neural Network biasa atau *feed forward neural network*, network akan mengeluarkan output yang merupakan hasil dari network tersebut dan output ini sifatnya independen terhadap input yang diterima. Namun pada RNN, input dan outputnya memiliki suatu hubungan, dimana output sebuah neuron RNN akan menjadi salah satu input neuron RNN tersebut pada iterasi berikutnya.

Masalah yang cocok ditangani aleh RNN adalah masalah yang memperhatikan *sequence* atau urutan input. Sebagai contoh untuk memprediksi sebuah kata yang akan muncul pada suatu kalimat, maka akan dibutuhkan kata-kata sebelumnya dimana kata yang muncul akan membentuk kalimat yang baik dan



Gambar 2.19 Recurrent Neural Network

benar. RNN unit akan diulangi sebanyak kata-kata sebelumnya yang disebut juga dengan *sequence length* atau *timesteps*.

Pada RNN, output suatu RNN unit akan menjadi input untuk RNN unit berikutnya sehingga dapat dikatakan bahwa RNN melakukan suatu aksi yang sama secara berulang-ulang (recurrent) untuk semua elemen pada informasi yang sekuential. Output yang dikeluarkan pun juga akan terpengaruh oleh input-input sebelumnya sehingga output dari RNN tidak independen terhadap 1 input saja seperti yang terjadi pada neural network biasa.

RNN dapat dianggap memiliki kemampuan untuk mengingat output yang pernah dihasilkan. Secara teori, RNN dapat mengingat informasi pada jumlah sequence yang sangat panjang, namun pada kenyataannya RNN tidak akan dapat mengingat sampai jauh ke belakang karena masalah vanishing gradient.

Gambar 2.19 menunjukkan diagram RNN sederhana yang dibuka sehingga terlihat jelas alur informasi yang diumpan. Sebagai contoh, apabila panjang sekuen yang ingin diolah menggunakan RNN memiliki panjang 3 kata, maka network akan diulang sebanyak 3 RNN unit seperti pada Gambar 2.18. Perhitungan dari RNN dapat dihitung dengan Persamaan 2.29 dan 2.30.

$$s_t = \sigma(Ux_t + Ws_{t-1}) \quad (2.29)$$

$$o_t = \sigma(Vs_t) \quad (2.30)$$

Pada Persamaan 2.29 dan 2.30, x_t merupakan input pada waktu t, U merupakan bobot koneksi antara input dengan hidden layer pada suatu waktu *t*

tertentu, W merupakan bobot antar hidden layer pada waktu t dan $t + 1$, V merupakan bobot antara hidden layer dengan output layer pada suatu waktu t tertentu, sedangkan o_t merupakan output dari suatu hidden layer pada suatu waktu t tertentu.

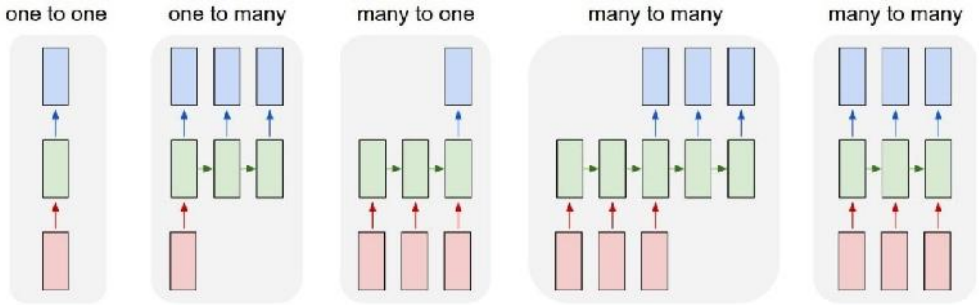
Nilai dari s_t merupakan hidden state dari network yang akan selalu diumpankan kehidden layer selanjutnya dengan kata lain nilai ini merupakan memori dari RNN. Nilai U,W,dan V yang terdapat pada semua hidden layer akan memiliki nilai yang sama sehingga dapat dikatakan bahwa pada setiap waktu, RNN melakukan aksi yang sama hanya saja input dan hidden state yang berbeda.

Pada Gambar 2.18 dapat dilihat bahwa pada setiap waktu, hidden layer akan menghasilkan suatu output. Output dari RNN unit pada tiap satuan t belum tentu akan selalu dioutputkan dari RNN mengingat terdapat berbagai macam kebutuhan yang ada. Contoh apabila RNN digunakan untuk menebak kata yang terakhir pada suatu kalimat yang belum lengkap, RNN tidak perlu mengeluarkan output dari setiap unit yang ada karena yang dibutuhkan hanya output dari unit yang paling terakhir yang diharapkan akan berisi kata-kata terakhir dari kalimat tersebut. Terdapat beberapa jenis RNN berdasarkan jumlah input dan jumlah output dari RNN.

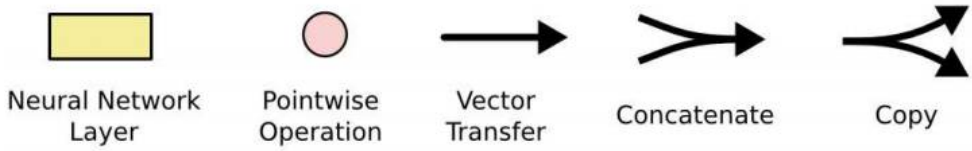
Pada Gambar 2.20 dapat dilihat bahwa jumlah input dan output dari RNN beranekaragam tergantung dari kebutuhannya. Pada Disertasi ini, akan digunakan *many to one* RNN dikarenakan pada disertasi ini memiliki frame sequence sebagai input dan sebuah output.

2.3.4 Long Short Term Memory Network (LSTM)

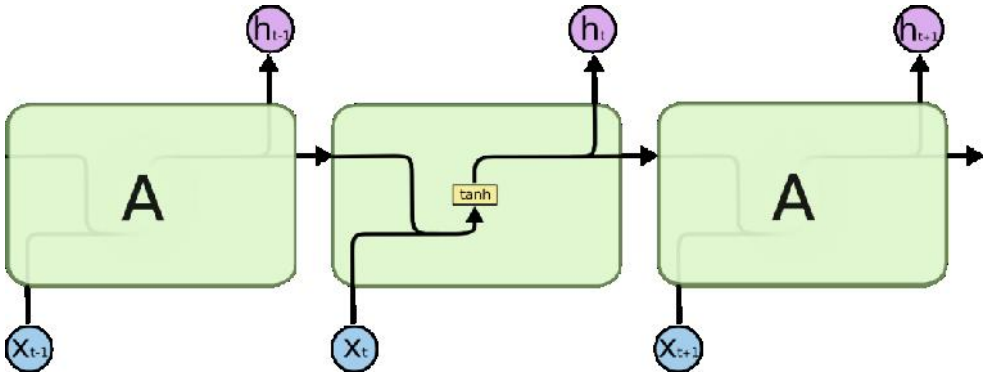
LSTM pertama kali ditemukan oleh Sepp Hochreitern dan Jürgen Schmidhuber pada tahun 1997 [25]. LSTM merupakan model yang paling banyak digunakan pada dunia *Deep Learning* untuk kepentingan *Natural Language Processing* hingga saat ini. LSTM juga disempurnakan oleh banyak orang seperti Felix Gers, Fred Cummins, Santiago Fernandez, Justin Bayer, Daan Wierstra, Julian Togelius, Faustian Gomez, Matteo Gagliolo, dan Alex Graves [26] [27].



Gambar 2.20 Beberapa Jenis Recurrent Neural Network



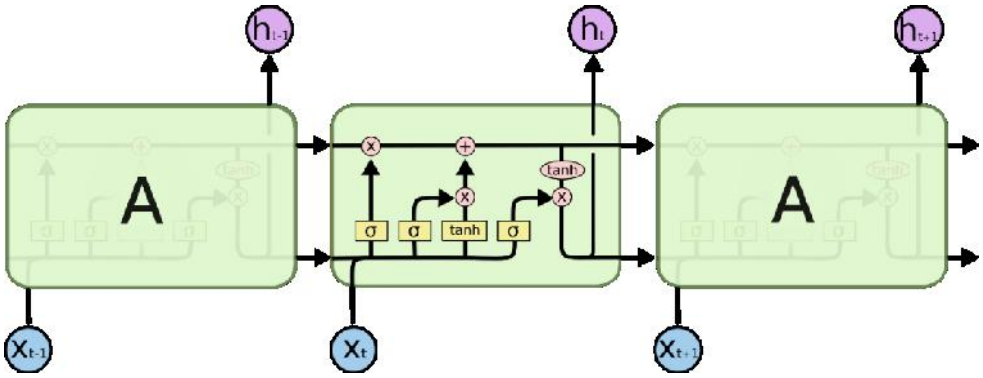
Gambar 2.21 Notasi Penjelas Unit



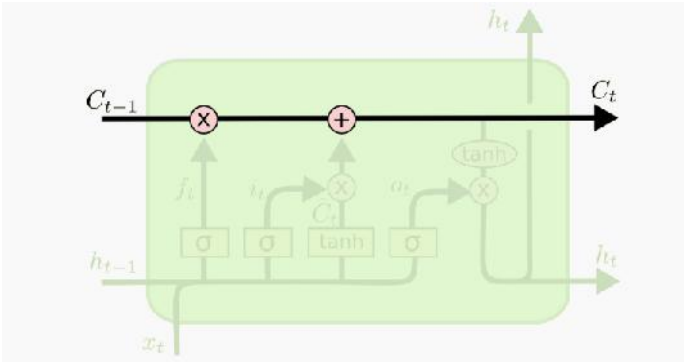
Gambar 2.22 RNN Unit

LSTM merupakan suatu unit yang didesain khusus untuk menangani masalah *long term dependencies*. Desain arsitektur sebuah RNN yang tersusun dari LSTM unit akan disebut sebagai LSTM network. Arsitektur LSTM Network memiliki arsitektur yang sama dengan RNN biasa, hanya saja komponen dari unit yang digunakan berbeda.

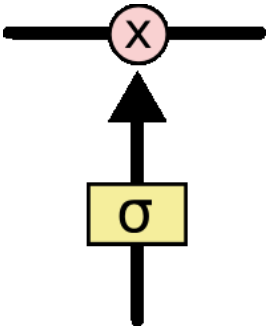
Gambar 2.21 merupakan notasi yang akan digunakan pada gambar-gambar selanjutnya dimana masing-masing garis mewakili suatu vektor yang membawa output dari satu node menjadi input pada node selanjutnya. Lingkaran berwarna



Gambar 2.23 LSTM Unit



Gambar 2.24 LSTM Cell State



Gambar 2.25 LSTM Gate

merah muda menandakan operasi yang dijalankan secara *pointwise*, dan kotak kuning merupakan layer neural network.

Dengan membandingkan Gambar 2.22 dan 2.23 dapat terlihat bahwa LSTM masih memiliki struktur yang sama dengan RNN biasa, hanya saja unit yang digunakan untuk penanganan memori lebih kompleks. Dapat dilihat bahwa pada LSTM terdapat 4 neural network yang menangani operasi berbeda yang mana pada

RNN unit hanya memiliki 1 neural network saja. Kunci dari LSTM adalah state dari cell yang dilambangkan dengan garis pada bagian atas diagram.

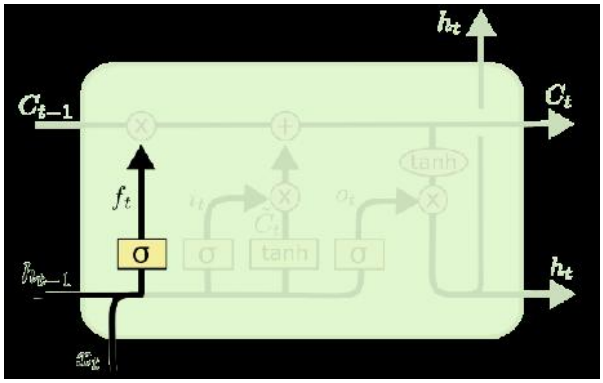
Dapat dilihat pada Gambar 2.24 bahwa informasi yang diterima dari hidden state sebelumnya hanya diteruskan menuju state berikutnya hanya dengan sedikit interaksi linear sehingga informasi yang lewat melalui suatu unit tidak akan berubah secara drastis. Hal ini lah yang sangat berpengaruh dalam pemecahan masalah *vanishing gradient problem*.

LSTM memiliki kemampuan untuk menghapus atau menambah informasi pada cell state, yang diregulasi dengan suatu struktur yang disebut *gate. Gate* merupakan suatu lintasan yang dapat membiarkan informasi lewat tanpa berubah. Gate tersusun dari suatu layer neural network sigmoid, dan operasi multiplikasi yang bersifat *pointwise*.

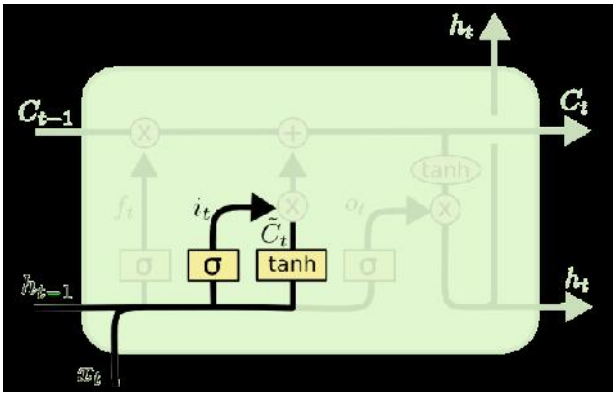
Layer sigmoid akan menghasilkan output dari antara 0 hingga 1 yang menunjukkan bahwa seberapa besar suatu informasi boleh dilewatkan begitu saja. Dimana angka 0 berarti tidak boleh ada informasi yang lewat sedangkan 1 adalah lewatkan semua informasi yang ada. LSTM unit memiliki 3 gate semacam ini dimana memiliki fungsi untuk mengontrol cell state.

Dalam LSTM unit, langkah pertama yang dilakukan adalah menentukan informasi apa yang dapat dibuang dari cell state. Keputusan ini dibuat oleh gate yang bernama *forget gate.* Forget gate akan membuat keputusan dengan memperhitungkan nilai hidden state dari hidden layer sebelumnya (h_{t-1}) dan input pada saat itu (x_t). Forget gate kemudian akan menghasilkan output antara 0 dan 1 untuk setiap cell state C_{t-1}. Nilai 1 berarti pertahankan seluruh cell state, dan nilai 0 berarti buang semua isi dari cell state. Ilustrasinya tampak pada gambar 2.26.

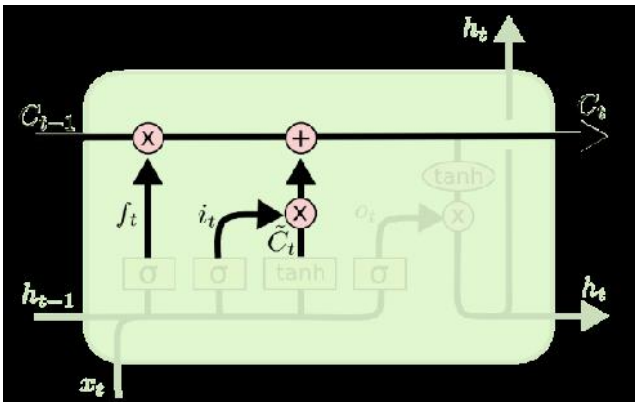
Langkah selanjutnya adalah menentukan informasi baru yang akan disimpan pada cell state. Langkah ini terdiri dari dua bagian, yang pertama adalah sebuah layer sigmoid yang bernama *input gate* yang akan menentukan informasi mana yang akan diperbaharui. Bagian kedua adalah candidate gate yang merupakan layer tanh yang akan membuat vektor kandidat baru yaitu $\hat{C}$t, yang akan



Gambar 2.26 Forget Gate



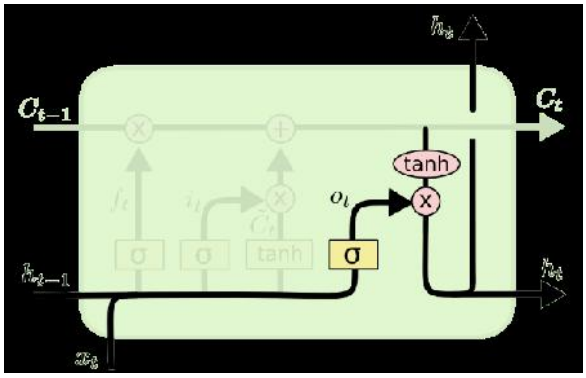
Gambar 2.27 Input Gate dan Candidate Gate



Gambar 2.28 Update Cell State

ditambahkan pada cell state. Kemudian kedua nilai ini akan digabungkan untuk menentukan mengupdate nilai dari cell state, dapat dilihat pada gambar 2.27.

Kemudian langkah berikutnya adalah mengupdate cell state lama C_{t-1} menjadi cell state yang baru C_t. Yang perlu dilakukan adalah mengalikan nilai cell



Gambar 2.29 Output Gate

state yang lama dengan f_t, dan menambah informasi baru yang didapat dari mengalikan $\hat{C}_t * i_t$ tampak pada gambar 2.28. Inilah nilai baru pada cell state C_t.

Yang terakhir, perlu ditentukan apakah yang akan dioutputkan. Output ini akan berdasarkan pada cell state yang telah difilter. Pertama-tama, kombinasi input pada *time* t dengan hidden state pada time t-1 digabungkan lalu dilewatkan ke sebuah sigmoid layer yang menentukan bagian manakah dari cell state yang dioutputkan. Setelah itu, cell state akan melewati tanh dan mengalikan cell state Ct dengan output dari sigmoid layer sehingga akan didapatkan output yang diinginkan, tampak pada ilustrasi gambar 2.29.

Secara keseluruhan, proses pada LSTM unit mulai dari input gate, forget gate, output gate, dan proses update memory serta penentuan output dapat dijelaskan dengan rumus 2.31 sampai 2.36.

$$f_t = \sigma\left(W_f[h_{t-1}, x_t] + b_f\right) \tag{2.31}$$

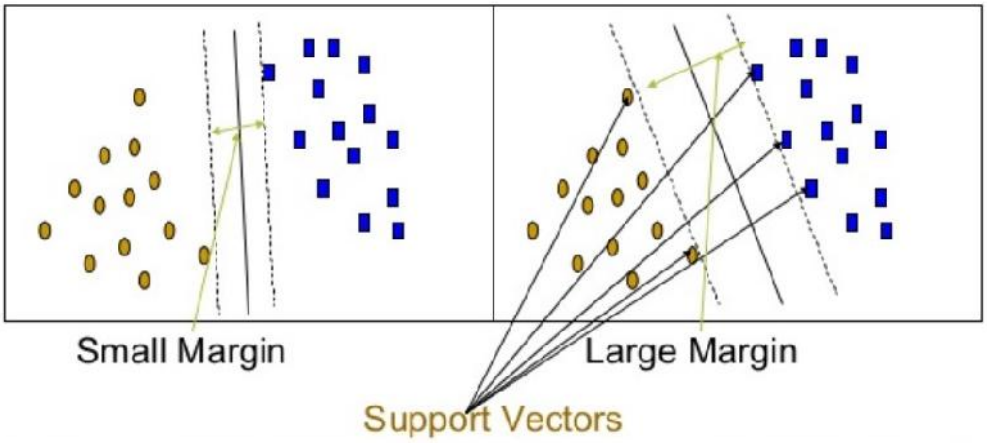
$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \tag{2.32}$$

$$\tilde{C}_t = tanh(W_C[h_{t-1}, x_t] + b_C) \tag{2.33}$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{2.34}$$

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{2.35}$$

$$h_t = o_t * \tanh(C_t) \tag{2.36}$$



Gambar 2.30 Margin pada SVM

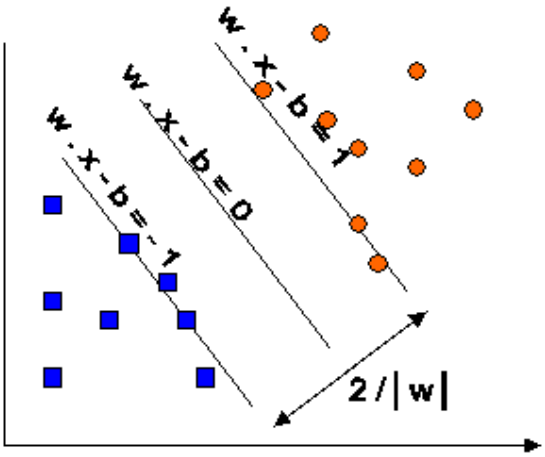
2.3.5 Support Vector Machine

Support Vector Machine (SVM) pertama kali diperkenalkan oleh Vladimir Vapnik dan Alexey Ya. Chervonenkis pada tahun 1963. Pada awalnya Support Vector Machine tidak mendapat banyak perhatian oleh para peneliti dibidang komputer. Baru setelah Vladimir Vapnik bersama Bernhard E. Boser dan Isabelle M. Guyon mempublikasikan trik kernel pada tahun 1993 kemudian SVM dapat berkembang pesat [28].

2.3.5.1 Linear SVM

Support Vector Machine awalnya termasuk dalam keluarga *linear classifier*. Tetapi sekarang SVM telah berkembang dan dapat mengklasifikasikan data yang *non-linear*. Pada subbab kali ini khusus akan dijelaskan mengenai Linear SVM. Kekhususan Support Vector Machine adalah SVM dapat meminimalkan kesalahan klasifikasi sekaligus memaksimalkan geometric margin. Sehingga SVM biasanya disebut sebagai *Large Margin Classifier*.

Cara kerja Linear Support Vector Machine adalah dengan memetakan semua data training kedalam suatu *hyper space* yang biasa disebut dengan *feature space*. Kemudian Support Vector Machine akan mencari batas atau boundary antar data dengan label yang berbeda. Setelah itu Support Vector Machine akan memperbaiki batasan atau boundary yang didapat dengan mencari Margin yang



Gambar 2.31 Paralel Hyperplane pada SVM

paling besar antara *Support Vector*. Contoh large gambar Support Vector Machine dapat dilihat pada gambar 2.29.

Jika dimisalkan terdapat sebuah dataset sebanyak n. Data tersebut akan dibentuk menjadi bentuk seperti formula 2.37. Dimana variabel y dapat bernilai satu atau negatif satu, untuk menentukan kelas target dari tiap-tiap x. Setiap x merupakan vektor dengan p-dimensi yang berisikan bilangan real. Scaling pada vektor x diperlukan agar mencegah terjadinya perbedaan besar pada nilai dalam vektor x.

$$\{(X_1, Y_1), (X_1, Y_1), (X_1, Y_1), \dots (X_n, Y_n)\} \tag{2.37}$$

$$w.x + b = 0 \tag{2.38}$$

Pada Persamaan 2.38 tampak formula hyperplane decision boundary yang memisahkan antar kelas. Dimana b adalah nilai skalar dan w adalah vektor dengan p-dimensi. Fungsi parameter b mengijinkan SVM untuk menggeser batas margin. Untuk dapat memaksimalkan margin pada SVM digunakan *pararel hyperplane* seperti pada pesamaan 2.39 dan 2.40 untuk dua kelas yang berbeda. Ilustrasinya dapat dilihat pada gambar 2.30.

$$w.x + b = 1 \tag{2.39}$$

$$w.x + b = -1 \tag{2.40}$$

Pada gambar 2.30 tampak bahwa data-data paling pinggir pada tiap kelas melekat pada *parallel hyperplane*. Sedangkan support vector adalah vektor normal yang arahnya tegak lurus dari *decision boundary* menuju titik-titik paling pinggir dari masing-masing kelas. Untuk mendapatkan margin atau jarak antara parallel hyperplane dapat menggunakan formula 2 / |W|. Sehingga untuk memaksimalkan jarak antara parallel hyperplane nilai dari |W| harus seminimal mungkin.

2.3.5.2 3.3 Non-Linear SVM

Pada tahun 1992 Bernhard E. Boser, Isabelle M. Guyon and Vladimir N. Vapnik mengusulkan cara untuk menciptakan non-linear classifier dengan mengaplikasikan trik kernel ke maximum-margin hyperplane [28]. SVM yang diusulkan tidak berubah, kecuali bahwa setiap fitur data digantikan menjadi kedekatan antara data dengan landmark menggunakan fungsi kernel non-linear. Hal ini memungkinkan SVM untuk dapat mengkalsifikasikan non-linear problem dengan melakukan *transformed feature space*.

Cara kerja non-linear SVM adalah dengan memetakan vektor x ke dalam feature space dengan dimensi yang lebih tinggi menggunakan trik kernel. Kemudian SVM mencari margin dengan jarak paling besar dalam feature space yang telah menggunakan trik kernel secara linier. Support Vector Machine mempunyai beberapa fungsi kernel yang popular dimanfaatkan. Pengguna dapat memilih fungsi kernel yang paling sesuai untuk data yang dihadapi. Berikut beberapa fungsi kernel pada Support Vector Machine :

1. Linear Kernel

$$K(X_i, X_j) = X_i^T X_j \tag{2.41}$$

2. *Polynomial* Kernel

$$K(X_i, X_j) = (X_i^T X_j + c)^d \tag{2.42}$$

3. RBF Kernel

$$K(X_i, X_j) = \exp(-\frac{||X_i, X_j||}{2\sigma^2}) \tag{2.43}$$

4. *Sigmoid* Kernel

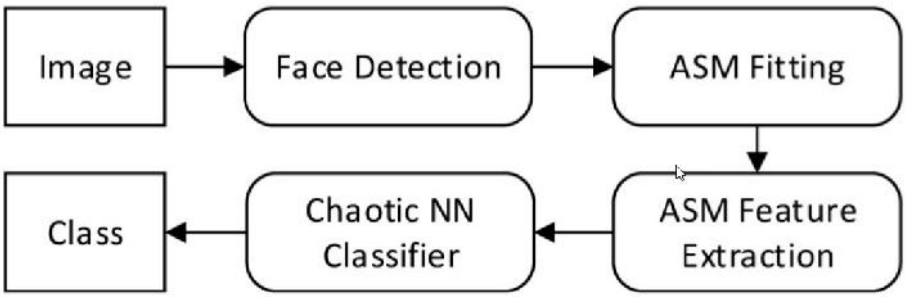
$$K(X_i, X_j) = \tanh(\gamma\, X_i^T X_j + r) \tag{3.8}$$

Dimana γ, σ dan d adalah parameter kernel. Diantara fungsi kernel diatas, RBF Kernel merupakan fungsi kernel yang paling sering dipakai untuk klasifikasi secara umum, sedangkan untuk masalah *text processing* kernel polynomial adalah yang paling sering dipakai.

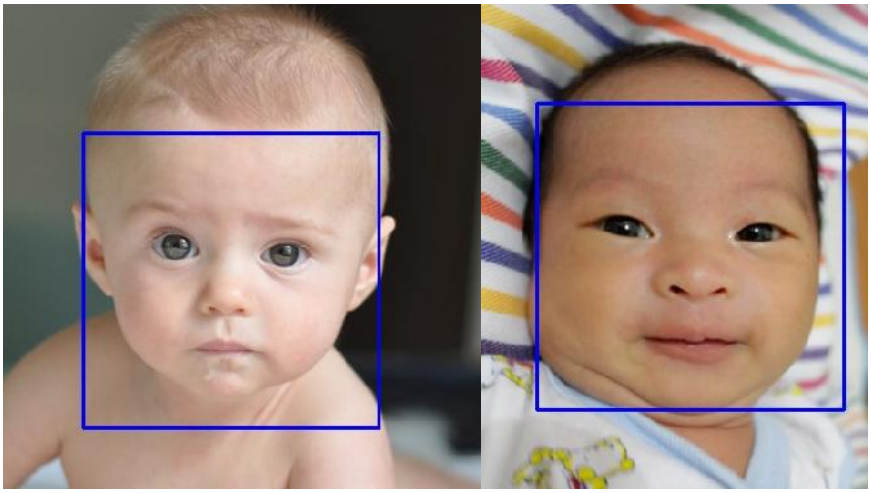
BAB 3
DETEKSI TANGIS PADA WAJAH BAYI MEMANFAATKAN FITUR GEOMETRI DAN ADACHI *CHAOTIC NEURAL NETWORK*

Pada bab ini dijelaskan mengenai riset awal disertasi ini yang berupa sistem pengenalan pola untuk mendeteksi apakah bayi menangis atau tidak berdasarkan fitur wajahnya. Tahapan sistem diawali dengan Deteksi wajah bayi dengan fitur Haar-like, pencarian *landmark* (titik-titik penting) wajah dengan Active Shape Model yang telah dilatih. Fitur yang diekstraksi kemudian diinputkan ke *Chaotic Neural Network Classifier*. Ketika pola testing bukan bayi yang tidak menangis, maka sistem akan menjadi *Chaotic*. Namun ketika pola testing adalah wajah bayi yang menangis maka sistem menjadi periodik.

Prediksi apakah bayi sedang menangis berdasarkan fitur wajah cukup menantang di bidang *computer vision*. Meskipun tangisan bayi dapat dideteksi dengan mudah berdasarkan suara, kebanyakan CCTV tidak memiliki microphone untuk merekam suara. Karena itu, penelitian ini mengolah fitur wajah saja. *Chaotic Neural Network* telah diperkenalkan sejak tahun 1989, tetapi pemanfaatannya pada bidang *computer vision* baru saja mulai populer. *Chaotic Neural Network* yang digunakan pada penelitian ini adalah Ideal Modified Adachi Neural Network (Ideal-M-AdNN). Penelitian ini masih bersifat preliminary dan baru memanfaatkan data yang jumlahnya minimal (72 data). Blok diagram penelitian ini dapat dilihat pada Gambar 3.1.



Gambar 3.1 Blok diagram Deteksi Tangis dengan Adachi Chaotic NN



Gambar 3.2 Contoh deteksi wajah bayi dengan metode Viola Jones

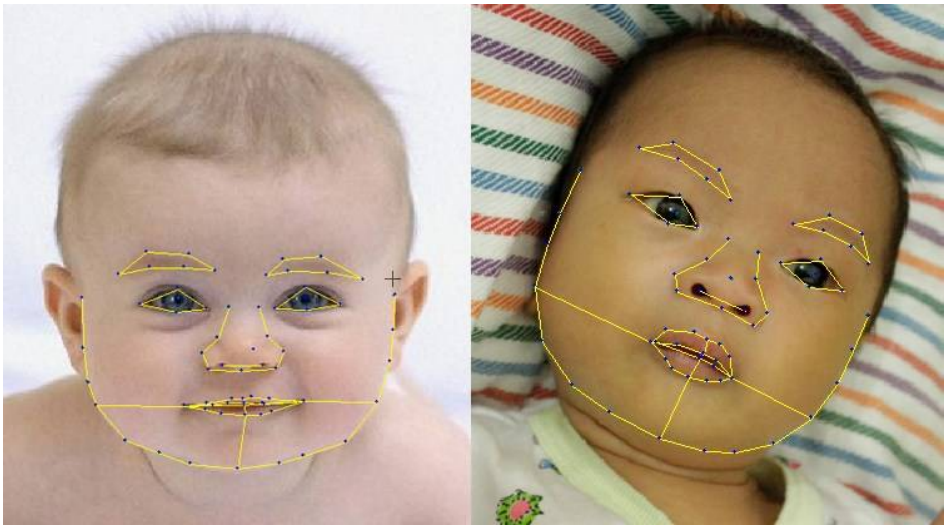
3.1 Praproses dan Ekstraksi Fitur

Untuk dapat melakukan ekstraksi fitur geometri dari wajah bayi, perlu dilakukan face detection terlebih dahulu. Metode face detection yang digunakan adalah haar like cascade dari penelitian viola jones [8]. Detail algoritma ini dapat dilihat pada subbab 2.1.1. Contoh penerapan face detection pada data penelitian ini tampak pada gambar 3.2. Dari 70 data yang ada dalam training set dan testing set algoritma ini mampu mendeteksi semua wajah bayi yang digunakan.

ASM adalah model statistic dimana model dapat berubah secara iterative untuk menyesuaikan model ke dalam dari gambar secara elastis. Metode ini menggunakan model yang fleskibel yang diperoleh dari sejumlah data sampel training. Pada penelitian ini training model ASM dilakukan dengan menggunakan 40 wajah bayi yang dilabeli manual landmarknya. Contoh citra wajah bayi yang dilabeli manual tampak pada Gambar 3.3.

Setelah dilatih, model ASM dapat diminta untuk melakukan fitting pada citra wajah baru. Contoh model ASM melakukan fitting pada wajah baru tampak pada Gambar 3.4. ASM memulai pencarian landmark dari menempelkan rata-rata bentuk dari data train yang diatur posisi dan ukuran sesuai dengan area wajah yang ditemukan oleh tahap face detection. Kemudian secara iteratif model diubah agar memiliki error terkecil. Penjelasan lebih detail tentang ASM dapat dilihat pada subbab 2.1.2.

Gambar 3.3 Pemberian Landmark Manual Pada Wajah Bayi



Gambar 3.4 Proses Fitting ASM pada wajah baru

Setelah landmark dari wajah bayi di temukan ekstraksi fitur geometri dari titik-titik wajah dapat dilakukan dengan mudah. Bayi yang menangis memiliki perbedaan mulut, dagu, mata, dan jarak alis. Titik-titik landmark yang dihasilkan oleh ASM diubah menjadi jarak ke tengah mulut, tengah mata, dan tengah hidung. Dari proses ini didapatkan 45 fitur jarak yang diambil dengan formula eucledian distance. Ilustrasi ke 45 fitur jarak ini dapat dilihat pada gambar 3.5.

Gambar 3.5 Fitur Geometri untuk Deteksi Tangis Bayi

3.2 *Chaotic NN Classifier*

Dalam penelitian ini, Chaotic NN dilatih dengan menggunakan wajah yang sedang menangis saja. Jika pola testing adalah fitur dari wajah yang menangis, maka sistem akan mengeluarkan output periodik tetapi ketika pola testing bukan fitur dari wajah bayi yang menangis, system akan mengeluarkan output yang chaotic. Algoritma training dan testing dilakukan berdasarkan [22] dan dimodifikasi berdasarkan [23]. Penjelasan Chaotic NN dapat dilihat pada subbab 2.3.1.

Karena terdapat 45 fitur dari ASM, dalam penelitian ini dibuat sebuah Chaotic NNclassifier dengan 45 recurrent neuron. Chaotic NNdilatih dengan fitur wajah menangis saja karena karakteristik Chaotic NNyang menghasilkan sinyal periodik ketika pola input dekat dengan pola yang dilatih. Ketika Chaotic NN menerima input fitur wajah bayi yang tidak sedang menangis, akan muncul output sinyal chaotic.

Tujuan dari penelitian ini adalah untuk mendeteksi apakah input adalah wajah bayi yang menangis atau tidak, sehingga Chaotic NN tidak dilatih dengan ekspresi lain. Sistem ini dirancang demikian, karena input video akan digunakan untuk pengembangan selanjutnya. Deteksi dilakukan dengan menjalankan Chaotic NN. Ketika sistem menghasilkan sinyal periodic, maka wajah bayi yang menangis berhasil dideteksi.

3.3 Hasil Penelitian

Chaotic NN classifier diuji dengan 30 wajah (10 wajah bayi menangis, 20 wajah normal atau tersenyum). Hasilnya cukup memuaskan, dimana dari 10 wajah menangis semuanya terdeteksi dengan benar, sedang dari 20 wajah normal atau tersenyum, 18 wajah terdeteksi tidak menangis, dan hanya dua yang salah terdeteksi sebagai menangis. Confusion matriks pada tabel 3.1. Performansi CNN classifier dibandingkan dengan SVM dan MLP tampak pada tabel 3.2. Untuk SVM, digunakan kernel Radial Basis Function dan optimisasi SMO. Untuk MLP digunakan backpropagation learning dan fungsi aktivasi logistic.

Tabel 3.1 Matriks confusion untuk Ideal-M-AdNN Classifier

		Prediksi	
		Menangis	Tidak Menangis
Kelas Real	Menangis	10	0
	Tidak Menangis	2	18

Tabel 3.2 Perbandingan Akurasi dengan Classifier Lain

	Training Set		
Classifier	**Crying**	**Non-Crying**	**Accuracy**
SVM (SMO)	20	20	100%
MLP (Backpropagation)	20	20	97%
CNN (Ideal-M-AdNN)	20	0	93%

3.4 *Concluding Remarks*

Penelitian ini menghasilkan akurasi klasifikasi hingga 93%. Deteksi tangis pada wajah bayi dilakukan dengan mengkombinasikan fitur wajah geometris dengan ASM dikombinasikan dengan Chaotic NN. Akurasi yang lebih rendah karena pada sistem chaotic NN tidak dilatih dengan wajah bayi yang tidak menangis.

Halaman ini sengaja dikosongkan

BAB 4
KLASIFIKASI NYERI PADA WAJAH BAYI MEMANFAATKAN FITUR TEKSTUR DAN FITUR GEOMETRI DAN SKENARIO KLASIFIKASI DUA TAHAP

Pada bagian ini diuraikan dataset, fitur-fitur yang dipakai dan scenario machine learning yang diterapkan dalam sistem klasifikasi nyeri pada wajah bayi dengan dua tahap klasifikasi yang diawali dengan deteksi tangis. Penelitian ini juga mengeksplorasi penggunaan fitur geometri dan fitur tekstur untuk deteksi tangis dan nyeri pada wajah bayi.

4.1 Teori Nyeri

Sejak dikenalkan oleh Ekman pada [29] Facial Action Coding System (FACS) yang berisi puluhan koding aksi ini banyak sekali digunakan pada riset-riset terkait ekspresi wajah. FACS adalah pemetaan pergerakan otot-otot wajah seperti "alis bagian dalam menurun", "hidung berkerut" atau "bibir bagian atas naik". Setiap FACS diberikan kode Action Unit (AU) untuk lebih memudahkan pembacaan dan setiap FACS memiliki nilai dari intensitas 1-5. Banyak peneliti yang mencoba mengenali FACS secara otomatis melalui teknik-teknik komputer vision.

Pada penelitian [30] Prkachin mempublikasikan sebuah formula yang merupakan kumpulan FACS yang menyusun ekspresi nyeri. Menurut [30], AU dan FACS yang terkait nyeri adalah:

- Alis menurun (AU 4)
- Pipi naik (AU 6)
- Kelopak mata menutup (AU 7)
- Hidung berkerut (AU 9)
- Bibit atas naik(AU 10)
- Pojok bibir tertarik (AU 12)
- Bibir tertarik secara horizontal (AU 20)

Gambar 4.1 Contoh AU pada wajah yang mengalami nyeri.

- Bibir atas-bawah terpisah (AU 25)
- Rahang turun (AU 26)
- Mulut tertarik (AU 27)
- Mata menutup (AU 43)

Pada penelitian yang lebih baru, [31] Prkachin dan Solomon mengkodekan ulang AU untuk nyeri menjadi lebih sederhana seperti tampak pada Persamaan (4.1). Persamaan ini berarti nyeri adalah penjumlahan AU4, AU6 atau AU7 (pilih yang lebih tinggi), AU9 atau AU10 (pilih yang lebih tinggi) and AU43. Visualisasi AU pada wajah bayi yang mengalami nyeri dapat dilihat pada gambar 4.1.

$$Pain = AU4 + (AU6||AU7) + (AU9||AU10) + AU43 \qquad (4.1)$$

4.2 Dataset

Pada penelitian sebelumnya [3], Hanindito merekam ekspresi wajah 23 bayi sebelum dan sesudah menjalani operasi sehingga menghasilkan 46 video. Kemudian ditambahkan 5 bayi lagi yang menjalani perekaman yang sama, sehingga didapatkan total 56 video ekspresi wajah bayi sebelum dan sesudah menjalani operasi. Selain merekam video, Hanindito juga melengkapi ke 56 video dengan data score FLACC dan pengukuran hormon cortisol, sehingga dari masing-masing video telah diketahui derajat nyeri yang dialami bayi dalam video.

Dari 56 video yang didapatkan dari penelitian Hanindito, ternyata ada 12 video yang tidak dapat diproses karena tidak terdapat citra wajah frontal yang dapat diekstrak. Hal ini disebabkan karena petugas pengambil videonya terlalu sering

(a) (b)

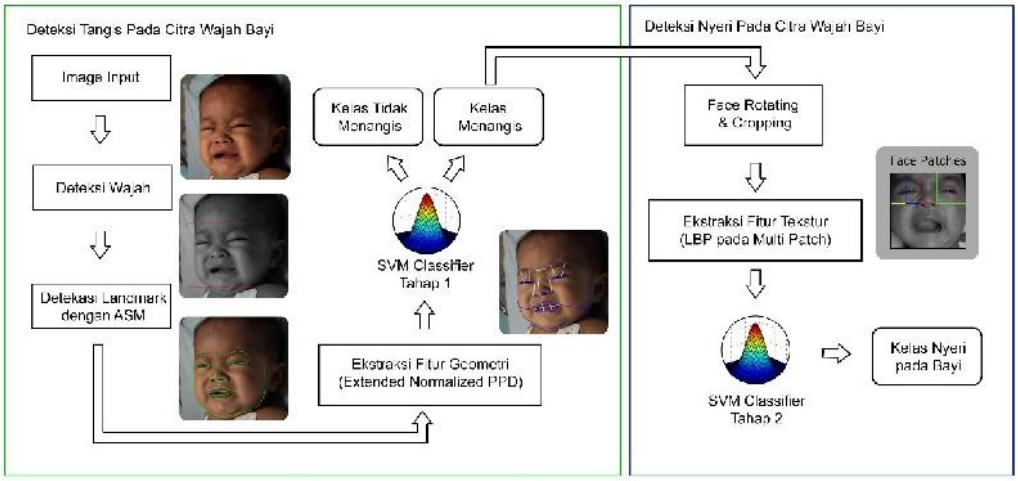
Gambar 4.2 a. Proses seleksi frame. b. Contoh citra dalam dataset.

zoom in dan zoom out serta merotasi kamera. Dalam penelitian ini diekstrak 3 frame dari 44 video yang tersisa sehingga menghasilkan 132 citra wajah bayi. Proses ekstraksi frame dari video dilakukan manual dengan mencari saat puncak bayi menangis. Contoh proses pemilihan frame dapat dilihat pada gambar 4.2a, frame yang dipilih ditandai dengan kotak merah.

Dari 132 citra wajah bayi yang digunakan, terdapat 54 yang menangis dan 78 yang tidak menangis. Terdapat 30 citra wajah bayi yang mengalami nyeri berat, 37 citra yang mengalami nyeri ringan sampai sedang, dan 65 citra tidak mengalami nyeri. Persebaran data ini dapat diamati pada tabel 4.1. Sedangkan contoh-contoh citra wajah bayi dalam dataset dapat dilihat pada gambar 4.2b.

Tabel 4.1 Persebaran kelas dalam dataset.

Kelas	**Menangis**	**Tidak Menangis**	**Total**
Nyeri Berat	30	0	30
Nyeri Ringan sampai Sedang	19	18	37
Tidak Nyeri	5	60	65
Total:	54	78	132



Gambar 4.3 Arsitektur Sistem [50]

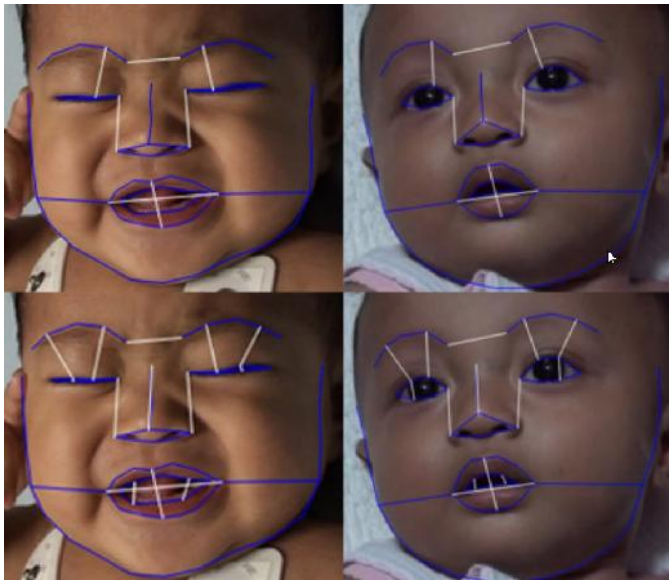
4.3 Arsitektur Sistem

Dalam penelitian ini dirancang sebuah sistem klasifikasi dua tahap yang memanfaatkan fitur geometri dan fitur tekstur. Pertama-tama sistem tetap melakukan praproses yang terdiri dari *face detection* dan *face landmark detection*, untuk mendapatkan area dan titik-titik penting wajah. Kemudian dilanjutkan dengan proses ekstraksi fitur geometri yang memanfaatkan *landmark* wajah dan klasifikasi apakah bayi menangis. Peneliti berhipotesa bahwa tangis akan lebih berpengaruh pada geometri wajah dari pada teksturnya, karena itu fitur geometri yang berupa jarak-jarak antara titik-titik wajah akan lebih mudah menentukan apakah bayi sedang menangis atau tidak. Sedangkan untuk tahap klasifikasi derajat nyeri, peneliti berhipotesa bahwa tekstur pada area-area tertentu wajah lebih menunjukkan nyeri dari pada geometrinya. Karena itu pada klasifikasi tahap kedua digunakan fitur tekstur untuk mendeteksi derajat nyeri pada wajah bayi. Gambaran arsitektur ini dapat dilihat pada Gambar 4.3.

4.4 Ekstraksi Fitur Geometri

Pada penelitian ini diekstrak dua set fitur geometri berupa jarak-jarak antara titik-titik landmark yang ditemukan oleh ASM:

1. **NPPD**: Normalized Point Pair Distance adalah jarak-jarak seperti pada penelitian schiavenato [16] namun dinormalisasi. Normalisasi dilakukan dengan cara dibagi dengan lebar wajah untuk jarak-jarak horizontal dan



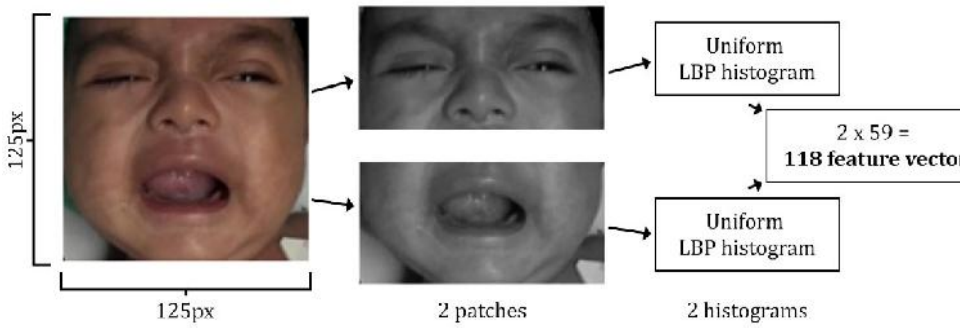
Gambar 4.4 Garis-garis putih pada baris pertama menunjukkan fitur NPPN, dan pada baris kedua menunjukkan fitur ENPPD.

dibagi dengan tinggi wajah untuk jarak-jarak vertikal. Disini peneliti menggunakan eucledian distance untuk mengukur jarak antara dua buah titik koordinat kartesian. Fitur ini terdiri dari 7 buah jarak yang digambarkan dengan garis-garis putih pada Gambar 4.4 baris pertama.

2. **ENPPD**: Extended Normalized Point Pair Distance adalah ekstensi dari NPPD yang ditambahkan demi memperoleh lebih banyak fitur dan informasi. Disini peneliti menambahkan 7 jarak tambahan yang bedanya dapat dilihat pada Gambar 4.4 baris kedua. Tambahan ini untuk menambah informasi terkait AU43(Mata menutup), dan AU25(Bibir terpisah) yang belum tercover oleh PPD pada penelitian Schiavenato.

4.5 Ekstraksi Fitur Tekstur

Tidak semua FACS dapat di tangkap dengan baik oleh fitur geometri, contohnya FACS hidung berkerut dan pipi yang naik tidak sepenuhnya dapat di ekstrak dengan fitur geometri. Karena perubahan yang terjadi pada wajah lebih ke arah kerutan maka ada penelitian ini ditambahkan fitur tekstur menggunakan *Local Binary Pattern* (LBP) seperti yang telah dijelaskan pada Subbab 2.2.2.



Gambar 4.5 Proses TBLBP81

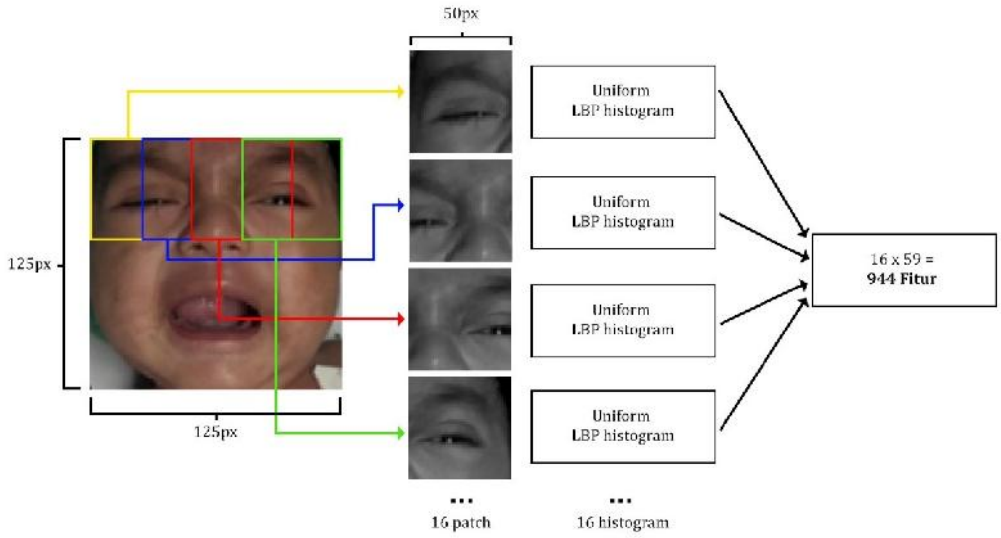
Menurut [20] LBP adalah tekstur *descriptor* local, dimana LBP lebih cocok diaplikasikan pada potongan kecil gambar daripada keseluruhan gambar. Oleh karena itu pada penelitian ini dilakukan percobaan dengan dua set fitur LBP terkait jumlah patch yang dipotongkan pada citra wajah bayi. Dua set fitur LBP yang digunakan adalah:

1. **TBLBP81**: Top Bottom LBP dengan N=8 dan R=1, disini peneliti membagi citra wajah menjadi dua *patch* (potongan gambar) secara horisontal (atas dan bawah) kemudian menjalankan *uniform LBP histogram* untuk masing masing *patch.* Dengan ini dihasilkan 118 fitur (59 x 2) untuk TBLBP81. Proses pengambilan fitur ini dapat dilihat pada Gambar 4.5.
2. **MPLBP81**: Multi patch LBP dengan N=8 dan R=1 neighbor, disini peneliti membagi citra wajah menjadi enam belas *patch* yang saling overlap. Untuk lebih jelas tentang MPLBP81 dapat dilihat pada Gambar 4.6. Cara ini menghasilkan 944 fitur (59*16).

4.6 Klasifikasi dengan SVM dan Variasi Kernelnya

Support Vector Machine atau SVM adalah salah satu teknik *machine learning* yang sering digunakan untuk masalah klasifikasi. SVM dikenalkan oleh Cortes & Vapnik sejak tahun 1995 [32], dan trainingnya menggunakan Sequential Minimal Optimization (SMO) dikenalkan oleh Platt pada tahun 1998 [33]. Sejak itu SVM manrik banyak perhatian dari kalangan peneliti *machine learning*.

Sebuah model SVM dikenal dengan sebutan *large margin classifier*, yang maksudnya SVM memetakan data sebagai titik-titk dalam ruang dimensi fitur yang



Gambar 4.6 Proses MPLBP81

baru kemudian mencari garis pemisah antar kelas dengan *gap* (ruang) selebar mungkin. Dimensi fitur yang baru didapatkan dengan menerapkan *kernel tricks*. Ada beberapa kernel yang popular digunakan dalam SVM yaitu: Linear, Sigmoid, RBF (*Radial Basis Function*), dan Polinomial.

4.7 *Metric* dan Skenario Ujicoba

Dalam penelitian ini digunakan k-fold cross validation menggunakan nilai k=10 yang berarti dataset dibagi menjadi 10 segmen kemudian melakukan training testing 10 kali dengan dengan menggunakan segmen yang berbeda-beda. Untuk mengukur performa sistem digunakan F1 Score dan Area Under the ROC curve (AUC), dua metric ini lebih reliabel dari akurasi karena mempertimbangkan lebih banyak hal.

F1 Score adalah harmonic mean dari precission dan recall yang dihitung dengan Persamaan 4.2. Untuk mengukur AUC, langkah pertama adalah membuat kurva Receiver Operating Characteristic (ROC). ROC curve dibuat dengan cara membandingkan *true positive rate* (TPR) dengan *false positive rate* (FPR) pada *threshold* yang bervariasi. TPR biasa disebut sensitivity, atau recall dalam machine learning, sedangkan FPR biasa disebut fall-out atau 1-specificity. ROC curve sering

disebut sebagai sensitivity sebagai fungsi dari fall-out dan AUC adalah luas area biawah kurva ini.

$$F1\ Score = \frac{2PR}{P+R} \tag{4.2}$$

4.8 Ujicoba Deteksi Tangis

Hasil ujicoba deteksi tangis dengan berbagai kernel dan fitur dapat dilihat pada Tabel 4.2. Dari hasil ini tampak bahwa fitur ENPPD menghasilkan AUC dan F1 Score terbaik. Bahkan kombinasi antara ENPPD dan MPLBP81 tidak meningkatkan performa, hal ini menunjukkan bahwa tekstur kurang berpengaruh dalam deteksi tangis. Kernel terbaik untuk SVM dalam percobaan ini adalah RBF yang performanya paling baik bahkan di hampir semua fitur.

Tabel 4.2 Hasil Deteksi Tangis

	NPPD		**ENPPD**		**TBLBP81**		**MPLBP81**		**ENPPD+ MPLBP81**	
SVM Kernels	**AUC**	**F1**	**AUC**	**F1**	**AUC**	**F1**	**AUC**	**F1**	**AUC**	**F1**
RBF	0.910	0.870	**0.983**	**0.961**	0.873	0.894	0.937	0.907	0.951	0.937
Polynomial	0.895	0.841	0.986	0.949	0.894	0.868	0.889	0.889	0.977	0.956
Linear	0.925	0.873	0.990	0.942	0.924	0.868	0.993	0.85	0.959	0.929
Sigmoid	0.404	0.743	0.869	0.783	0.548	0.733	0.503	0.729	0.949	0.922

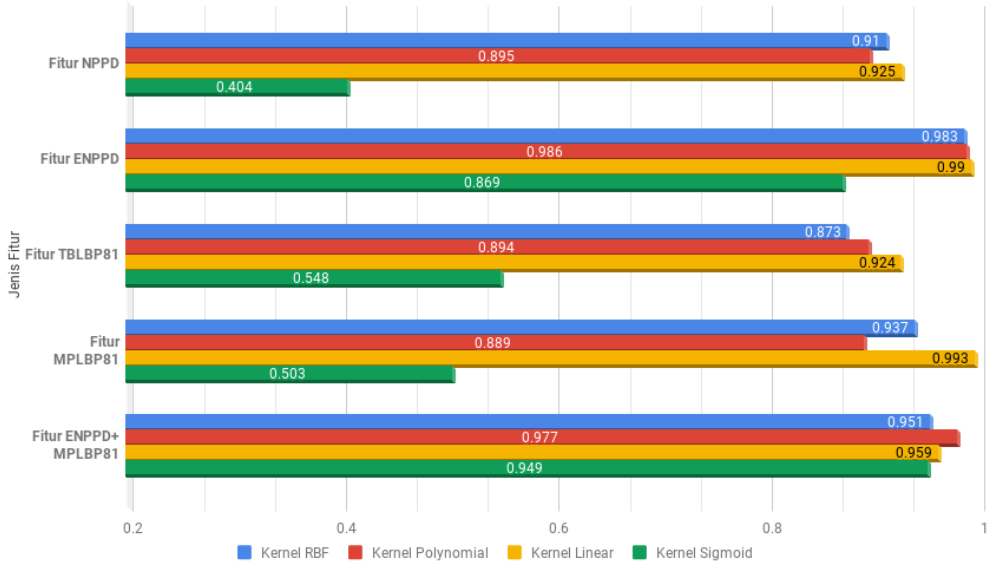
4.9 Ujicoba Klasifikasi Nyeri

Untuk ujicoba klasifikasi nyeri dilakukan dua skenario, yang pertama adalah kalsifikasi satu tahap dengan mengumpankan fitur langsung ke dalam *classifier*. Skenario kedua adalah dengan melakukan klasifikasi dua tahap dengan diawali klasifikasi tangis dahulu, hanya data yang terdeteksi menangis yang diumpankan ke *classifier* kedua.

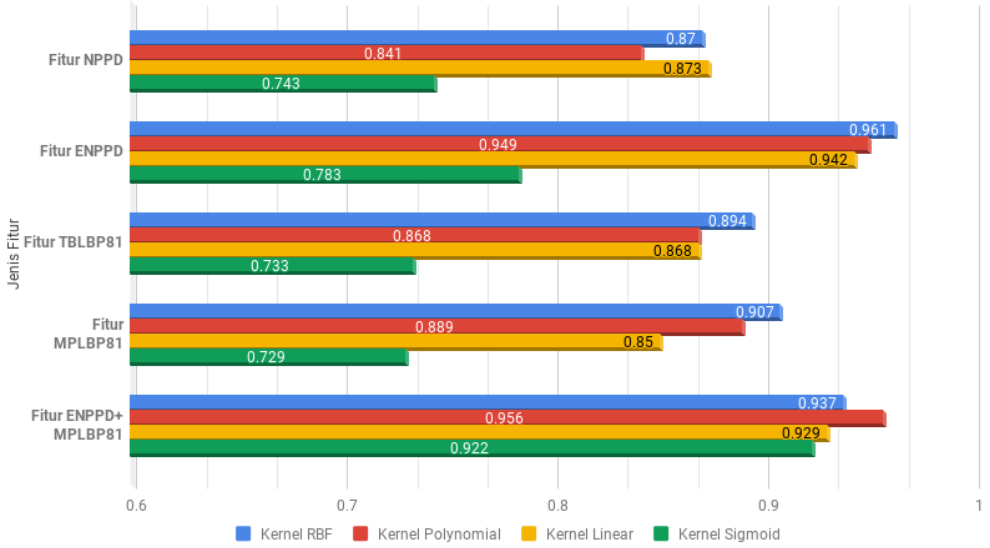
Hasil dari skenario satu tahap dapat dilihat pada Tabel 4.3. Pada percobaan ini semua menggunakan SVM dengan RBF kernel. Tampak fitur terbaik adalah gabungan fitur geometrid dan fitur tekstur.

Tabel 4.3 Hasil Deteksi Nyeri untuk Satu Tahap

Features	**CA**	**Average F1**	**AUC**
NPPD	0.847	0.870	0.92
ENDPPD	0.834	0.871	0.96
TBLBP81	0.841	0.813	0.96



Gambar 4.7 Visualisasi Nilai Area Under ROC Curve Untuk Deteksi Tangis Memanfaatkan Variasi Fitur dan Kernel SVM



Gambar 4.8 Grafik Nilai F1 Score Untuk Deteksi Tangis Memanfaatkan Variasi Fitur dan Kernel SVM Visualisasi

MPLBP81	0.887	0.853	0.94
ENPPD+TBLBP81	0.857	0.853	0.98
ENPPD+MPLBP81	**0.887**	**0.871**	**0.97**

Hasil dari skenario dua tahap dapat dilihat pada Tabel 4.4. Pada percobaan ini semua *classifier* masih menggunakan SVM dengan RBF kernel. Tampak fitur

Gambar 4.9 Contoh Ujicoba dari input, intermediate result, sampai hasil saat ujicoba

terbaik adalah *classifier* dengan fitur tekstur saja. Contoh proses keseluruhan dari input, intermediate result, sampai hasilnya dapat dilihat pada Gambar 4.9.

Tabel 4.4 Hasil Deteksi Nyeri untuk Dua Tahap

Feature	**CA**	**Average F1**	**AUC**
TBLBP81	0.870	0.903	0.91
MPLBP81	**0.923**	**0.952**	**0.91**
ENPPD+TBLBP81	0.867	0.903	0.90
ENPPD+MPLBP81	0.887	0.921	0.91

4.10 *Concluding Remarks*

Dalam penelitian ini dikenalkan pendekatan baru untuk dapat mengklasifikasikan nyeri pada wajah bayi yang diawali dengan deteksi tangis yang memanfaatkan kombinasi fitur geometri dan tekstur. Dalam penelitian ini telah ditunjukkan bahwa skenario dua tahap dengan mendeteksi tangis terlebih dahulu dapat meningkatkan F1 Score dan AUC.

Fitur Geometri memiliki performa yang lebih baik dalam deteksi tangis karena saat menangis titik-titik penting pada wajah bayi secara geometri memang berpindah. Sedangkan untuk klasifikasi nyeri fitur tekstur memiliki performa yang lebih baik karena kerutan di area hidung, mata, dan pipi tampak lebih ekstrim saat bayi mengalami nyeri berat.

Halaman ini sengaja dikosongkan

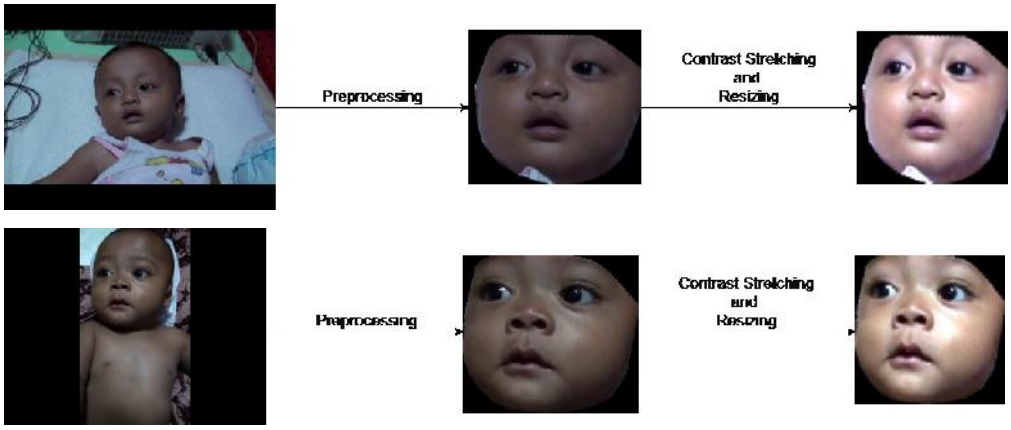
BAB 5
PEMANFAATAN DCNN AUTOENCODER DAN LSTM UNTUK KLASIFIKASI TANGIS DAN NYERI PADA VIDEO WAJAH BAYI

Pada bab ini dijelaskan tentang penelitian untuk klasifikasi nyeri dan tangis pada bayi berdasarkan video ekspresi wajah memanfaatkan *DCNN Autoencoder* dan *Long Short Term Memmory Network*. Beberapa penelitian terdahulu sudah memanfaatkan citra wajah bayi dan menggunakan *handcrafted feature* dengan bantuan *machine learning* namun hingga saat ini masih belum ada penelitian yang mencoba mengklasifikasikan nyeri pada ekspresi wajah bayi yang menggunakan deep learning. Dengan memanfaatkan deep learning diharapkan system akan mendapatkan hasil yang lebih baik karena tidak perlu lagi mencari fitur secara manual.

Meskipun memberikan hasil yang cukup memuaskan, pada penelitian-penelitian sebelumnya masih bergantung pada *handcrafted feature* yang didefinisikan manual oleh penelitinya. Hal ini sering menjadi kendala karena proses ekstraksi fitur yang dilakukan juga tidak 100% berhasil dengan baik. Karena itu dalam penelitian ini diterapkan ekstraksi fitur secara *feature learning* memanfaatkan *autoencoder*.

5.1 Dataset

Infant Facial Pain Level Dataset (IFPaLD) merupakan dataset yang berhasil didapatkan dalam penelitian ini. Video yang menjadi dasar pembuatan dataset IFPaLD diambil dari 28 bayi 16 laki-laki dan 12 perempuan sebelum dan setelah menjalani prosedur operasi. Video-video ini diambil dalam penelitian Hanindito dkk. pada tahun 2013 [3]. Dalam penelitian tersebut, setiap bayi dalam video diukur FLACC dan hormon cortisol-nya untuk menentukan tingkat nyeri yang dialami bayi. Tingkat nyeri dibedakan menjadi 3 yaitu: nyeri berat (1), nyeri ringan sampai sedang (2) dan tidak nyeri (3).



Gambar 5.1 Proses pembentukan dataset dari video.

Citra-citra pada dataset ini berformat jpg, diambil dari video dengan frekuensi 3 frame per detik. Total dataset terdiri dari 15.105 citra dan dibagi menjadi 11.800 data training dan 3.305 data testing. Dalam melakukan ekstraksi citra wajah bayi dilakukan face detection dan face landmark detection untuk mendapatkan ROI wajah bayi dan melakukan rotasi sesuai kemiringan mata. Setelah itu dilakukan *cropping* dan *contrast streaching* untuk mendapatkan citra-citra yang bersih dan memili kontras yang setara.

IFPaLD tidak memiliki timestep yang terbatas, panjang sekuen pada dataset ini adalah sepanjang frame video wajah bayi yang diekstraksi. Pengguna dataset ini dapat menentukan timestep atau panjang sekuen yang akan dipakai dan disesuaikan dengan arsitektur RNN yang akan digunakan. Dalam dataset disediakan citra yang telah di crop dan yang belum diproses. Contoh proses ekstraksi dapat dilihat pada gambar 5.1. Dataset ini juga dapat di download pada laman http://dataset.stts.edu/IFPaLD.

5.2 Convolutional Neural Network

Artificial Neural Network (ANN) merupakan sebuah model jaringan yang menerima input berbentuk *vector* satu dimensi yang mentransformasikan input menjadi output dengan cara mempropagasikan hasil *dot product* yang diaktivasi pada *layer-layer* yang saling berhubungan [24]. Setiap neuron pada sebuah *layer* berhubungan penuh dengan semua neuron di layer sebelumnya, dan memiliki bobot independen, arsitektur ini disebut *multi layer perceptron* (MLP) [34]. Layer ini

disebut sebagai *fully-connected layer* dan dilatih dengan algoritme *Backpropagation* [35].

Pada pemrosesan data yang berukuran besar, *fully-connected layer* dapat menyebabkan beberapa masalah. Sebagai contoh, citra-citra pada *dataset* CIFAR-10 [36] memiliki ukuran 32x32x3, apabila data ini diumpankan pada sebuah MLP maka satu neuron pada *layer* pertama akan memiliki setidaknya 32x32x3=3072 bobot. Angka ini akan meningkat dengan cepat bila kita menggunakan banyak neuron, banyak *laye*r, atau input dengan dimensi yang lebih besar. Dengan banyaknya jumlah bobot yang harus dilatih oleh ANN, akan menyebabkan terjadinya *overfitting* [24].

Selain jumlah bobot yang sangat besar *fully-connected layer* juga tidak memperhitungkan struktur spasial dari sebuah citra. Sebagai contoh, input *pixel* pada posisi yang jauh dan dekat akan memiliki kedudukan yang sama. *Convolutional Neural Network* mengatasi dua masalah ini dengan menerapkan weight sharing dan mekanisme konvolusi *filter*. Sebuah jaringan yang dapat memperhitungkan struktur spasial dari citra akan memiliki performa yang lebih baik untuk menangani input berupa citra [24].

Ide CNN sebenarnya sudah muncul pada tahun 1970, namun paper yang merupakan dasar dari CNN modern baru muncul pada tahun 1998. Paper tersebut dibuat oleh Yann Lecun, Leon Bottou, Yoshua Bengio, dan Patrick Haffne dengan judul "*Gradient-based learning applied to document recognition"* [37].

CNN tidak langsung mendapatkan momentum untuk menjadi *classifier* terbaik karena keterbatasan teknologi. Baru pada tahun 2012, CNN mendapatkan ketenaran karena Alex Krizhevsky memenangkan tantangan *ImageNet* [38] dengan akurasi yang selisih jauh dari tahun sebelumnya. Saat ini CNN sendiri sudah banyak digunakan untuk klasifikasi pada berbagai bidang karena tingkat kesuksesannya yang cukup besar. Pada penelitian-penelitian terkini CNN telah dimanfaatkan untuk: klasifikasi sepeda motor [39], deteksi pejalan kaki [40], dan klasifikasi umur dan jenis kelamin manusia [41].

Terdapat 3 tipe *layer* untuk membuat arsitektur ConvNet yaitu *Convolutional Layer* (CONV), *Pooling Laye*r (POOL), dan *Fully Connected* (FC) [42]. Kita dapat menyusun layer-layer tersebut menjadi satu untuk membentuk

suatu arsitektur CNN. Contoh untuk pembuatan sebuah CNN yang bertujuan untuk melakukan *image classification* pada *dataset* CIFAR-10 arsitekturnya dapat berbentuk: INPUT-CONV-POOL-CONV-POOL-FC-FC.

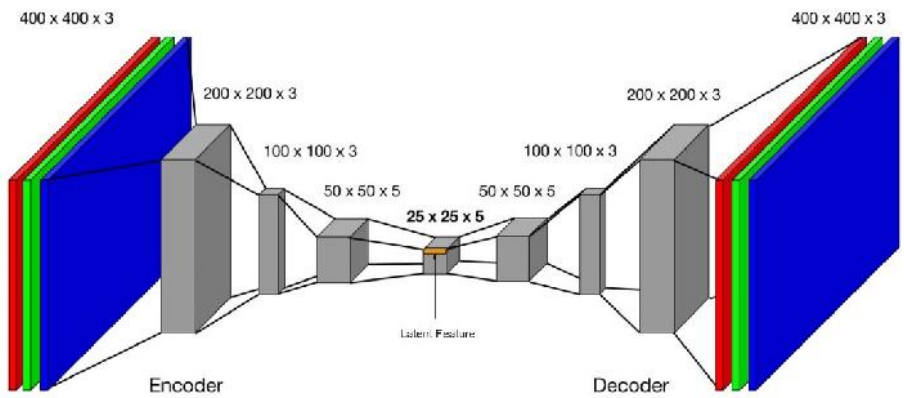
5.3 *Convolutional Autoencoder*

Selain CNN, terdapat pula suatu *network* yang cukup terkenal di mana *network* ini dilatih dengan metode *supervised learning* namun tidak membutuhkan kelas dari data sebagai target. *Network* ini terkenal dengan nama *autoencoder* atau *auto-associator*. *Autoencoder* adalah model *neural network* yang memiliki input dan output yang sama, ilustrasinya dapat dilihat pada Gambar. 5.2 [43]. *Autoencoder* memiliki *weight* yang sama antara *encoder* dan *decoder*-nya yang biasa disebut dengan *tied weight. Autoencoder* berusaha untuk mempelajari data input dan berusaha untuk merekonstruksi kembali input tersebut namun dengan informasi yang lebih sedikit. Informasi ini biasa disebut sebagai *latent feature.*

Kegunaan *autoencoder* adalah untuk mengurangi dimensi dari fitur yang didapat dari suatu input atau bisa disebut dengan *Dimensionality Reduction* [44]. Cara memanfaatkan *autoencoder* adalah dengan memotong network pada bagian *encoder*-nya kemudian *latent feature* yang dihasilkan diumpankan ke classifier. *Autoencoder* juga dapat digunakan untuk *denoising* atau menghilangkan *noise* pada suatu data input. *Autoencoder* yang menghilangkan *noise* disebut juga dengan *denoising autoencoder* [45].

Metode paling umum yang digunakan untuk masalah *Dimensionality Reduction* adalah *Principal Component Analysis* (PCA) [46]. PCA cukup unggul sebagai metode reduksi dimensi karena sederhana untuk diimplementasikan dan mampu mengeliminasi korelasi antar variabel input. Akan tetapi, PCA hanya mereduksi data secara linear, dan tidak memperhatikan hubungan antar data secara spasial.

Sama seperti PCA, sebuah *neural network autoencoder* konvensional yang menerima input berbentuk citra, juga tidak dapat memperhitungkan informasi spasial dari data input mengingat bahwa bentuk *layer* dari *autoencoder* adalah *flat layer*. Maka dari itu, terciptalah ide yang bernama *convolutional autoencoder* (CAE). CAE memiliki konsep yang sama dengan CNN dan memiliki tujuan dan



Gambar 5.2 Contoh Arsitektur Convolutional Autoencoder

pola pikir yang sama dengan *autoencoder*. CAE juga memiliki komponen yang sama dengan CNN pada umumnya yaitu *Conv Layer*, *Activation Layer*, dan *Pooling Layer*.

Pada umumnya, CAE tidak menggunakan *Fully Connected Layer* dikarenakan tujuan dari CAE adalah mengekstraksi fitur penting dari citra lalu merekonstruksi nya kembali. Konsep *tied weights* pada autoencoder juga diterapkan pada CAE dimana filter untuk bagian encoder sama dengan bagian decoder hanya saja urutanya terbalik [47].

Jadi pada penelitian ini CAE berfungsi sebagai *feature learner* atau ekstraksi fitur otomatis. Fitur-fitur yang didapat kemudian diklasifikasikan menggunakan *Recurrent Neural Network* (RNN) karena berbentuk video atau memiliki urutan waktu. Pada bab berikutnya akan dijelaskan tentang arsitektur RNN yang dipakai yakni LSTM.

5.4 Long Short Term Memory Network

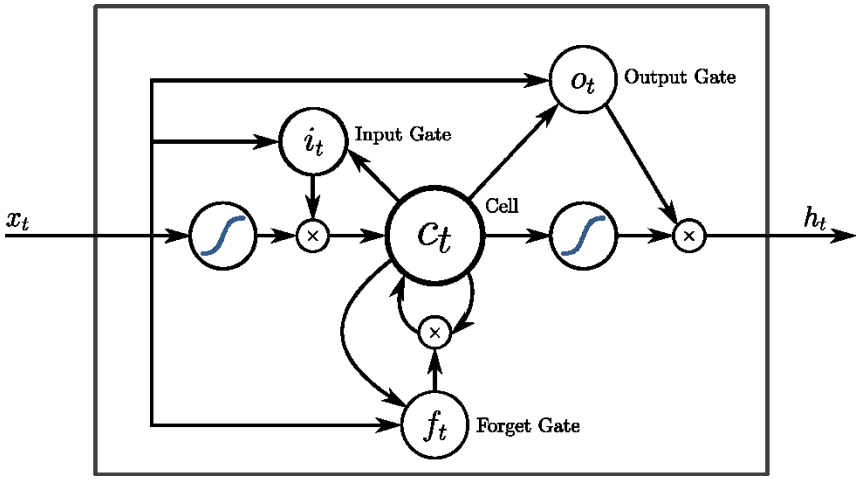
Neural network juga dapat digunakan untuk memproses data sekuensial. Sebagai contoh, *neural network* bisa memprediksi kata yang akan muncul berdasar kemunculan kata-kata sebelumnya. *Recurrent Neural Network* (RNN) adalah jenis *neural network* yang digunakan untuk memproses data sekuensial semacam itu [48]. Pada RNN, selain rangkaian data sesuai urutan yang menjadi input, output RNN pada suatu tahap juga akan dimasukkan kembali menjadi input tambahan

untuk tahap selanjutnya. Mekanisme ini yang membuat RNN bisa melakukan deduksi berdasar urutan data yang telah diproses pada langkah-langkah sebelumnya. Panjang sekuen pada RNN disebut juga dengan *sequence length* atau *timesteps*.

Secara teori, RNN bisa memproses data sekuensial sepanjang apapun. Jumlah data sebelumnya yang menjadi *feedback* input untuk tahap selanjutnya adalah sepanjang pola input yang hendak dikenali. Namun pada praktik-nya, topologi *recurrent* dinyatakan dengan melakukan *cloning* terhadap unit RNN sebanyak jumlah langkah ke belakang yang diperlukan. Deretan *clone* ini akan membentuk untaian *neural network* yang panjang. Karena pelatihan yang digunakan adalah *backpropagation* seperti *neural network* pada umumnya, untaian yang terlalu panjang akan cenderung mengalami masalah *vanishing gradient* [49]. Masalah ini terjadi bila propagasi *error* ke layer awal pada saat pelatihan mengalami penurunan nilai secara eksponensial, dan membuat bobot *layer-layer* awal sulit tar-update.

Long Short Term Memory (LSTM) adalah varian dari RNN yang bisa mengatasi masalah *vanishing gradient*. LSTM diciptakan oleh Sepp Hochreiter dan Jürgen Schmidhuber pada tahun 1997 [25]. Saat ini LSTM dan merupakan model yang paling banyak digunakan di dunia *Deep Learning* untuk kepentingan *Natural Language Processing*. LSTM juga disempurnakan oleh banyak orang seperti Felix Gers, Fred Cummins, Santiago Fernandez, Justin Bayer, Daan Wierstra, Julian Togelius, Faustian Gomez, Matteo Gagliolo, dan Alex Graves. LSTM merupakan suatu unit yang didesain khusus untuk menangani masalah *long term dependencies*. Desain arsitektur sebuah RNN yang tersusun dari LSTM unit akan disebut sebagai LSTM *network* [48].

LSTM Network memiliki arsitektur yang sama dengan RNN biasa, namun komponen dari unit yang digunakan cukup berbeda. Pada Gambar 5.3, dapat dilihat bahwa LSTM unit memiliki komponen yang lebih kompleks daripada RNN. Sebuah unit RNN hanya berupa gerbang aktivasi sederhana, sedangkan pada unit LSTM, terdapat beberapa *gate*. Yang termasuk dalam *gate-gate* tersebut adalah *cell state gate*, *forget gate*, *input gate*, *candidate gate*, dan *output gate* [25]. *Cell state gate* berfungsi untuk meneruskan *hidden state* dari *layer* sebelumnya. *Gate* inilah

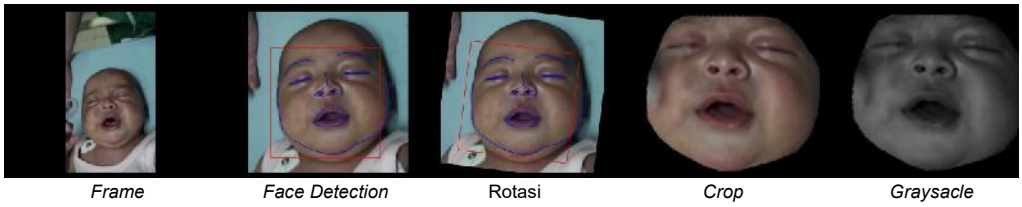


Gambar 5.3 Ilustrasi LSTM *Unit*

kunci utama dari LSTM yang dapat menghindari masalah *vanishing gradient*. Dalam meneruskan informasi, ada beberapa informasi yang perlu dihilangkan atau dipertahankan. Di sini *forget gate* berperan untuk menentukan informasi pada *cell state* yang perlu dihilangkan atau dipertahankan. Fungsi dari *input gate* dan *candidate gate* adalah untuk menentukan nilai dari *cell state* yang perlu di-*update*. Sedangkan output *gate* berfungsi untuk menentukan nilai yang dari *hidden state* yang akan diteruskan sebagai input pada hidden layer selanjutnya.

5.5 Klasifikasi Nyeri dan Tangis Pada Wajah Bayi

Dengan memanfaatkan *deep learning*, maka ekstraksi fitur secara *handcrafted* tidak perlu dilakukan seperti yang dilakukan pada penelitian-penelitian sebelumnya [2], [20], [50]. Bagian yang melakukan *feature learning* dalam penelitian ini adalah *convolutional autoencoder*. Bagian ini dengan sendirinya dapat melakukan pendeteksian tepi, pendeteksian warna, kontras, pola, dan fitur-fitur lain yang lebih kompleks yang dibutuhkan dalam proses klasifikasi. Namun untuk memaksimalkan hasil dan untuk mendapatkan input yang konsisten dalam segi ukuran, pewarnaan, dan area yang dianggap penting, maka dalam penelitian ini tetap dilakukanlah tahap praproses yang dijelaskan dalam sub bab berikutnya.



Gambar 5.4 Tahap *Preprocessing*

5.5.1 Tahap Praproses

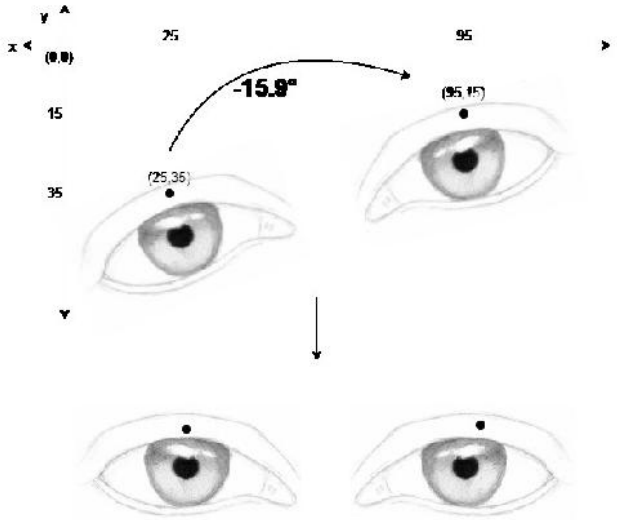
Tahap-tahap praproses yang dilakukan antara lain adalah mendeteksi wajah, mendeteksi titik-titik (*Landmark*) penting dari wajah, merotasi wajah, mengambil *Region of Interest* dan melakukan *grayscaling.* Ilustrasi tahap-tahap praproses yang dilakukan dalam penelitian ini dapat dilihat pada Gambar 5.4.

Tahap deteksi wajah berguna untuk mendapatkan area wajah secara persegi sehingga proses selanjutnya tidak dilakukan pada seluruh area gambar, namun hanya dilakukan di area wajah saja. Untuk tahap deteksi wajah digunakan implementasi algoritme Viola Jones [8] pada DLIB *library* yang memanfaatkan adaptive boosting seperti pada penelitian [51].

Setelah mendapatkan persegi area wajah, gambar akan di-crop sesuai dengan area wajah yang diperluas 150%. Setelah itu, akan dilanjutkan dengan proses deteksi poin-poin *landmark* wajah. Tujuan dari tahapan ini adalah untuk mendapatkan titik-titik pada bagian mata, bagian alis, dan bagian dagu. Bagian mata akan berguna sebagai pedoman pada tahap selanjutnya yakni rotasi wajah. Sedangkan titik dagu dan alis sebagai pedoman pengambilan ROI. Untuk melakukan deteksi titik-titik penting wajah dalam penelitian ini digunakan implementasi DLIB dari algoritme Kazemi dkk [52].

Setelah melalui tahap deteksi titik-titik penting wajah, dilanjutkan dengan proses rotasi Citra, dimana citra akan dirotasi menurut sudut dari posisi kedua titik *landmark* pada mata dengan garis horisontal. Gambar 5.5 merupakan ilustrasi dari rotasi wajah berdasarkan titik *landmark* pada mata ini.

Tujuan dari tahap rotasi gambar berdasarkan posisi mata adalah agar gambar yang diumpan ke dalam *network* merupakan gambar yang memiliki standarisasi yaitu wajah yang tegak lurus. Setelah melalui tahap tersebut, dilanjutkan dengan pengambilan ROI berdasarkan alis dan dagu. Setelah melalui



Gambar 5.5 Ilustrasi Rotasi Berdasarkan Posisi Mata

tahap tersebut, akan dilanjutkan dengan mengubah gambar diatas menjadi *grayscale.* Setelah itu gambar siap diumpankan ke proses *autoencoder*.

5.5.2 Tahap Convolution Autoencoder

Untuk melatih sebuah *convolutional autoencoder* yang baik diperlukan beberapa parameter yang harus diatur. Hal-hal yang harus dipertimbangkan adalah: Seberapa kecil reduksi yang kita lakukan, berapa layer konvolusi yang akan diterapkan, jumlah dan dimensi spatial filter yang akan dipakai. Filter yang digunakan pada *network* ini berukuran 3x3 di mana untuk setiap *convolutional layer* akan memiliki *depth* atau jumlah filter sebanyak 10. Stride yang digunakan adalah sebesar 2 dan menggunakan *same padding*. *Stride* adalah jarak lompatan filter dalam proses konvolusi. Dengan digunakan *stride* sebesar 2 akan mereduksi dimensi menjadi setengah dimensi input. Sedangkan *same padding* akan menambahkan nol pada pinggir input secara seimbang kanan-kiri, dan atas-bawah agar filter konvolusi dapat bekerja dengan pas.

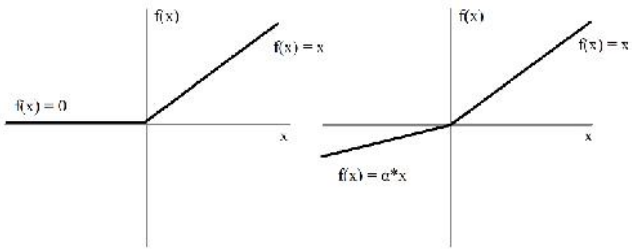
Sedangkan untuk *layer* aktifasi yang akan dicoba adalah *rectifier linear unit* (ReLU) dan *Leaky* ReLU [53]. Persamaan ReLU dapat dilihat pada Persamaan (5.1) dan *Leaky* ReLU pada Persamaan (5.2). Pada penelitian ini digunakan parameter a sebesar 0.2. Perbandingan grafik ReLU dan *Leaky* ReLU dapat dilihat pada Gambar. 5.6. Leaky ReLU dipakai dalam penelitian ini karena penggunaan

ReLU dapat menyebabkan *Dying ReLU Unit* (ReLU *unit* yang selalu mengeluarkan output 0).

$$f(x) = \begin{cases} x, & if\ x > 0 \\ 0, & lainya \end{cases} \tag{5.1}$$

$$f(x) = \begin{cases} x, & if\ x > 0 \\ ax, & lainya \end{cases} \tag{5.2}$$

Pada tahap *autoencoder*, sebuah model dapat dikatakan baik apabila input dan output yang dihasilkan memiliki kemiripan yang tinggi. Penggunaan *mean square error* (MSE) atau *peak signal to noise ratio* (PSNR) sebagai alat ukur kemiripan dua buah citra ternyata kurang baik karena lebih fokus pada selisih citra per *pixel* sehingga mengakibatkan hasil *autoencoder* lebih *noisy*.



Gambar 5.6 Grafik ReLU dibandingkan dengan Leaky ReLU

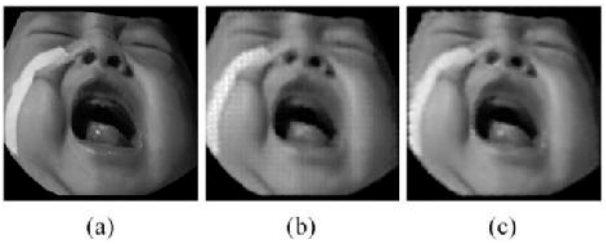
Dalam penelitian ini untuk mengukur tingkat kemiripan dua buah citra digunakan *Structural Similarity Index* (SSIM) [54] yang melihat perbedaan dua citra lebih global, Persamaan SSIM dapat dilihat pada Persamaan 5.3.

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{5.3}$$

$$c_1 = (0{,}01 * L)^2 \tag{5.4}$$

$$c_2 = (0{,}03 * L)^2 \tag{5.5}$$

Dalam Persamaan 5.3 x dan y adalah dua image yang berbeda. Dimana μ_x dan μ_y merupakan rata-rata *graylevel* untuk citra x dan y. Simbol σ_x^2 dan σ_y^2 adalah *variance* antar dua buah citra, sedangkan σ_{xy} adalah *covariance* dari dua buah citra. Untuk c_1 Persamaan 5.4 dan c_2 Persamaan 5.5 adalah konstanta yang digunakan untuk menstabilkan pembagian dengan nilai kecil. L adalah konstanta yang didapat



Gambar 5.7 Performa Autoencoder. (a) Citra Input, (b) Hasil Autoencoder dengan Loss MSE, dan (c) Hasil Autoencoder dengan Loss SSIM.

dari $2^{\text{jumlah bit}}$-1, dalam penelitian ini *L* bernilai 255. Perbandingan hasil *autoencoder* antara memanfaatkan loss MSE dan SSIM dapat dilihat pada Gambar 5.7.

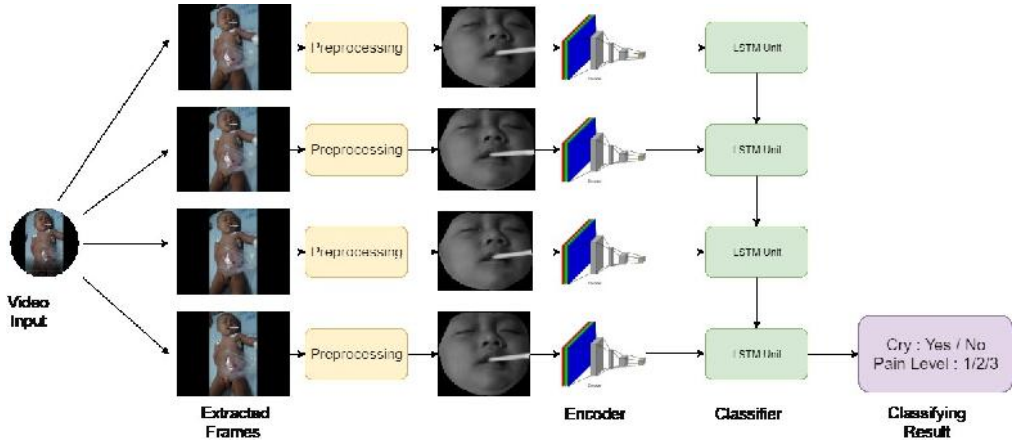
Pooling layer tidak digunakan dalam pembentukan CAE, mengingat pada *decoder* nantinya model harus mampu untuk mengembalikan *latent feature* menjadi input kembali. Apabila menggunakan *pooling layer*, maka akan banyak data yang hilang dan sulit untuk dapat dikembalikan.

Setelah *autoencoder* terlatih, model yang akan digunakan adalah *convolutional encoder* yang merupakan potongan setengah bagian depan dari *convolutional autoencoder*. Pada penelitian ini bagian akhir dari *encoder* memiliki dimensi 8x8x10, inilah yang merupakan *latent feature*. Setelah kita memiliki *network* yang dapat mengubah gambar menjadi data dengan dimensi yang lebih kecil, kita akan lanjutkan dengan memasukkan data tersebut ke dalam *classifier*.

5.5.3 Tahap Klasifikasi Menggunakan LSTM

Alasan digunakan RNN adalah karena input merupakan deretan gambar yang sekuensial sehingga urutan waktu menjadi informasi yang substansial. Dengan memanfaatkan RNN maka informasi sekuen dari gambar-gambar yang dipelajari juga tidak akan hilang. Sedangkan LSTM dipilih karena terbukti pada beberapa penelitian sebelumnya mampu menghasilkan akurasi yang cukup tinggi untuk data-data sekuensial [55] [48].

LSTM akan menerima hasil *encoding* dari *autoencoder* atau network sebelumnya. Sebelum data diumpankan ke dalam LSTM, data dari *encoder* harus



Gambar 5.8 Ilustrasi Sistem Klasifikasi Tangis dan Nyeri dari Video Wajah Bayi

Gambar 5.9 Contoh Potongan Video Output dari Sistem

diubah ke flat *layer* terlebih dahulu dikarenakan input dari LSTM merupakan *tensor* satu dimensi. LSTM yang digunakan akan memiliki panjang sekuen dan jumlah *hidden unit* yang bervariasi. Serangkaian percobaan akan dilakukan untuk membuktikan jumlah sekuen dan jumlah *hidden unit* terbaik.

Dalam implementasinya, apabila LSTM memiliki jumlah sekuen empat, maka program akan mengambil *frame* video yang berurutan sebanyak empat dimana masing-masing *frame* akan diumpankan kedalam *convolutional encoder*. Hasilnya adalah empat set *latent feature* yang diumpankan ke dalam LSTM berdasarkan waktu. Ilustrasi sistem dari input hingga klasifikasi LSTM dapat dilihat pada Gambar. 5.8.

Setelah data diumpankan ke dalam LSTM, LSTM akan mengeluarkan label untuk data sekuen yang diumpankan. Hal itu akan diulang terhadap semua

frame sekuen yang ada pada video input. Setelah itu, semua *frame* yang telah diklasifikasikan tersebut akan dikonstruksi kembali menjadi sebuah video yang memiliki informasi kelas nyeri dan tangis didalamnya. Video inilah yang menjadi output dari sistem pada penelitian ini, contoh potongan dari video tersebut ditunjukan pada Gambar 5.9.

5.6 Ujicoba Sistem

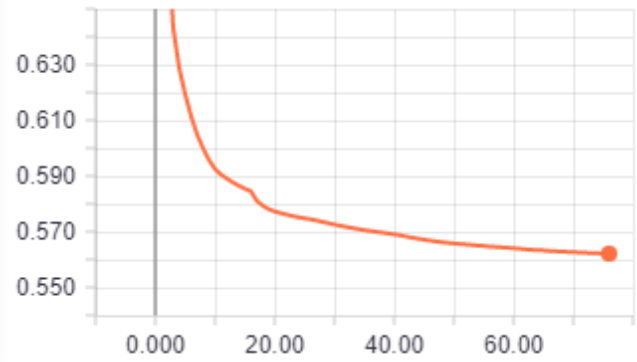
Dalam bab ini dijelaskan tentang uji coba yang dilakukan demi mendapatkan model terbaik dengan cara mengganti parameter-parameter seperti jumlah *layer*, *learning rate*, dan jumlah frame. Tahap pertama dari rangkaian uji coba yang dilakukan adalah uji coba rekonstruksi citra menggunakan DCNN *autoencoder*. Parameter yang dicoba adalah variasi *learning rate,* fungsi aktivasi, dan jumlah hidden layer. Sedangkan fungsi loss pada semua percobaan menggunakan SSIM

Pada proses training DCNN *autoencoder*, semua filter memiliki ukuran 3x3, *stride* 2, dan *depth* 10. Uji Coba DCNN *autoencoder* dilakukan dengan menggunakan 879 citra sebagai *training* set dan 121 citra sebagai *testing set*. Semua uji coba yang akan dilakukan untuk *autoencoder* pada penelitian ini ditampilkan pada Tabel 5.1. Didapatkan bahwa *autoencoder* terbaik menggunakan *leaky* ReLU serta memakai 3 *layer Conv* dan 3 *layer Deconv*.

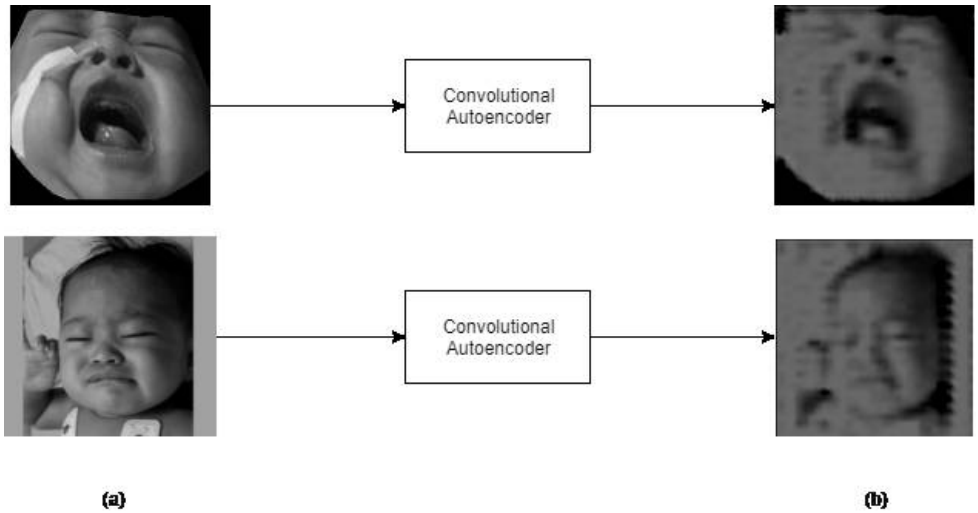
Tabel 5.1 Hasil Uji Coba Autoencoder

Learning Rate	Fungsi Aktivasi	Jumlah Percobaan	Jumlah *Layer Convolutional dan Deconvolution*	Waktu Rata-rata (detik)	*Minimum Loss*	*Maximum Loss*
0.001	ReLU	4	4	2781.4	0.581	0.609
0.01	LReLU	2	4	6567.4	0.145	0.155
0.01	**LReLU**	**8**	**3**	**1637.9**	**0.037**	**0.2**
0.01	LReLU	3	2	1167.1	0.038	0.47

Pada percobaan pertama dengan jumlah convolutional layer adalah 4 dan jumlah deconvolutional layer adalah 4. Pada percobaan yang pertama ini, digunakan fungsi aktivasi ReLU. Model sekanrio pertama dilatih dengan metode *early stopping* dimana apabila model sudah tidak mampu lagi untuk menurunkan



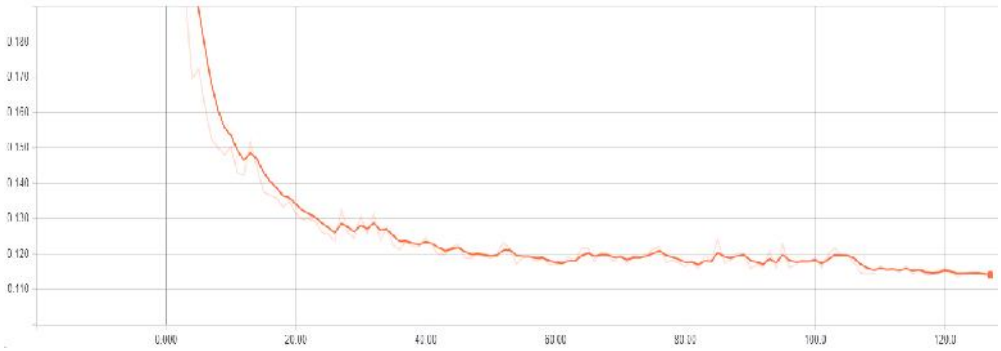
Gambar 5.10 Grafik Loss pada ujicoba CAE skenario pertama



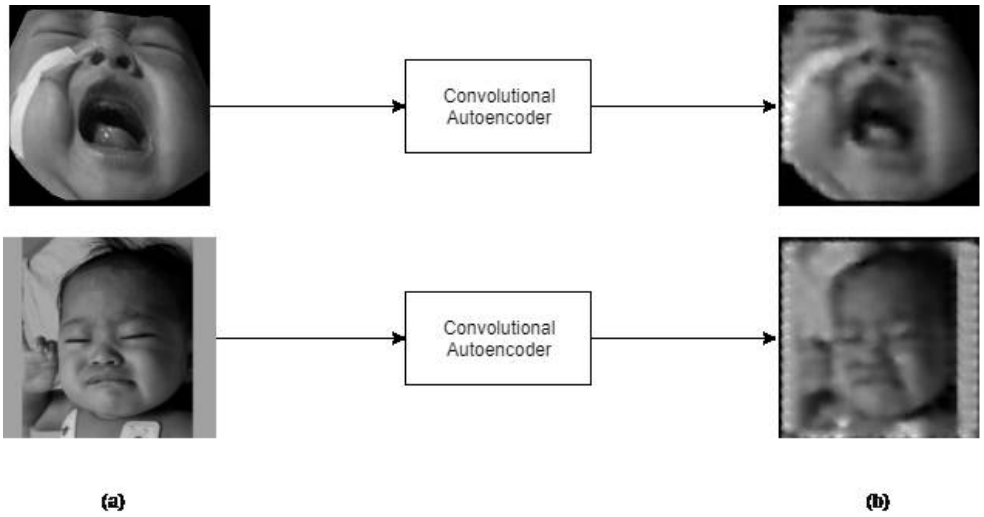
Gambar 5.11 Citra Input (a) Beserta Citra Hasil Rekonstruksi Dari Model Skenario Pertama (b)

loss nya, maka iterasi akan dihentikan. Gambar 5.10 menunjukkan grafik loss dan Gambar 5.11 merupakan gambar input (a) beserta hasil rekonstruksi dari model pada skenario pertama (b). Hasil rekonstruksi tampak buruk dan detail wajah banyak yang hilang.

Percobaan dilanjutkan dengan dengan mengubah learning rate dari 0.001 menjadi 0.01 serta mengubah fungsi aktivasi dari ReLU menjadi Leaky ReLU pada seknario kedua. Hal ini bertujuan untuk mengetahui pengaruh learning rate terhadap hasil yang didapat dari model serta mengetahui perbedaan antara performa dari Leaky ReLU dan ReLU.

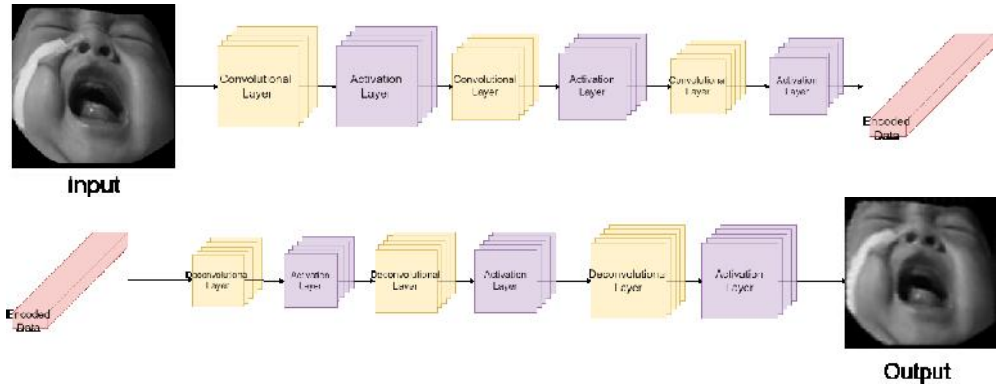


Gambar 5.12 Grafik Hasil Loss Pada CAE Skenario Kedua

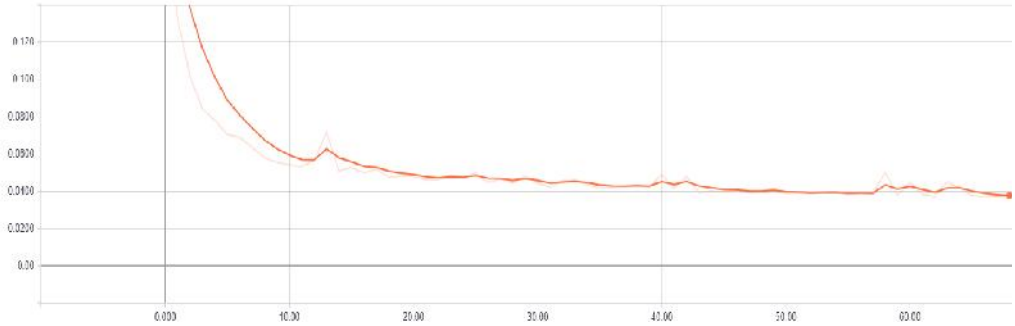


Gambar 5.13 Citra Input (a) Beserta Citra Hasil Rekonstruksi Dari Model Skenario Kedua (b)

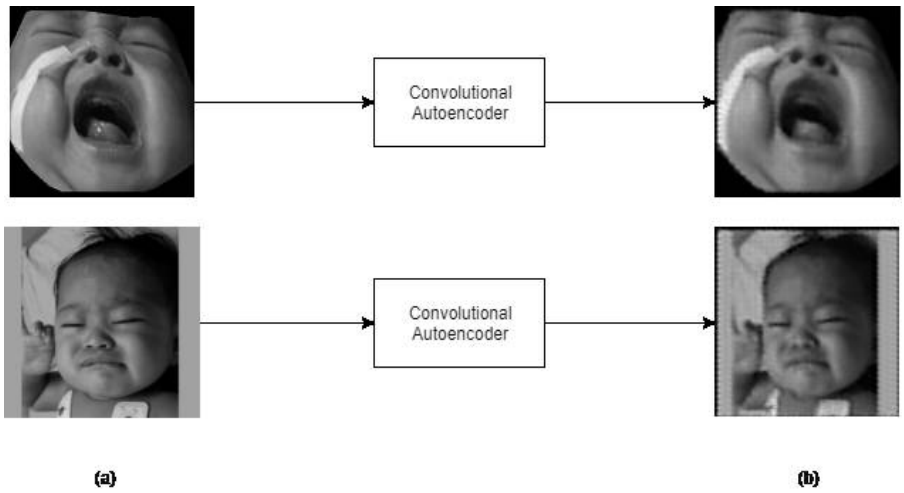
Pada percobaan kedua (baris kedua tabel 5,1) dapat dilihat bahwa performa model meningkat cukup drastis. Hal ini berarti pengubahan fungsi aktivasi dari ReLU menjadi Leaky ReLU membawa dampak yang cukup positif. Pada uji coba ini juga dilakukan dengan teknik early stopping. Pada gambar 5.12 tampak bahwa nilai loss turun cukup jauh dibandingkan dengan skenario pertama. Gambar 5.13 menunjukkan peningkatan kualitas hasil rekonstruksi yang cukup membaik meskipun beberapa detail wajah masih banyak yang hilang.



Gambar 5.14 Arsitektur Convolutional Autoencoder Pada Skenario Ketiga

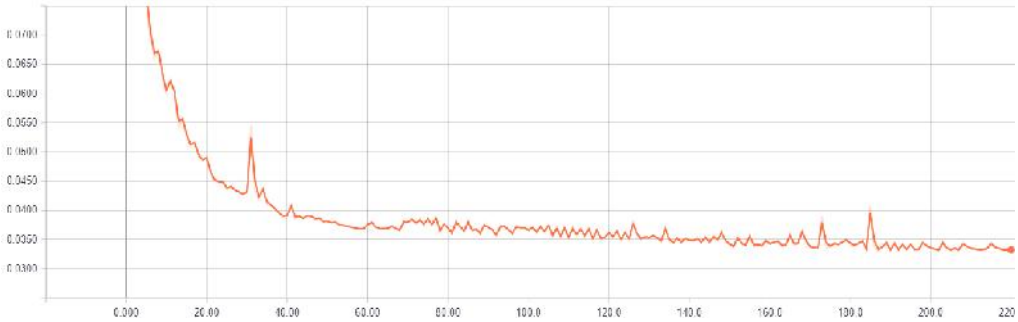


Gambar 5.15 Grafik Hasil Loss Skenario Ketiga

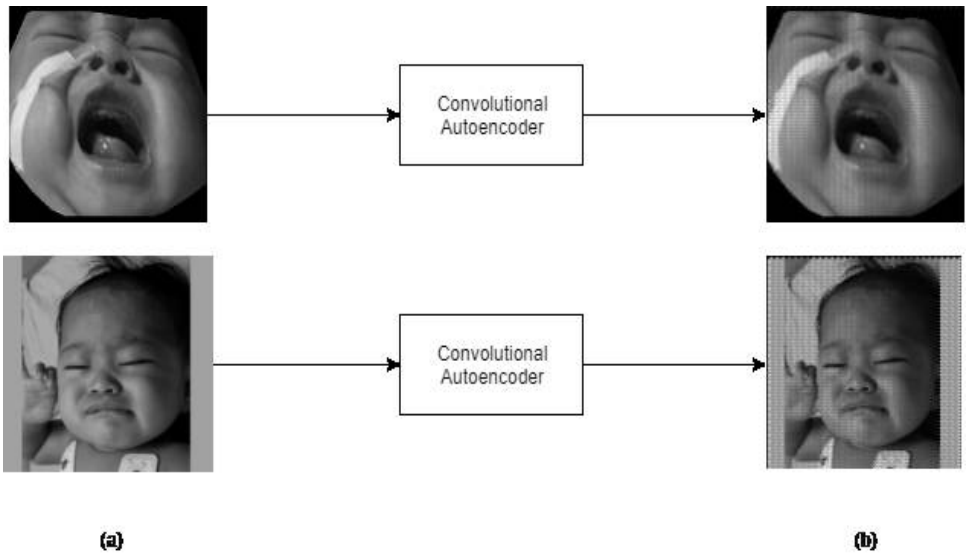


Gambar 5.16 Citra Input (a) Beserta Citra Hasil Rekonstruksi Dari Model Skenario Ketiga (b)

Percobaan berikutnya dilakukan dengan mengurangi jumlah convolutional layer dari 4 menjadi 3 dengan asumsi CAE yang dibentuk mengalami *overfitting*. Dengan mengurangi jumlah layer, diharapkan dapat mengurang *overfitting* dan



Gambar 5.17 Grafik Hasil Loss Skenario Keempat



Gambar 5.18 Citra Input (a) Beserta Citra Hasil Rekonstruksi Dari Model Skenario Keempat (b)

waktu komputasi. Pada Gambar 5.14 tampak ilustrasi arsitektur CAE untuk uji coba skenario ketiga.

Pada Gambar 5.15 tampak hasil percobaan skenario ketiga memiliki loss yang lebih rendah dari dua skenario sebelumnya, dan pada Gambar 5.16 tampak hasil rekonstrukci citra dengan autoencoder juga mengalami perbaikan. Dari hasil uji coba tersebut, disimpulkan bahwa 3 convolutional layer sebanyak 3 memiliki performa yang lebih baik daripada 4 convolutional layer.

Pada skenario keempat, peneliti mecoba mengurangi lagi jumlah convolutional layer dan deconvolutional layer dari 3 menjadi 2. Hal ini bertujuan untuk mengetahui apakah dengan mengurangi jumlah layer convolution dan deconvolution lebih sedikit lagi, masih akan mampu mengurangi loss yang

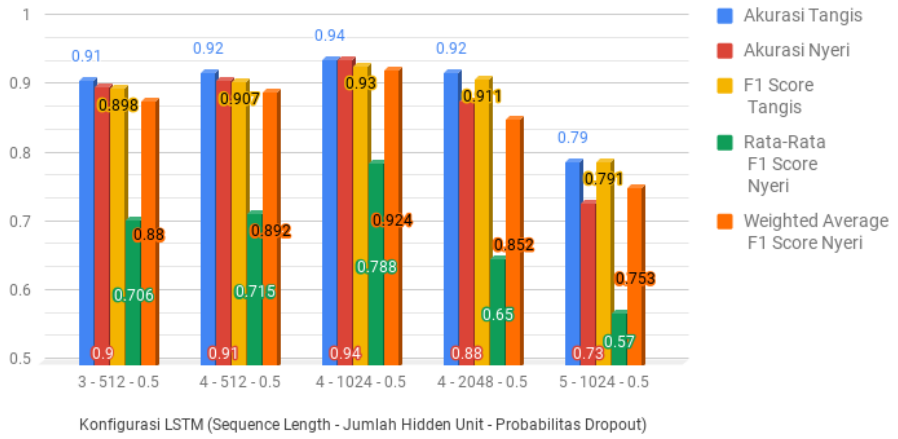
dihasilkan dan meningkatkan kualitas hasil rekonstruksi. Gambar 5.17 menunjukkan grafik loss pada uji coba skenario keempat dapat sedikit membaik dari skaneriuo ketiga, namun hasil rekonstruksi citra pada Gambar 5.18 tampak lebih noisy dan grainny. Karena itu untuk CAE digunakan arsitektur seperti gambar 5.14.

Setelah melakukan beberapa percobaan pada DCNN *autoencoder*, dilanjutkan dengan percobaan klasifikasi menggunakan LSTM. Percobaan ini dilakukan dengan memanfaatkan model *autoencoder* yang mempunyai nilai *loss* terendah yaitu 0.037.

Uji Coba LSTM dilakukan dengan memanfaatkan *dataset* IFPaLD yang telah dijelaskan sebelumnya. Uji coba pada LSTM akan dilakukan dengan mengganti beberapa hyperparameter LSTM, menambah dan mengurangi jumlah *hidden* unit yang ada, serta mengubah panjang sekuen atau *time steps*. Sistem pada penelitian ini dirancang untuk tidak memiliki panjang sekuen yang terlalu panjang mengingat jumlah data yang terbatas akibat dari performa *face detection* yang belum mencapai 100%.

Pada semua uji coba LSTM yang dilakukan pada penelitian ini, tekniknya adalah dengan membagi *dataset training* menjadi 20% data validasi dan 80% data *training* secara random. Untuk menghindari *overfit*, akan digunakan regularisasi L2 dengan parameter lambda sebesar 0.001. Tabel 5.2 merupakan konfigurasi dan hasil dari semua uji coba LSTM yang dilakukan.

Setelah melakukan percobaan sebelumnya, didapatkan model terbaik dengan panjang sekuen 4 dan jumlah *hidden* unit 1024. Percobaan berikutnya dilanjutkan dengan mencoba mengubah probabilitas *dropout* untuk mengurangi *overfitting* mengacu pada penelitian Srivastava dkk. pada tahun 2014 [56]. Nilai probabilitas *dropout* menunjukkan jumlah unit yang di-*drop* secara *random* untuk tiap layernya. Nilai probabilitas *dropout*=0.0 artinya tidak ada unit yang di-*drop*. Hasil uji coba pengaruh *dropout layer* terhadap LSTM dapat dilihat pada Tabel 5.3, hasil terbaik untuk deteksi tangis didapatkan dengan memakai probabilitas *dropout* sebesar 0.5, dan hasil terbaik untuk klasifikasi nyeri didapatkan dengan *dropout* sebesar 0.25. Untuk memperjelas perbandingan hasil percobaan pada Tabel 5.3 digambarkan menjadi grafik pada Gambar 5.19.



Gambar 5.19 Visualisasi Hasil Uji Coba LSTM

Tabel 5.2 Uji Coba Long Short Term Memory

Sequence Length	Jumlah *Hidden Unit*	Waktu *Training* (detik)	Akurasi Tangis	Akurasi Nyeri	F1 *Score* Tangis	Rata-Rata F1 *Score* Nyeri	*Weighted Average* F1 *Score* Nyeri
3	512	944.3	0.91	0.90	0.898	0.706	0.880
4	512	3347.5	0.92	0.91	0.907	0.715	0.892
4	**1024**	**5229.7**	**0.94**	**0.94**	**0.930**	**0.788**	**0.924**
4	2048	6967.3	0.92	0.88	0.911	0.65	0.852
5	1024	26459.8	0.79	0.73	0.791	0.57	0.753

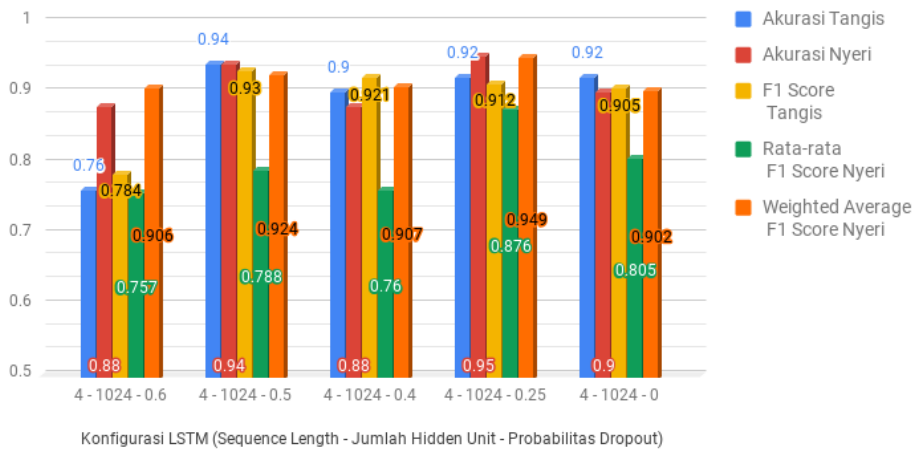
Tabel 5.3 Uji Coba Dropout Pada Long Short Term Memory Network

Probabilitas Dropout	Akurasi Tangis	Akurasi Nyeri	F1 *Score* Tangis	Waktu *Training* (detik)	Rata-rata F1 *Score* Nyeri	*Weighted Average* F1 *Score* Nyeri
0.6	0.76	0.88	0.784	5098.2	0.757	0.906
0.5	0.94	0.94	0.930	5229.7	0.788	0.924
0.4	0.90	0.88	0.921	5611.9	0.760	0.907
0.25	**0.92**	**0.95**	**0.912**	**5844.6**	**0.876**	**0.949**
0.0	0.92	0.90	0.905	6102.2	0.805	0.902

Tabel 5.4 Confusion Matrix dari Deteksi Tangis

		Prediksi		
		Menangis	Tdk Menangis	**Total**
Data Riil	Menangis	**1137**	35	1172
	Tdk Menangis	137	**1674**	1811
	Total	1274	1709	**2983**

Untuk lebih memperjelas hasil yang diperoleh, ditampilkan *confusion matriks* dari model terbaik untuk deteksi tangis pada Tabel 5.4 dan *confusion matrix* untuk klasifikasi nyeri pada Tabel 5.5. Dari Tabel 5.4 tampak bahwa sistem dapat



Gambar 5.20 Visualisasi Hasil Uji Coba Dropout Pada LSTM

mengklasifikasikan tangis dengan baik. Sedangkan dari Tabel 5.5 tampak bahwa sistem dapat mengklasifikasikan tingkat nyeri tinggi dan tidak nyeri dengan baik, namun mengalami kesulitan dalam mengklasifikasikan tingkat nyeri sedang.

Tabel 5.5 Confusion Matrix dari Klasifikasi Nyeri

		Prediksi			
		Tidak Nyeri	Sedang	Tinggi	**Total**
Data Riil	Tidak Nyeri	**943**	2	0	945
	Sedang	85	**139**	22	246
	Tinggi	21	11	**1760**	1792
	Total	1049	152	1782	2983

Tabel 5.6 Perbandingan Dengan Penelitian Sebelumnya

Metric	Penelitian Ini	Penelitian Sebelumnya [50]
F1 Score Kelas Tangis	0.942	**0.961**
Weighted Avg F1 Score Nyeri	**0.952**	0.949
F1 Score Kelas Nyeri Tinggi	**0.985**	0.952

Perbandingan dengan penelitian sebelumnya [50] dapat dilihat pada Tabel 5.6. Dari tabel ini tampak bahwa performa penelitian ini dalam klasifikasi tangis masih sedikit dibawah penelitian sebelumnya, hal ini kemungkinan disebabkan karena dataset yang digunakan pada penelitian ini juga lebih banyak dari penelitian sebelumnya. Pada kesimpulan penelitian sebelumnya, mereka fokus dalam mengklasifikasikan tingkat nyeri tinggi. Hasil terbaik pada penelitian ini masih

sedikit lebih baik dari penelitian sebelumnya khusus untuk klasifikasi derajat nyeri tinggi pada bayi.

5.7 *Concluding Remarks*

Dari penelitian yang dilakukan, dapat diambil beberapa kesimpulan bahwa hasil dari klasifikasi LSTM ataupun *feature extraction* dari *autoencoder* sangatlah dipengaruhi oleh seberapa baik fase *face detection* dan *landmark detection*. Dengan semakin baiknya fase ini, maka semakin banyak dan baik pula wajah yang terdeteksi dimana akan menghasilkan sebuah pengetahuan yang bagus untuk arsitektur jaringan.

Dari hasil uji coba *autoencoder*, dapat dilihat bahwa hasil terbaik didapatkan dengan *convolutional layer* sebanyak 3 *layer* dan *deconvolutional layer* sebanyak 3 *layer*. Dari hasil penelitian juga dapat disimpulkan bahwa fungsi aktivasi *leaky* relu ternyata lebih baik daripada fungsi aktivasi relu pada umumnya. Dengan arsitektur tersebut, dihasilkan sebuah model dengan *loss* sebesar 0.037.

Sedangkan dari hasil uji coba LSTM, dapat dilihat bahwa hasil terbaik untuk klasifikasi tingkat nyeri didapatkan dengan panjang sekuen 4, *hidden* unit 1024 dan probabilitas *dropout layer* sebesar 0.25. Dengan menggunakan arsitektur tersebut, dihasilkan sebuah model dengan *weighted average F1 Score* sebesar 0.948 dan F1 Score untuk kelas nyeri tinggi mencapai 0.985.

Halaman ini sengaja dikosongkan

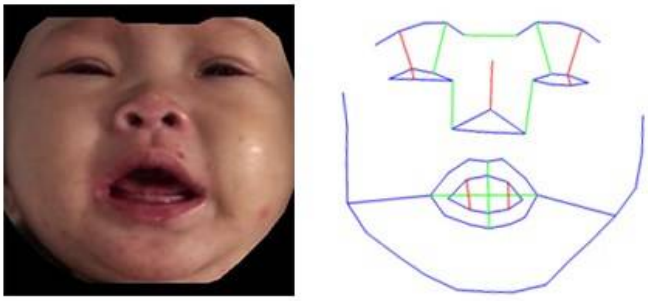
BAB 6
PENUTUP

Pada bab penutup ini dibahas kesimpulan-kesimpulan yang telah diperoleh dari hasil penelitian serta rencana usulan penelitian selanjutnya.

6.1 Kesimpulan Hasil Penelitian

Berdasarkan hasil penelitian, dapat disimpulkan beberapa hal berikut:

1. Chaotic NN yang diteliti pada paper seminar sbenarnya memiliki potensi untuk dikembangkan lebih lanjut, namun arah perkembangan dunia nural network adalah kearah feature learning menggunakan deep learning.
2. Fitur Geometri yang dikembangkan mampu menjadi fitur yang baik untuk klasifikasi tangis pada wajah bayi. Fitur ENPPD (Extended Normalized Point Pair Distance) yang dikembangkan pada paper Jurnal Internasional performanya mengalahkan PPD standar yang dikembangkan pada penelitian sebelumnya.
3. Fitur Tekstur yang dikembangkan pada paper jurnal dengan skenario multiple overlapping patches ditambah LBP ternyata mampu menjadi fitur yang baik dalam mengklasifikasikan wajah bayi yang mengalami nyeri dengan akurasi yang cukup memuaskan.
4. Pemanfaatan autoencoder berbasis convolutional neural network sebagai feature learner ternyata mampu menggantikan tahap ekstraksi fitur manual (handcrafted feature) dan mampu membantu classifier dalam mengklasifikasikan derajad nyeri pada wajah bayi dengan akurasi yang cukup baik.
5. Pemanfaatan LSTM sebagai classifier untuk sekuen citra memiliki performa yang cukup baik dalam mengkalsifikasikan nyeri pada deretan frame wajah bayi.



Gambar 6.1 Dua representasi citra wajah, (kiri) Matrix Pixel RGB, (kanan) titik-titik landmark wajah.

Tabel 6.1 Dua macam arsitektur DCNN yang akan dicoba.

Layer No.	**Layer Kind**	**MR - RGB**		**MR-*Graylevel***	
Layer 0	Input	3x250x250	2x48	1x250x250	2x48
Layer 1	Conv	N=16 F=3x3, S=1, P=1, A=ReLu		N=32 F=3x3, S=1, P=1, A=ReLu	
Layer 2	Max Pool	F=2x2, S=2		F=2x2, S=2	
Layer 3	Conv	N=20 F=3x3, S=1, P=1, A=ReLu		N=20 F=3x3, S=1, P=1, A=ReLu	
Layer 4	Max Pool	F=2x2, S=2		F=2x2, S=2	
Layer 5	Conv	N=24 F=3x3, S=1, P=1, A=ReLu		N=24 F=3x3, S=1, P=1, A=ReLu	
Layer 6	Max Pool	F=2x2, S=2		F=2x2, S=2	
Layer 7	Conv	N=28 F=3x3, S=1, P=1, A=ReLu		N=28 F=3x3, S=1, P=1, A=ReLu	
Layer 8	Max Pool	F=2x2, S=2		F=2x2, S=2	
Layer 9	Conv	N=14 F=3x3, S=1, P=1, A=ReLu		N=14 F=3x3, S=1, P=1, A=ReLu	
Layer 10	Max Pool	F=2x2, S=2		F=2x2, S=2	
Layer 11	Flaten	686	96	686	96
Layer 12	Fully Connected	110, A= tanh	50, A=tanh	110, A= tanh	50, A=tanh
Layer 13	Fully Connected		15, A=tanh		15, A=tanh
Layer 14	Concatenate	125, A=tanh		125, A=tanh	
Layer 15	Fully Connected	25, A=tanh		25, A=tanh	
Layer 16	Softmax	3		3	

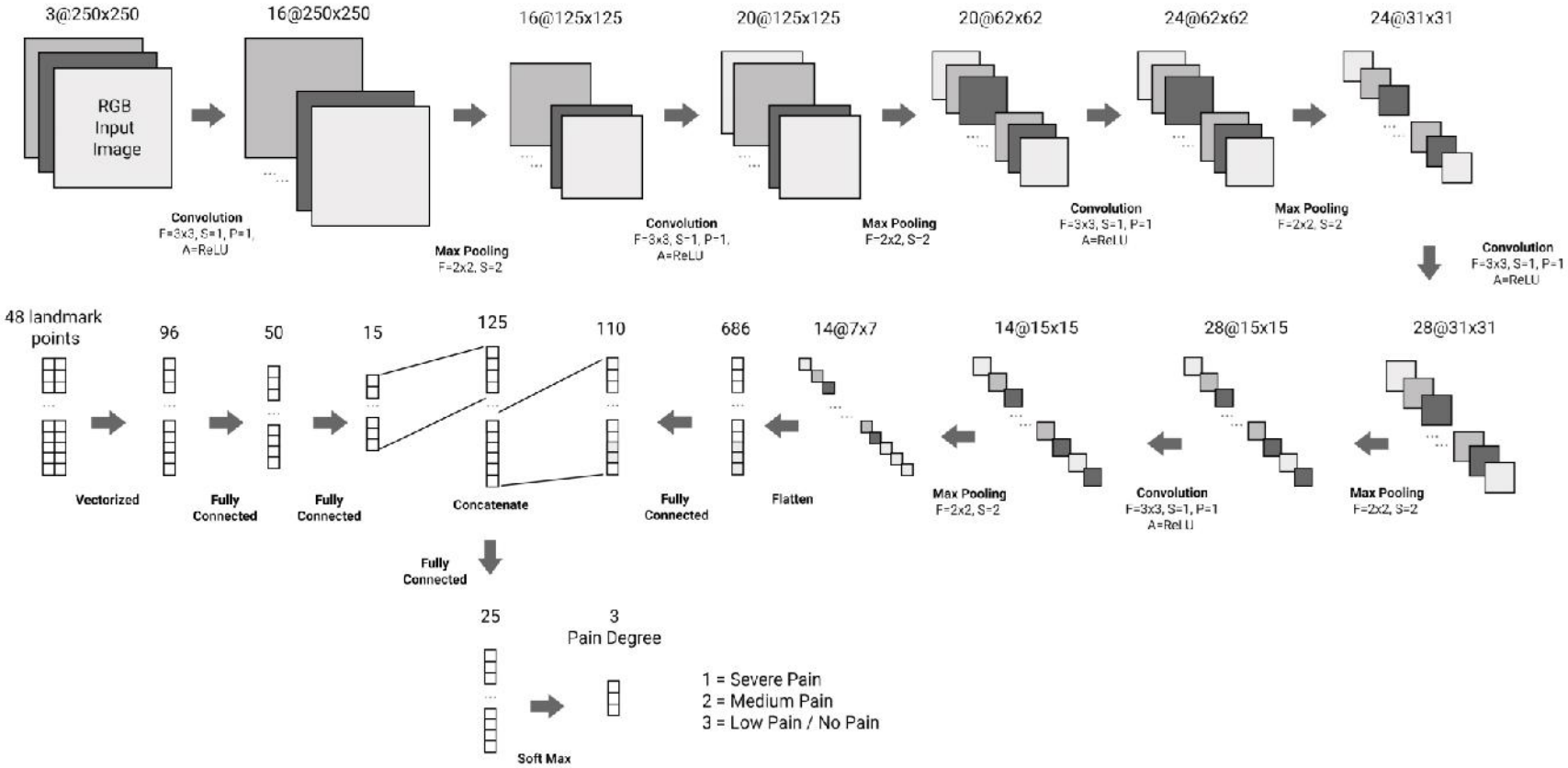
6.2 Rencana Penelitian Selanjutnya

Berdasarkan hasil-hasil yang telah diperoleh pada penelitian sebelumnya, maka rencana penelitian selanjutnya adalah tentang pemanfaatan representasi citra wajah dengan landmarknya untuk diklasifikasikan derajat nyerinya menggunakan multi representation (MR) DCNN.

Dengan menggunakan DCNN kita dapat melakukan penggabungan fitur melalui urutan layer yang berbeda. Pada penelitian selanjutnya akan dicoba mengumpankan dua buah representasi wajah seperti pada gambar 6.1 ke sebuah DCNN namun baru mengalami merging di tengah.

Pada penelitian selanjutnya juga akan dicoba apakah warna (RGB) memiliki pengaruh yang signifikan dalam upaya deteksi nyeri pada wajah bayi. Dua buah arsitektur DCNN yang akan dicoba dalam penelitian selanjutnya tampak pada tabel 6.1. Yang membedakan dua kolom ini adalah input pada MR-RGB menggunakan 3 *channel*, sedangkan MR-*Grayscale* hanya menggunakan 1 *channel*.

Pada tabel 6.1 tampak bahwa representasi wajah berupa matrix pixel dilewatkan *convolutional layer* dan *max pooling* sedangkan titik-titik *ladmark* wajah diproses menggunakan *fully connected layer.* Masih di tabel 6.1 tampak bahwa penggabungan layer baru terjadi pada layer ke-14 setelah matrix pixel melalui serangkaian *convolutional layer* sehingga didapatkan *latent feature*-nya. Ilustrasi layer-layer pada arsitektur DCNN yang direncanakan dapat dilihat pada gambar 6.2



Gambar 6.2 Dua representasi citra wajah bayi diumpankan ke dalam DCNN dengan target derajat nyeri yang dialami.

DAFTAR PUSTAKA

[1] K. M. Prkachin, N. A. Currie, dan K. D. Craig, "Judging nonverbal expressions of pain.," *Canadian Journal of Behavioural Science/Revue canadienne des sciences du comportement*, vol. 15, no. 4, hal. 409, 1983.

[2] P. Lucey, J. F. Cohn, I. Matthews, S. Lucey, S. Sridharan, J. Howlett, dan K. M. Prkachin, "Automatically detecting pain in video through facial action units," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 41, no. 3, hal. 664–674, 2011.

[3] E. Hanindito, "Dynamic Acoustic Pattern as Pain Indicator on Baby Cries Post Surgery Procedure," Universitas Airlangga, 2013.

[4] T. Voepel-Lewis, J. R. Shayevitz, dan S. Malviya, "The FLACC: a behavioral scale for scoring postoperative pain in young children," *Pediatr Nurs*, vol. 23, hal. 293–297, 1997.

[5] S. W. Krechel dan J. BILDNER, "CRIES: a new neonatal postoperative pain measurement score. Initial testing of validity and reliability," *Pediatric Anesthesia*, vol. 5, no. 1, hal. 53–61, 1995.

[6] M. Buchholz, H. W. Karl, M. Pomietto, dan A. Lynn, "Pain scores in infants: a modified infant pain scale versus visual analogue," *Journal of pain and symptom management*, vol. 15, no. 2, hal. 117–124, 1998.

[7] H. Elizeus, "Dynamic Acoustic Pattern as Pain Indicator on Baby Cries Post Surgery Procedure," Airlangga University, Surabaya, Jawa Timur, Indonesia, 2013.

[8] P. Viola dan M. J. Jones, "Robust real-time face detection," *International journal of computer vision*, vol. 57, no. 2, hal. 137–154, 2004.

[9] T. F. Cootes, C. J. Taylor, D. H. Cooper, dan J. Graham, "Active shape models-their training and application," *Computer vision and image understanding*, vol. 61, no. 1, hal. 38–59, 1995.

[10] B. Van Ginneken, A. F. Frangi, J. J. Staal, B. M. Ter Haar Romeny, dan M. A. Viergever, "Active shape model segmentation with optimal features," *IEEE Transactions on Medical Imaging*, vol. 21, no. 8, hal. 924–933, 2002.

[11] P. P. Smyth, C. J. Taylor, dan J. E. Adams, "Automatic measurement of

vertebral shape using active shape models," *Image and Vision Computing*, vol. 15, no. 8, hal. 575–581, 1997.

[12] A. Souza dan J. K. Udupa, "Automatic landmark selection for active shape models," no. 215, hal. 1377, 2005.

[13] C. Sagonas, E. Antonakos, G. Tzimiropoulos, S. Zafeiriou, dan M. Pantic, "300 faces in-the-wild challenge: Database and results," *Image and vision computing*, vol. 47, hal. 3–18, 2016.

[14] Z. Zhang, M. J. Lyons, M. Schuster, dan S. Akamatus, "Comparison between geometry-based and Gabor-wavelets-based facial expression recognition using multi-layer perceptron," *IEEE International Conference on Automatic Face & Gesture Recognition*, no. November 2012, 1998.

[15] T. Ahonen, A. Hadid, dan M. Pietikainen, "Face Description with Local Binary Patterns: Application to Face Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 12, hal. 2037–2041, 2006.

[16] M. Schiavenato, J. F. Byers, P. Scovanner, J. M. McMahon, Y. Xia, N. Lu, dan H. He, "Neonatal pain facial expression: Evaluating the primal face of pain," *Pain*, vol. 138, no. 2, hal. 460–471, 2008.

[17] T. Ojala, M. Pietikäinen, dan T. Mäenpää, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, no. 7, hal. 971–987, 2002.

[18] C. Shan, S. Gong, dan P. W. McOwan, "Facial expression recognition based on local binary patterns: A comprehensive study," *Image and Vision Computing*, vol. 27, no. 6, hal. 803–816, 2009.

[19] N. Algaraawi dan T. Morris, "Study on Aging Effect on facial Expression Recognition," *Proceedings of the World Congress on Engineering*, vol. I, 2016.

[20] L. Nanni, S. Brahnam, dan A. Lumini, "A local approach based on a Local Binary Patterns variant texture descriptor for classifying pain states," *Expert Systems with Applications*, vol. 37, no. 12, hal. 7888–7894, 2010.

[21] M. Adachi dan K. Aihara, "Associative dynamics in a chaotic neural

network," *Neural Networks*, vol. 10, no. 1, hal. 83–98, 1997.

[22] D. Calitoiu, B. J. Oommen, dan D. Nussbaum, "Periodicity and stability issues of a chaotic pattern recognition neural network," *Pattern analysis and applications*, vol. 10, no. 3, hal. 175–188, 2007.

[23] K. Qin dan B. Oommen, "Ideal chaotic pattern recognition using the modified adachi neural network," *Chaotic Modeling and Simulation (CMSIM) 2012*, hal. 701–710, 2012.

[24] Y. Lecun, Y. Bengio, dan G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, hal. 436–444, 2015.

[25] S. Hochreiter dan J. J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, hal. 1–32, 1997.

[26] F. A. Gers, J. Schmidhuber, dan F. Cummins, "Learning to forget: Continual prediction with LSTM," *Neural Computation*, vol. 12, no. 10, hal. 2451–2471, 2000.

[27] J. Bayer, D. Wierstra, dan J. Togelius, "Evolving memory cell structures for sequence learning," *Artificial Neural Networks–ICANN 2009*, hal. 755–764, 2009.

[28] B. E. Boser, I. M. Guyon, dan V. N. Vapnik, "A training algorithm for optimal margin classifiers," *Proceedings of the fifth annual workshop on Computational learning theory - COLT '92*, hal. 144–152, 1993.

[29] P. Ekman dan W. Friesen., "Facial Action Coding System: A Technique for the Measurement of Facial Movement.," *Consulting Psychologists Press*, 1978.

[30] K. M. Prkachin, "The consistency of facial expressions of pain: a comparison across modalities," *Pain*, vol. 51, no. 3, hal. 297–306, 1992.

[31] K. M. Prkachin dan P. E. Solomon, "The structure, reliability and validity of pain expression: Evidence from patients with shoulder pain," *Pain*, vol. 139, no. 2, hal. 267–274, 2008.

[32] C. Cortes dan V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, hal. 273–297, 1995.

[33] J. C. Platt, "Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines," *Advances in kernel methods*, hal. 185–208, 1998.

[34] M. W. Gardner dan S. R. Dorling, “Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences,” *Atmospheric environment*, vol. 32, no. 14–15, hal. 2627–2636, 1998.

[35] R. Hecht-Nielsen, “Theory of the Backpropagation Neural Network,” *Proceedings Of The International Joint Conference On Neural Networks*, vol. 1, hal. 593–605, 1989.

[36] A. Krizhevsky, “Learning Multiple Layers of Features from Tiny Images,” *Science Department, University of Toronto*, hal. 1–60, 2009.

[37] Y. LeCun, L. Bottou, Y. Bengio, dan P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, hal. 2278–2324, 1998.

[38] A. Krizhevsky, I. Sutskever, dan G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in neural information processing systems*, 2012, hal. 1097–1105.

[39] S. E. Limantoro, Y. Kristian, dan D. D. Purwanto, “Pemanfaatan Deep Learning pada Video Dash Cam untuk Deteksi Pengendara Sepeda Motor,” *JNTETI*, vol. 7, no. 2, 2018.

[40] W. Ouyang dan X. Wang, “Joint deep learning for pedestrian detection,” in *Computer Vision (ICCV), 2013 IEEE International Conference on*, 2013, hal. 2056–2063.

[41] G. Levi dan T. Hassner, “Age and gender classification using convolutional neural networks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2015, hal. 34–42.

[42] K. O’Shea dan R. Nash, “An Introduction to Convolutional Neural Networks,” *arXiv preprint*, vol. 1511, hal. 1–11, 2015.

[43] W. Wang, Y. Huang, Y. Wang, dan L. Wang, “Generalized autoencoder: A neural network framework for dimensionality reduction,” *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, hal. 496–503, 2014.

[44] G. E. Hinton dan R. R. Salakhutdinov, “Reducing the dimensionality of data with neural networks,” *science*, vol. 313, no. 5786, hal. 504–507, 2006.

[45] P. Vincent dan H. Larochelle, “Stacked Denoising Autoencoders: Learning

Useful Representations in a Deep Network with a Local Denoising Criterion Pierre-Antoine Manzagol," *Journal of Machine Learning Research*, vol. 11, hal. 3371–3408, 2010.

[46] S. Wold, K. Esbensen, dan P. Geladi, "Principal component analysis," *Chemometrics and intelligent laboratory systems*, vol. 2, no. 1–3, hal. 37–52, 1987.

[47] J. Masci, U. Meier, D. Cireşan, dan J. Schmidhuber, "Stacked Convolutional Auto-Encoders for Hierarchical Feature Extraction BT - Artificial Neural Networks and Machine Learning – ICANN 2011," hal. 52–59, 2011.

[48] H. Sak, A. Senior, dan F. Beaufays, "Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling," *Interspeech 2014*, no. September, hal. 338–342, 2014.

[49] S. Hochreiter, "The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 6, no. 2, hal. 107–116, 1998.

[50] Y. Kristian, H. Takahashi, I. K. E. Purnama, K. Yoshimoto, E. I. Setiawan, E. Hanindito, dan M. H. Purnomo, "A novel approach on infant facial pain classification using multi stage classifier and geometrical-textural features combination," *IAENG International Journal of Computer Science*, vol. 44, no. 1, 2017.

[51] Y. Yamasari, S. M. S. Nugroho, D. F. Suyatno, dan M. H. Purnomo, "Meta-Algoritme Adaptive Boosting untuk Meningkatkan Kinerja Metode Klasifikasi pada Prestasi Belajar Mahasiswa," *Jurnal Nasional Teknik Elektro dan Teknologi Informasi (JNTETI)*, vol. 6, no. 3, 2017.

[52] V. Kazemi dan J. Sullivan, "One millisecond face alignment with an ensemble of regression trees," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, hal. 1867–1874.

[53] B. Xu, N. Wang, T. Chen, dan M. Li, "Empirical evaluation of rectified activations in convolutional network," *arXiv preprint arXiv:1505.00853*, 2015.

[54] H. R. S. and E. P. S. Z. Wang, A. C. Bovik, "Image quality assessment: From error visibility to structural similarity," *IEEE Transactions on Image*

Processing, vol. 13, no. 4, hal. 600–612, 2004.

[55] F. J. Ordóñez dan D. Roggen, “Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition,” *Sensors*, vol. 16, no. 1, hal. 115, 2016.

[56] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, dan R. Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting,” *The Journal of Machine Learning Research*, vol. 15, no. 1, hal. 1929–1958, 2014.

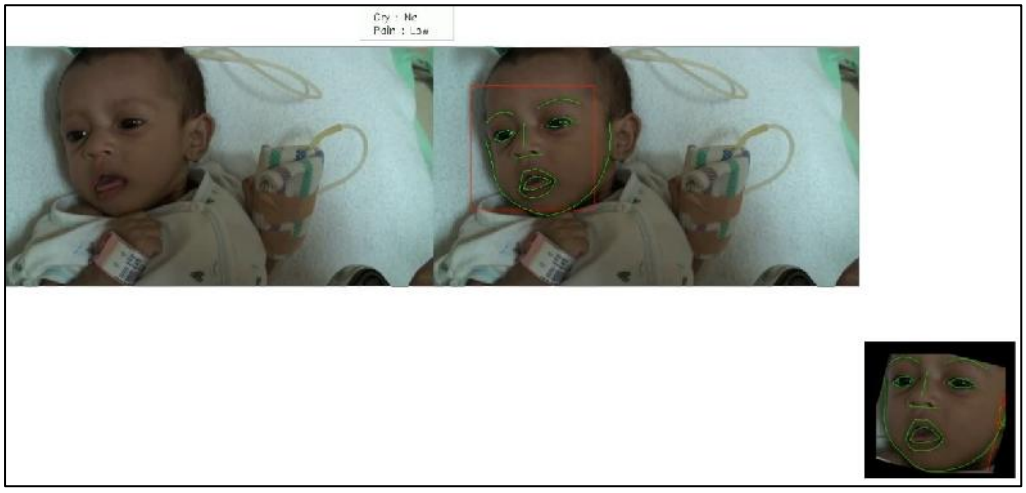
[57] A. Lanitis, C. J. Taylor, dan T. F. Cootes, “Automatic Interpretation and Coding of Face Images Using Flexible Models,” *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, hal. 743–756, 1997.

[58] T. F. Cootes, C. J. Taylor, D. H. Cooper, dan J. Graham, “Active Shape Models-Their Training and Application,” *Computer Vision and Image Understanding*, vol. 61, no. 1. hal. 38–59, 1995.

LAMPIRAN A

Frame-frame Video Hasil Klasifikasi Nyeri dan Tangis

Cry : Yes
Pain : Medium


Cry : No
Pain : Low

Cry : Yes
Pain : Medium

BIODATA PENULIS

Nama	:	Yosi Kristian
NRP	:	07111360010008
Program	:	Doktor
Program Studi	:	Teknik Elektro
Fakultas	:	Fakultas Teknik Elektro Institut Teknologi Sepuluh November Surabaya (ITS)
Tempat/tanggal lahir	:	Tuban, 25 Desember 1981
Agama	:	Katholik
Pekerjaan	:	Dosen Tetap Sekolah Tinggi Teknik Surabaya
Alamat Kantor	:	Kampus Sekolah Tinggi Teknik Surabaya, Jl. Ngagel Jaya Tengah 73-77 Surabaya
Telp	:	031-5027920
Alamat Surat	:	Purimas Regency B8 / 16 Surabaya 60294
Email	:	yosi@stts.edu yosik81@gmail.com yosi13@mhs.ee.its.ac.id
Nama Orang tua	:	Rudiono Sudibyo (almarhum) Sri Rahayu
Nama Isti	:	Maria Bernadette Ria Ardhani
Nama Anak	:	Nick Frederic Kristian (TK St. Clara Surabaya)

Riwayat Pendidikan:

A. Pendidikan Dasar dan Menengah:

1. TK Katolik St. Petrus Tuban – **(1985-1988)**
2. SD Katolik St. Petrus Tuban – **(1988-1994)**
3. SMP Negeri I, Tuban – **(1994-1997)**
4. SMA Negeri I, Tuban – **(1997-2000)**

B. Pendidikan Tinggi:

1. Strata-1 : Sarjana Komputer – (2000-2004)
   Sekolah Tinggi Teknik Surabaya
   Skripsi : Automatic Interpretation on Manual Correction in Scanned Text Document
   Pembimbing : Prof. Dr. Ir. Kuswara Setiawan, M.T.

2. Strata-2 : Magister Komputer – (2006-2008)
   Sekolah Tinggi Teknik Surabaya
   Tesis : Improvement of CORE categorical data clustering Algorithm
   Pembimbing : Prof. Dr. Ir. Kuswara Setiawan, M.T.

3. Strata-3 : Doktor Teknik Elektro – (2013-Sekarang)
   Institut Teknologi Sepuluh November Surabaya (ITS)
   Disertasi : Analisa Citra Wajah Bayi Untuk Deteksi Nyeri Dan Tangis Menggunakan Multi Stage Classification Dan Deep Learning
   Promotor : Prof. Dr. Ir. Mauridhi Hery Purnomo, M.Eng.
   Co-Promotor : Dr. I Ketut Eddy Purnama, ST., MT.

Daftar publikasi selama studi program Doktor:

A. Jurnal Internasional

1. Kristian, Yosi, Hideya Takahashi, I. Ketut Eddy Purnama, Kayo Yoshimoto, Esther Irawati Setiawan, Elizeus Hanindito, and Mauridhi Hery Purnomo. "A Novel Approach on Infant Facial Pain Classification using Multi Stage Classifier and Geometrical-Textural Features Combination." IAENG International Journal of Computer Science 44, no. 1 (Feb 2017).
   ISSN: 1819-9224 (online version); 1819-656X (print version)
   Quartile: Q2
   H Index: 13
   http://www.iaeng.org/IJCS/issues_v44/issue_1/IJCS_44_1_13.pdf

B. Jurnal Nasional Terakreditasi

1. Kristian, Yosi, I. Ketut Eddy Purnama, Effendy Hadi Sutanto, Lukman Zaman, dan Mauridhi Hery Purnomo. “Klasifikasi Nyeri Dan Tangis Pada Bayi Berdasarkan Video Expresi Wajah Menggunakan DCNN Autoencoder dan LSTM” JNTETI Vol. 7, No. 3. (2018)
   ISSN: 2460-5719 (Print), 2301-4156 (Online)

C. Seminar Internasional

1. Kristian, Yosi, Mochamad Hariadi, and Mauridhi Hery Purnomo. "Ideal modified adachi chaotic neural networks and active shape model for infant facial cry detection on still image." In Neural Networks (IJCNN), 2014 International Joint Conference on, pp. 2783-2787. IEEE, Beijing 2014.
   http://ieeexplore.ieee.org/document/6889520/

2. Purnomo, Mauridhi Hery, Yosi Kristian, Endang Setyati, Ulla Delfana Rosiani, and Esther Irawati Setiawan. "Limitless possibilities of pervasive biomedical engineering: Directing the implementation of affective computing on automatic health monitoring system." In *Information Technology and Electrical Engineering (ICITEE), 2016 8th International Conference on*, pp. 1-4. IEEE, 2016. https://ieeexplore.ieee.org/abstract/document/7863314/

Halaman ini sengaja dikosongkan