Klasifikasi Nyeri pada Video Ekspresi Wajah Bayi Menggunakan DCNN *Autoencoder* dan LSTM

Yosi Kristian[1,3], I Ketut Eddy Purnama[1,2], Effendy Hadi Sutanto[3], Lukman Zaman[1,3], Esther Irawati Setiawan[1,3], Mauridhi Hery Purnomo[1,2]

***Abstract*—Babies are still unable to inform the pain they experience, therefore, babies cry when experiencing pain. With the rapid development of computer vision technologies, in the last few years, many researchers have tried to recognize pain from babies expressions using machine learning and image processing. In this paper, a research using Deep Convolution Neural Network (DCNN) Autoencoder and Long-Short Term Memory (LSTM) Network is conducted to detect cry and pain level from baby facial expression on video. DCNN Autoencoder is used to extract latent features from a single frame of baby face. Sequences of extracted latent features are then fed to LSTM so the pain level and cry can be recognized. Face detection and face landmark detection is also used to frontalize baby facial image before it is processed by DCNN Autoencoder. From the testing on DCNN autoencoder, the result shows that the best architecture used three convolutional layers and three transposed convolutional layers. As for the LSTM classifier, the best model is using four frame sequences.**

***Intisari*—Bayi belum dapat menginformasikan rasa nyeri yang dialami, karena itu bayi menangis saat 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 makalah ini diteliti pemanfaatan *Deep Convolution Neural Network* (DCNN) *Autoencoder* dan *Long-Short Term Memory* (LSTM) *Network* untuk deteksi tangis dan tingkat nyeri pada video wajah bayi. DCNN *Autoencoder* berguna untuk melakukan ekstraksi *latent feature* dari satu *frame* wajah bayi. Deretan *latent feature* ini kemudian diumpankan ke LSTM untuk dikenali tangis dan tingkat nyerinya. Selain itu, digunakan juga teknik *face detection* dan *face landmark detection* untuk meluruskan/menegakkan wajah bayi sebelum diproses oleh DCNN *autoencoder*. Dari pengujian DCNN *autoencoder*, didapatkan hasil terbaik dengan menggunakan tiga *convolutional* layer dan tiga *transposed convolutional layer*. Sedangkan untuk LSTM *classifier*, model terbaik didapatkan dalam percobaan dengan empat runtun *frame*.**

***Kata Kunci*—Neural network, DCNN, LSTM, autoencoder, klasifikasi nyeri, ekspresi wajah bayi.**

[1] *Departemen Teknik Elektro, Fakultas Teknologi Elektro, Institut Teknologi Sepuluh Nopember, Kampus ITS, Sukolilo, Surabaya 60111, Jawa Timur, Indonesia (tlp: 031-5947302, fax: 031-5931237, e-mail: yosi13@mhs.ee.its.ac.id, ketut@ee.its.ac.id, hery@ee.its. ac.id)*

[2] *Departemen Teknik Komputer, Fakultas Teknologi Elektro, Institut Teknologi Sepuluh Nopember, Kampus ITS, Sukolilo, Surabaya 60111, Jawa Timur, Indonesia (tlp: 031-5922936, ketut@ee.its.ac.id, hery@ee.its.ac.id)*

[3] *Departemen Teknik Informatika, Sekolah Tinggi Teknik Surabaya, Jl. Ngagel Jaya Tengah 73-77, Surabaya, Jawa Timur, Indonesia 60284 (tlp:031-5027920; e-mail: yosi@stts.edu, tancejang@gmail.com, lz@stts.edu, esther@stts.edu)*

I. PENDAHULUAN

Manusia dewasa dapat menunjukkan atau memperlihatkan rasa sakitnya dengan berbagai macam cara [1]. Salah satu cara yang paling sering digunakan untuk mengukur tingkat sakit pasien adalah dengan laporan pasien itu sendiri, sehingga tidak diperlukan keahlian khusus maupun teknologi lanjut untuk mengetahui tingkat sakit yang dirasakan oleh pasien tersebut [2]. Namun, hal ini tidak dapat diterapkan pada pasien bayi karena bayi belum dapat berbicara sehingga tidak dapat mengungkapkan rasa sakitnya dengan jelas.

Untuk mendeteksi rasa sakit pada bayi, dapat digunakan ekspresi wajah bayi sebagai salah satu sumber informasi. Dibandingkan dengan respons secara tingkah laku maupun psikis, ekspresi wajah untuk rasa sakit lebih spesifik dan konsisten dalam menunjukkan rasa sakit [3]. Inilah sebabnya pada beberapa instrumen untuk mengukur nyeri pada bayi, seperti CRIES [4], *Face, Legs, Activity, Cry, and Consolability* (FLACC) [3], dan *Modified Infant Pain Scale* (MIPS) [5], lebih fokus pada ekspresi wajah bayi.

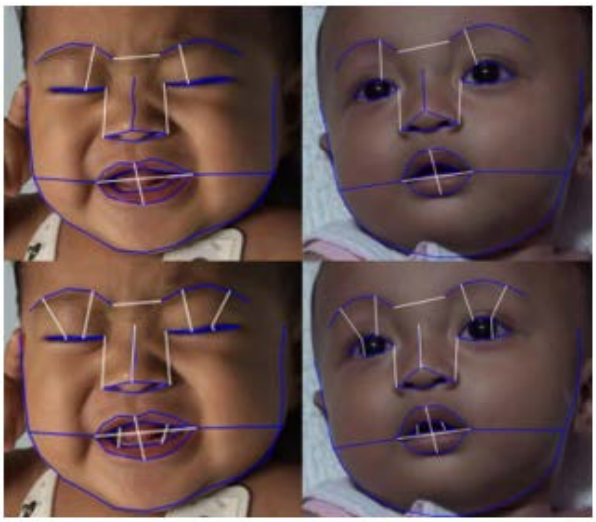
Pada banyak rumah sakit di Indonesia, observasi dan penilaian FLACC merupakan metode pengukuran yang sangat umum digunakan untuk mengukur rasa sakit pada bayi [6]. Meskipun demikian, dengan banyaknya jumlah bayi dan terbatasnya jumlah dokter dan perawat pada rumah sakit, sangat sulit bagi pihak rumah sakit untuk menjaga dan mengawasi para bayi secara terus menerus. Untuk itulah, dibutuhkan sebuah sistem cerdas yang dapat mendeteksi rasa sakit pada bayi secara otomatis dan memiliki tingkat akurasi yang tinggi.

Pada beberapa penelitian terdahulu ada beberapa metode yang digunakan untuk mendeteksi rasa sakit pada bayi, yaitu menggunakan suara tangis bayi. Metode ini menjadi tidak efektif apabila diimplementasikan di ruangan dengan jumlah bayi lebih dari satu, karena bayi-bayi dapat menangis bersamaan. Beberapa penelitian lain sudah memanfaatkan citra wajah bayi dan menggunakan *handcrafted feature* dengan bantuan *machine learning*, tetapi hingga saat ini masih belum ada penelitian yang mencoba mengklasifikasikan nyeri pada ekspresi wajah bayi menggunakan *deep learning*. Dengan memanfaatkan *deep learning*, diharapkan sistem akan mendapatkan hasil yang lebih baik karena tidak perlu lagi mencari fitur secara manual. Maksud dari kata *deep* dalam *deep learning* adalah jumlah *layer* dalam *neural network* yang berlapis-lapis.

Secara singkat, makalah ini memiliki kontribusi utama sebagai berikut.

1. *Dataset* yang digunakan disebut dengan *Infant Facial Pain Level Dataset* (IFPaLD).
2. Mendesain sebuah *Deep Convolution Neural Network* (DCNN) *autoencoder* untuk meng-hasilkan *latent features* yang cocok.

3. Mendesain dan membuat *long short term memory network* (LSTM) untuk klasifikasi tangis dan tingkat nyeri bayi berdasarkan deretan *frame*.



Gbr. 1 Contoh ekstraksi fitur penelitian sebelumnya.

II. PENELITIAN DETEKSI NYERI PADA WAJAH

Pada penelitian sebelumnya telah digunakan pola akustik pada tangisan bayi untuk mendeteksi nyeri [6]. Selama percobaan, dibuat 56 video yang diambil dari 28 bayi sebelum dan sesudah melakukan prosedur operasi. Skala FLACC dinilai dan hormon *cortisol* dari *saliva* setiap bayi diukur sebelum dan sesudah operasi. Video inilah yang digunakan sebagai sumber *dataset* pada percobaan ini.

Pada tahun 2008, diperkenalkan beberapa pasang jarak titik-titik penting wajah yang digunakan untuk mengukur nyeri pada bayi yang disebut *point pair difference* (PPD) [7]. Uji coba dilangsungkan berdasarkan 57 bayi yang berjenis kelamin laki-laki dan perempuan dari berbagai etnis. Disimpulkan bahwa rasa nyeri tidak berkorelasi terhadap jenis kelamin atau etnis.

Pada tahun 2013, telah dimanfaatkan fitur tekstur berbasis *Local Binary Patterns* (LBP) untuk mendeteksi nyeri pada wajah bayi [8]. Metodenya membagi citra wajah bayi menjadi 7 x 9 *overlapping cell*, dengan setiap sel memiliki ukuran 25 x 25 dimensi piksel. Dari setiap sel ini dilakukan ekstraksi LBP, kemudian dilakukan reduksi *Uniform* LBP sehingga mengurangi jumlah fitur dari 256 menjadi 59 untuk setiap selnya.

Pada penelitian yang selanjutnya, diterbitkan sebuah makalah mengenai deteksi tangis pada wajah bayi [9]. Pada penelitian ini dikenalkan fitur geometrik yang didapatkan dengan menggunakan *Active Shape Model* (ASM) untuk mendeteksi tangis pada wajah bayi. Juga digunakan *Ideal Modified Adachi Chaotic Neural Network* (Ideal-M-AdNN) yang cukup unik. Metode ini mampu menghasilkan akurasi yang cukup tinggi untuk deteksi tangis pada wajah bayi.

Pada penelitian lainnya yang bertujuan untuk mendeteksi nyeri pada orang dewasa, dibuat sebuah sistem yang memanfaatkan *Active Appearance Model* (AAM) dan SVM [2]. Digunakan *dataset McMaster Shoulder Pain Expression Archive* yang mengandung 129 video. AAM digunakan untuk mengekstraksi fitur bentuk dan tekstur wajah secara langsung.

Pada tahun 2017, dilakukan modifikasi dari PPD Schiavenato yang disebut *Extended Normalized Point Pair Difference* (ENPPD) sebagai salah satu fitur yang digunakan untuk deteksi tangis dan nyeri pada bayi [10]. ENPPD ditampilkan dengan menggunakan garis putih pada Gbr. 1, sedangkan garis biru merupakan *facial landmark* yang dihasilkan, diperoleh dengan menggunakan ASM. Pada penelitian ini, digabungkan dua jenis fitur: fitur geometri dan fitur tekstur untuk dapat mengenali tingkat nyeri pada wajah bayi.

Meskipun memberikan hasil yang cukup memuaskan, 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 makalah ini diterapkan ekstraksi fitur secara *feature learning* memanfaatkan *autoencoder*. Pada bagian berikutnya dijelaskan tentang *Convolutional Neural Network* (CNN) dan *Convolutional Autoencoder* (CAE) yang dimanfaatkan dalam makalah ini.

III. *CONVOLUTIONAL NEURAL NETWORK* DAN *CONVOLUTIONAL AUTOENCODER*

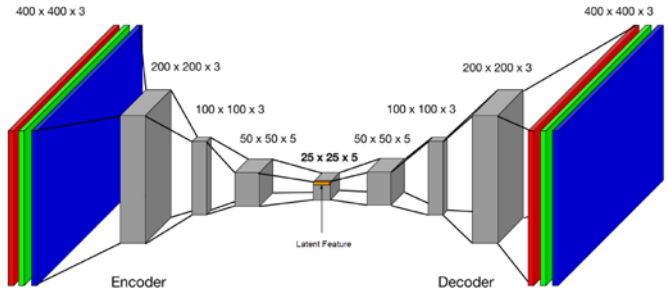
Artificial Neural Network (ANN) merupakan sebuah model jaringan yang menerima masukan berbentuk vektor satu dimensi yang mentransformasikan masukan menjadi keluaran dengan cara mempropagasikan hasil *dot product* yang diaktivasi pada *layer-layer* yang saling berhubungan [11]. Setiap neuron pada sebuah *layer* berhubungan penuh dengan semua neuron di *layer* sebelumnya dan memiliki bobot independen. Arsitektur ini disebut *multi layer perceptron* (MLP) [12]. *Layer* ini disebut sebagai *fully-connected layer* dan dilatih dengan algoritme *Backpropagation* [13].

Pada pemrosesan data yang berukuran besar, *fully-connected layer* dapat menyebabkan beberapa masalah. Sebagai contoh, citra-citra pada *dataset* CIFAR-10 memiliki ukuran 32x32x3 [14]. Apabila data ini diumpankan pada sebuah MLP, maka satu neuron pada *layer* pertama akan memiliki setidaknya 32x32x3=3.072 bobot. Angka ini akan meningkat dengan cepat bila digunakan banyak neuron, banyak *laye*r, atau masukan dengan dimensi yang lebih besar. Banyaknya jumlah bobot yang harus dilatih oleh ANN akan menyebabkan terjadinya *overfitting* [11].

Selain jumlah bobot yang sangat besar, *fully-connected layer* juga tidak memperhitungkan struktur spasial dari sebuah citra. Sebagai contoh, masukan piksel pada posisi yang jauh dan dekat akan memiliki kedudukan yang sama. CNN mengatasi dua masalah ini dengan menerapkan *weight sharing* dan mekanisme konvolusi filter. Sebuah jaringan yang dapat memperhitungkan struktur spasial dari citra akan memiliki kinerja yang lebih baik untuk menangani masukan berupa citra [11].

Ide CNN sebenarnya sudah muncul pada tahun 1970, tetapi makalah yang merupakan dasar dari CNN modern baru muncul pada tahun 1998. Makalah tersebut dibuat oleh Yann

Lecun, Leon Bottou, Yoshua Bengio, dan Patrick Haffne dengan judul "*Gradient-based learning applied to document recognition*" [15].



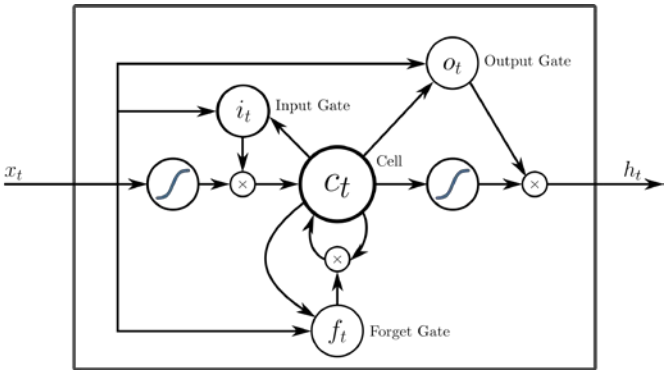
Gbr. 2 Contoh arsitektur CAE.

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* dengan akurasi yang selisih jauh dari tahun sebelumnya [16]. Saat ini CNN sendiri sudah banyak digunakan untuk klasifikasi pada berbagai bidang karena tingkat kesuksesannya yang cukup besar. Pada penelitian-penelitian terkini, CNN telah di-manfaatkan untuk klasifikasi sepeda motor [17], deteksi pejalan kaki [18], dan klasifikasi umur dan jenis kelamin manusia [19].

Terdapat tiga tipe *layer* untuk membuat arsitektur ConvNet, yaitu *Convolutional Layer* (CONV), *Pooling Laye*r (POOL), dan *Fully Connected* (FC) [20]. Layer-layer tersebut dapat disusun menjadi satu untuk membentuk suatu arsitektur CNN. Sebagai 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.

Selain CNN, terdapat pula suatu jaringan *(network)* yang cukup terkenal. Jaringan ini dilatih dengan metode *supervised learning*, tetapi tidak membutuhkan kelas dari data sebagai target. Jaringan ini terkenal dengan nama *autoencoder* atau *auto-associator*. *Autoencoder* adalah model *neural network* yang memiliki masukan dan keluaran yang sama, yang ilustrasinya ditunjukkan pada Gbr. 2 [21]. *Autoencoder* memiliki bobot *(weight)* yang sama antara *encoder* dan *decoder*-nya, yang biasa disebut dengan *tied weight. Autoencoder* berusaha untuk mempelajari data masukan dan berusaha untuk merekonstruksi kembali masukan tersebut, tetapi dengan informasi yang lebih sedikit. Informasi ini biasa disebut sebagai *latent feature.*

Kegunaan *autoencoder* adalah untuk mengurangi dimensi dari fitur yang didapat dari suatu masukan atau bisa disebut dengan *Dimensionality Reduction* [22]. Cara memanfaatkan *autoencoder* adalah dengan memotong jaringan pada bagian *encoder*-nya, kemudian *latent feature* yang dihasilkan diumpankan ke *classifier*. *Autoencoder* juga dapat digunakan untuk *denoising* atau menghilangkan derau pada suatu data masukan. *Autoencoder* yang menghilangkan derau disebut juga dengan *denoising autoencoder* [23].



Gbr. 3 Ilustrasi unit LSTM.

Metode paling umum yang digunakan untuk masalah *Dimensionality Reduction* adalah *Principal Component Analysis* (PCA) [24]. PCA cukup unggul sebagai metode reduksi dimensi karena sederhana untuk diimplementasikan dan mampu mengeliminasi korelasi antar variabel masukan. Akan tetapi, PCA hanya mereduksi data secara linear, dan tidak memperhatikan hubungan antar data secara spasial.

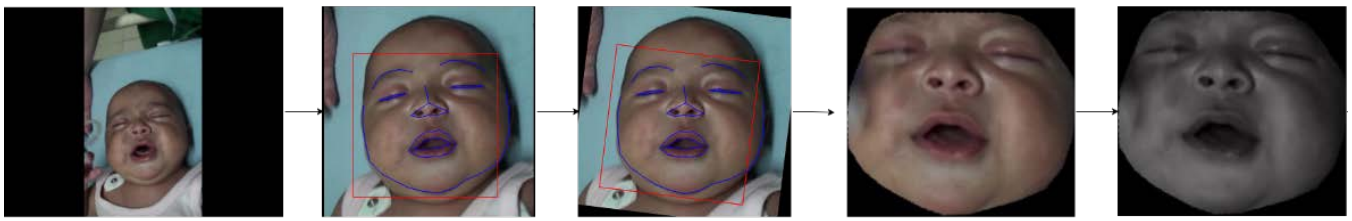
Sama seperti PCA, sebuah *neural network autoencoder* konvensional yang menerima masukan berbentuk citra juga tidak dapat memperhitungkan informasi spasial dari data masukan, mengingat bahwa bentuk *layer* dari *autoencoder* adalah *flat layer*. Oleh karena itu, terciptalah ide yang bernama CAE. CAE memiliki konsep yang sama dengan CNN dan memiliki tujuan dan 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 merekonstruksinya kembali. Konsep *tied weights* pada *autoencoder* juga diterapkan pada CAE, yaitu filter untuk bagian *encoder* sama dengan bagian *decoder*, hanya saja urutanya terbalik [25].

Jadi, pada makalah 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 bagian berikutnya dijelaskan tentang arsitektur RNN yang dipakai, yakni LSTM.

IV. LONG SHORT TERM MEMORY NETWORK

Neural network juga dapat digunakan untuk memproses data sekuensial. Sebagai contoh, *neural network* dapat memprediksi kata yang akan muncul berdasar kemunculan kata-kata sebelumnya. RNN adalah jenis *neural network* yang digunakan untuk memproses data sekuensial semacam itu [26]. Pada RNN, selain rangkaian data sesuai urutan yang menjadi masukan, keluaran RNN pada suatu tahap juga akan dimasukkan kembali menjadi masukan tambahan untuk tahap selanjutnya. Mekanisme ini yang membuat RNN dapat melakukan deduksi berdasar urutan data yang telah diproses pada langkah-langkah sebelumnya. Panjang runtun *(sequence)* pada RNN disebut juga dengan *sequence length* atau *timesteps*.



Gbr. 4 Tahap praproses.

Secara teori, RNN dapat memproses data sekuensial sepanjang apapun. Jumlah data sebelumnya yang menjadi *feedback* masukan untuk tahap selanjutnya adalah sepanjang pola masukan yang hendak dikenali. Namun, pada praktiknya, 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* [27]. Masalah ini terjadi bila propagasi *error* ke *layer* awal pada saat pelatihan mengalami penurunan nilai secara eksponensial dan membuat bobot *layer-layer* awal sulit ter-*update*.

LSTM adalah varian dari RNN yang dapat mengatasi masalah *vanishing gradient*. LSTM diciptakan oleh Sepp Hochreiter dan Jürgen Schmidhuber pada tahun 1997 [28]. Saat ini LSTM 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 unit LSTM disebut sebagai LSTM *network* [21].

LSTM *network* memiliki arsitektur yang sama dengan RNN biasa, tetapi komponen dari unit yang digunakan cukup berbeda. Pada Gbr. 3, terlihat bahwa unit LSTM 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* [24]. *Cell state gate* berfungsi untuk meneruskan *hidden state* dari *layer* sebelumnya. *Gate* inilah 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 dari *hidden state* yang akan diteruskan sebagai masukan pada *hidden layer* selanjutnya.

V. *INFANT FACIAL PAIN LEVEL DETECTION DATASET*

IFPaLD merupakan *dataset* yang berhasil didapatkan dalam penelitian ini. Video yang menjadi dasar pembuatan *dataset* IFPaLD diambil dari 28 bayi, yaitu 16 laki-laki dan 12 perempuan, sebelum dan setelah menjalani prosedur operasi. Video-video ini diambil dalam sebuah penelitian pada tahun 2013 [6]. Dalam penelitian tersebut, setiap bayi dalam video diukur FLACC dan hormon *cortisol*-nya untuk menentukan tingkat nyeri yang dialami bayi. Tingkat nyeri dibedakan menjadi tiga, yaitu nyeri berat, nyeri sedang, dan tidak nyeri.

Citra-citra pada *dataset* ini berformat *.jpg*, diambil dari video dengan frekuensi 3 *frame* per detik. Total *dataset* terdiri atas 15.105 citra dan dibagi menjadi 11.800 data pelatihan *(training)* dan 3.305 data pengujian *(testing)*. IFPaLD tidak memiliki *timestep* yang terbatas, panjang runtun pada *dataset* ini adalah sepanjang *frame* video wajah bayi yang diekstraksi. Pengguna *dataset* ini dapat menentukan *timestep* atau panjang runtun yang dipakai dan disesuaikan dengan arsitektur RNN yang digunakan. Dalam *dataset* disediakan citra yang telah di-*crop* dan yang belum diproses. Untuk lebih jelasnya, *dataset* ini dapat dilihat pada laman http://dataset.stts.edu/IFPaLD.

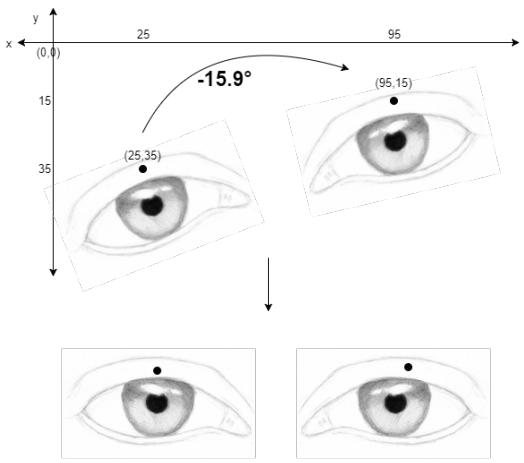
Pada bagian berikutnya dijelaskan proses-proses yang dilakukan agar dapat mengklasifikasikan nyeri dan tangis dari deretan *frame* wajah bayi yang diekstrak dari video.

VI. KLASIFIKASI NYERI DAN TANGIS PADA WAJAH BAYI

Dengan memanfaatkan *deep learning*, maka tidak perlu dilakukan ekstraksi fitur secara *handcrafted* seperti yang dilakukan dalam penelitian sebelumnya [2], [8], [10]. Bagian yang melakukan *feature learning* di sini adalah CAE. 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 masukan yang konsisten dalam segi ukuran, pewarnaan, dan area yang dianggap penting, dalam penelitian ini tetap dilakukan tahap praproses yang dijelaskan dalam subbagian berikutnya.

A. *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* (ROI), dan melakukan *grayscaling*. Ilustrasi tahap-tahap praproses yang dilakukan ditunjukkan pada Gbr. 4.



Gbr. 5 Ilustrasi rotasi berdasarkan posisi mata.

Tahap deteksi wajah berguna untuk mendapatkan area wajah secara persegi, sehingga proses selanjutnya tidak dilakukan pada seluruh area gambar, tetapi hanya di area wajah saja. Untuk tahap deteksi wajah, digunakan implementasi algoritme Viola Jones [29] pada DLIB *library* yang memanfaatkan *adaptive boosting* seperti pada penelitian sebelumnya [30].

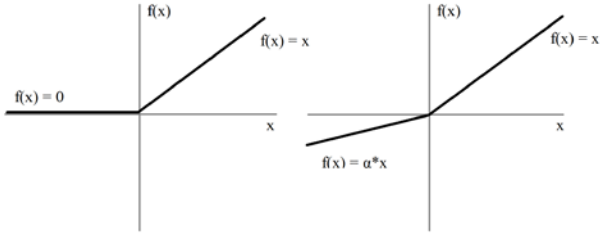
Setelah mendapatkan persegi area wajah, gambar di-*crop* sesuai dengan area wajah yang diperluas 150%. Setelah itu, dilanjutkan dengan proses deteksi poin-poin *landmark* wajah. Tujuan dari tahapan ini adalah untuk mendapatkan titik-titik pada bagian mata, bagian alis, dan bagian dagu. Bagian mata berguna sebagai pedoman pada tahap selanjutnya, yakni rotasi wajah. Sedangkan titik dagu dan alis sebagai pedoman pengambilan ROI. Untuk melakukan deteksi titik-titik penting wajah, digunakan implementasi DLIB dari algoritme Kazemi dkk. [31].

Setelah melalui tahap deteksi titik-titik penting wajah, dilanjutkan dengan proses rotasi citra. Citra dirotasi menurut sudut dari posisi kedua titik *landmark* pada mata dengan garis horizontal. Gbr. 5 merupakan ilustrasi dari rotasi wajah berdasarkan titik *landmark* pada mata ini.

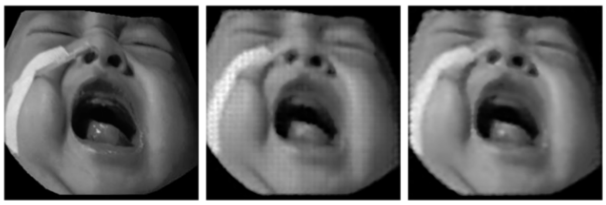
Tujuan dari tahap rotasi gambar berdasarkan posisi mata adalah agar gambar yang diumpan ke dalam *network* merupakan gambar yang memiliki standardisasi, yaitu wajah yang tegak lurus. Setelah melalui tahap tersebut, dilanjutkan dengan pengambilan ROI berdasarkan alis dan dagu. Setelah melalui tahap tersebut, dilanjutkan dengan mengubah gambar menjadi *grayscale.* Setelah itu, gambar siap diumpankan ke proses *autoencoder*.

B. Tahap Convolution Autoencoder

Untuk melatih sebuah CAE yang baik, diperlukan beberapa parameter yang harus diatur. Hal-hal yang harus dipertimbangkan adalah tingkat reduksi yang dilakukan, jumlah *layer* konvolusi yang diterapkan, serta jumlah dan dimensi spasial filter yang dipakai. Filter yang digunakan pada jaringan ini berukuran 3x3, dengan setiap CAE memiliki *depth* atau jumlah filter sebanyak sepuluh. *Stride* yang digunakan adalah sebesar 2 dan menggunakan *same padding*. *Stride* adalah jarak lompatan filter dalam proses konvolusi. Dengan digunakan *stride* sebesar 2 dimensi tereduksi menjadi setengah dimensi masukan. Sedangkan *same padding* menambahkan nol pada pinggir masukan secara seimbang kanan-kiri, dan atas-bawah, agar filter konvolusi dapat bekerja dengan tepat.



Gbr. 6 ReLU dibandingkan dengan *Leaky* ReLU.



Gbr. 7 Kinerja *autoencoder*. (a) Citra masukan, (b) Hasil *autoencoder* dengan *loss MSE*, (c) Hasil *autoencoder* dengan *loss SSIM*.

Sedangkan *layer* aktivasi yang dicoba adalah *rectifier linear unit* (ReLU) dan *Leaky* ReLU [32]. Persamaan ReLU ditunjukkan pada (1) dan *Leaky* ReLU pada (2). Dalam makalah ini digunakan parameter *a* sebesar 0,2. Perbandingan grafik ReLU dan *Leaky* ReLU diperlihatkan pada Gbr. 6. *Leaky* ReLU digunakan dalam makalah ini karena penggunaan ReLU dapat menyebabkan *Dying ReLU Unit* (unit ReLU yang selalu mengeluarkan keluaran 0).

$$f(x) = \begin{cases} x, \ if \ x > 0 \\ 0, \ lainya \end{cases} \quad (1)$$

$$f(x) = \begin{cases} x, \ if \ x > 0 \\ ax, \ lainya \end{cases} \quad (2)$$

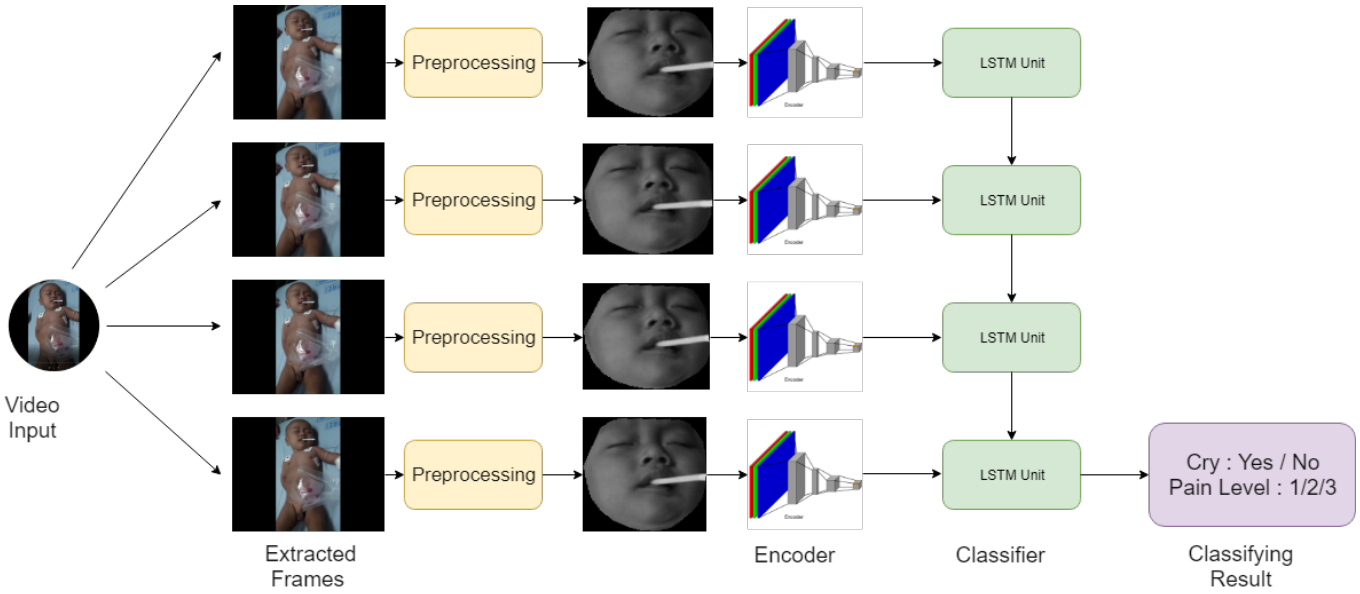
Pada tahap *autoencoder,* sebuah model dapat dikatakan baik apabila masukan dan keluran yang dihasilkan memiliki kemiripan yang tinggi. Penggunaan *mean square error* (MSE) atau *peak signal to noise ratio* (PSNR) sebagai alat ukur kemiripan dua buah citra ternyata kurang baik karena lebih fokus pada selisih citra per piksel sehingga mengakibatkan hasil *autoencoder* lebih *noisy*. Gbr. 7 memperlihatkan kinerja *autoencoder*.

Dalam makalah ini, untuk mengukur tingkat kemiripan dua buah citra, digunakan *Structural Similarity Index* (SSIM), yang melihat perbedaan dua citra lebih global [33]. Persamaan SSIM dituliskan pada (3).

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \quad (3)$$

$$c_1 = (0{,}01 * L)^2 \quad (4)$$

$$c_2 = (0{,}03 * L)^2 \quad (5)$$



Gbr. 8 Ilustrasi sistem klasifikasi tangis dan nyeri dari video wajah bayi.

Dalam (3), *x* dan *y* adalah dua citra yang berbeda, denganμ_x dan μ_y merupakan rata-rata *gray level* untuk citra *x* dan *y*. Simbol σ_x^2 dan σ_y^2 adalah varians antara dua buah citra, sedangkan σ_{xy} adalah *covariance* dari dua buah citra. c_1 pada (4) dan c_2 pada (5) adalah konstanta yang digunakan untuk menstabilkan pembagian dengan nilai kecil dan *L* adalah konstanta yang didapat dari $2^{\text{jumlah bit}}$-1. Dalam makalah ini, *L* bernilai 255.

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

Setelah *autoencoder* terlatih, model yang digunakan adalah *convolutional encoder*, yang merupakan potongan setengah bagian depan dari CAE. Dalam makalah ini, bagian akhir dari *encoder* memiliki dimensi 8x8x10. Inilah yang merupakan *latent feature*. Setelah diperoleh jaringan yang dapat mengubah gambar menjadi data dengan dimensi yang lebih kecil, dilanjutkan dengan memasukkan data tersebut ke dalam *classifier*.

C. Tahap Klasifikasi Menggunakan LSTM

Alasan digunakannya RNN adalah karena masukan merupakan deretan gambar yang sekuensial sehingga urutan waktu menjadi informasi yang substansial. Dengan memanfaatkan RNN, informasi runtun 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 [26], [34].

LSTM menerima hasil *encoding* dari *autoencoder* atau jaringan sebelumnya. Sebelum data diumpankan ke dalam LSTM, data dari *encoder* harus diubah ke *flat layer* terlebih dahulu karena masukan dari LSTM merupakan *tensor* satu dimensi. LSTM yang digunakan memiliki panjang runtun dan jumlah *hidden unit* yang bervariasi. Serangkaian percobaan dilakukan untuk membuktikan jumlah runtun dan jumlah *hidden unit* terbaik.

Dalam implementasinya, apabila LSTM memiliki jumlah runtun empat, maka program akan mengambil *frame* video yang berurutan sebanyak empat, dengan masing-masing *frame* akan diumpankan ke dalam *convolutional encoder*. Hasilnya adalah empat set *latent feature* yang diumpankan ke dalam LSTM berdasarkan waktu. Ilustrasi sistem dari masukan hingga klasifikasi LSTM diperlihatkan pada Gbr. 8.

Setelah data diumpankan ke dalam LSTM, LSTM mengeluarkan label untuk data runtun yang diumpankan. Hal itu diulang terhadap semua *frame* runtun yang ada pada video masukan. Setelah itu, semua *frame* yang telah diklasifikasikan tersebut dikonstruksi kembali menjadi sebuah video yang memiliki informasi kelas nyeri dan tangis di dalamnya. Video inilah yang menjadi keluaran dari sistem. Contoh potongan dari video tersebut ditunjukan pada Gbr. 9.

VII. UJI COBA

Dalam bagian ini, dijelaskan tentang uji coba yang dilakukan untuk mendapatkan model terbaik dengan cara mengganti parameter-parameter, seperti jumlah *layer*, *learning rate*, dan jumlah *frame*. Alat ukur yang digunakan adalah akurasi, yang dituliskan pada (6), dan *F1 Score*, yang dituliskan pada (7). *F1 Score* adalah ukuran level sebuah *classifier* dapat mengklasifikasikan sebuah kelas. *F1 Score* dapat digunakan untuk mengukur kinerja multi kelas dengan cara melakukan rata-rata berbobot (*weighted average*) hasil *F1 Score* semua kelas. Bobot di sini adalah jumlah data pendukung (*support*) masing-masing kelas.

Tahap pertama dari rangkaian uji coba yang dilakukan adalah uji coba rekonstruksi citra menggunakan DCNN *autoencoder*. Pada tahap ini, yang dilakukan adalah mengganti beberapa parameter pada DCNN *autoencoder*. Parameter yang dicoba adalah variasi *learning rate*, fungsi aktivasi, dan jumlah *hidden layer*.

TABEL I
HASIL UJI COBA *AUTOENCODER*

Learning Rate	Fungsi Aktivasi	Jumlah Percobaan	Jumlah *Layer Convolutional*	Jumlah *Layer Deconvolutional*	*Minimum Loss*	*Maximum Loss*
0,001	ReLU	4	4	4	0,581	0,609
0,01	LReLU	2	4	4	0,145	0,155
0,01	**LReLU**	**8**	**3**	**3**	**0,037**	**0,2**
0,01	LReLU	3	2	2	0,038	0,47

TABEL II
HASIL UJI COBA *LONG SHORT TERM MEMORY*

Sequence Length	Jumlah Hidden Unit	Probabilitas Dropout	*Epoch*	Waktu Pelatihan (detik)	*Loss*	Akurasi Tangis	Akurasi Nyeri	*F1 Score* Tangis	Rata-Rata *F1 Score* Nyeri	*Weighted Average F1 Score* Nyeri
3	512	0,5	620	9.443.24	0,037	0,91	0,90	0,898	0,706	0,880
4	512	0,5	157	3.347.50	0,514	0,92	0,91	0,907	0,715	0,892
4	**1.024**	**0,5**	**249**	**5.229.71**	**0,30**	**0,94**	**0,94**	**0,930**	**0,788**	**0,924**
4	2.048	0,5	254	6.967.32	0,03	0,92	0,88	0,911	0,65	0,852
5	1.024	0,5	708	26.459.86	0,34	0,79	0,73	0,791	0,57	0,753

TABEL III
HASIL UJI COBA *DROPOUT LAYER* PADA JARINGAN LSTM

Probabilitas *Dropout*	*Epoch*	Waktu (detik)	*Loss*	Akurasi Tangis	Akurasi Nyeri	*F1 Score* Tangis	Rata-rata *F1 Score* Nyeri	*Weighted Average F1 Score* Nyeri
0,6	200	4.820,20	0,63	0,76	0,88	0,784	0,757	0,906
0,5	249	5.229,71	0,30	0,94	0,94	0,930	0,788	0,924
0,4	119	2.756,24	0,49	0,90	0,88	0,921	0,760	0,907
0,25	**164**	**3.803,86**	**0,29**	**0,92**	**0,95**	**0,912**	**0,876**	**0,949**
0,0	291	6.836,63	0,35	0,92	0,90	0,905	0,805	0,902

$$Akurasi = \frac{TruePositive}{TotalData} \quad (6)$$

$$F1Score = \frac{2*Precission*Recall}{Precission+Recall} \quad (7)$$

$$Precission = \frac{TruePositive}{TruePositive+FalsePositive} \quad (8)$$

$$Recall = \frac{TruePositive}{TruePositive+FalseNegative} \quad (9)$$

Pada proses pelatihan DCNN *autoencoder*, semua filter memiliki ukuran 3x3, *stride* 2, dan *depth* 10. Uji Coba DCNN *autoencoder* dilakukan dengan menggunakan 879 citra sebagai set pelatihan dan 121 citra sebagai set pengujian. Semua uji coba yang dilakukan untuk *autoencoder* pada makalah ini ditampilkan pada Tabel I. Didapatkan bahwa *autoencoder* terbaik menggunakan *Leaky* ReLU serta menggunakan tiga *layer Conv* dan tiga *layer Deconv*.

Setelah dilakukan 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 dilakukan dengan mengganti beberapa *hyperparameter* LSTM, menambah dan mengurangi jumlah *hidden unit* yang ada, serta mengubah panjang runtun atau *time steps*. Sistem dirancang untuk tidak memiliki panjang runtun yang terlalu panjang, mengingat jumlah data yang terbatas akibat dari kinerja *face detection* yang belum mencapai 100%.

Pada semua uji coba LSTM yang dilakukan, tekniknya adalah dengan membagi *dataset* pelatihan menjadi 20% data validasi dan 80% data pelatihan secara acak. Untuk menghindari *overfit*, digunakan regularisasi L2 dengan parameter *lambda* sebesar 0,001. Tabel II merupakan konfigurasi dan hasil semua uji coba LSTM yang dilakukan.

Setelah melakukan percobaan sebelumnya, didapatkan model terbaik dengan panjang runtun 4 dan jumlah *hidden unit* 1024. Percobaan berikutnya dilanjutkan dengan mencoba mengubah probabilitas *dropout* untuk mengurangi *overfitting*, mengacu pada penelitian sebelumnya [35]. Nilai probabilitas *dropout* menunjukkan jumlah unit yang di-*drop* secara acak untuk tiap *layer*. Nilai probabilitas *dropout* = 0,0 artinya tidak ada unit yang di-*drop*. Hasil uji coba pengaruh *dropout layer* terhadap LSTM ditunjukkan pada Tabel III. 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 lebih memperjelas hasil yang diperoleh, ditampilkan *confusion matrix* dari model terbaik untuk deteksi tangis pada Tabel IV dan *confusion matrix* untuk klasifikasi nyeri pada Tabel V. Dari Tabel IV tampak bahwa sistem dapat mengklasifikasikan tangis dengan baik. Sedangkan dari Tabel V tampak bahwa sistem dapat mengklasifikasikan tingkat nyeri tinggi dan tidak nyeri dengan baik, tetapi mengalami kesulitan dalam mengklasifikasikan tingkat nyeri sedang.

Gbr. 9 Contoh potongan video keluaran dari sistem.

TABEL IV
CONFUSION MATRIX DARI DETEKSI TANGIS

		Prediksi		
		Menangis	**Tdk Menangis**	**Total**
Data Riil	Menangis	**1.137**	35	1.172
	Tdk Menangis	137	**1.674**	1.811
	Total	1.274	1.709	**2.983**

TABEL V
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	**1.760**	1.792
	Total	1.049	152	1.782	2.983

TABEL VI
PERBANDINGAN DENGAN PENELITIAN SEBELUMNYA

Metrik	**Penelitian Ini**	**Penelitian Sebelumnya**
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 ditunjukkan pada Tabel VI [10]. Dari tabel tampak bahwa kinerja penelitian ini dalam klasifikasi tangis masih sedikit di bawah penelitian sebelumnya. Hal ini kemungkinan disebabkan karena *dataset* yang digunakan pada penelitian ini juga lebih banyak dari penelitian sebelumnya. Pada kesimpulan penelitian sebelumnya, penelitian tersebut fokus dalam mengklasifikasikan tingkat nyeri tinggi. Hasil terbaik pada penelitian ini masih sedikit lebih baik daripada penelitian sebelumnya, khusus untuk klasifikasi derajat nyeri tinggi pada bayi.

VIII. KESIMPULAN

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

Dari hasil uji coba *autoencoder*, dapat dilihat bahwa hasil terbaik didapatkan dengan *convolutional layer* sebanyak tiga *layer* dan *deconvolutional layer* sebanyak tiga *layer*. Dari hasil pengujian juga dapat disimpulkan bahwa fungsi aktivasi *leaky* ReLU 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, tampak bahwa hasil terbaik untuk klasifikasi tingkat nyeri didapatkan dengan panjang runtun 4, *hidden unit* 1.024, 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.

REFERENSI

[1] K. M. Prkachin, N. A. Currie, dan K. D. Craig, "Judging Nonverbal Expressions of Pain.," *Can. J. Behav. Sci. Can. des Sci. du Comport.*, 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," *Syst. Man, Cybern. Part B Cybern. IEEE Trans.*, Vol. 41, No. 3, hal. 664–674, 2011.

[3] 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, No. 3, hal. 293–297, 1997.

[4] S. W. Krechel dan J. Bildner, "CRIES: A New Neonatal Postoperative Pain Measurement Score. Initial Testing of Validity and Reliability,"

Pediatr. Anesth., Vol. 5, No. 1, hal. 53–61, 1995.

[5] M. Buchholz, H. W. Karl, M. Pomietto, dan A. Lynn, "Pain Scores in Infants: A Modified Infant Pain Scale Versus Visual Analogue," *J. Pain Symptom Manage.*, Vol. 15, No. 2, hal. 117–124, 1998.

[6] H. Elizeus, "Dynamic Acoustic Pattern as Pain Indicator on Baby Cries Post Surgery Procedure," Disertasi, Universitas Airlangga, Surabaya, Indonesia, 2013.

[7] 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.

[8] L. Nanni, S. Brahnam, dan A. Lumini, "A Local Approach Based on a Local Binary Patterns Variant Texture Descriptor for Classifying Pain States," *Expert Syst. Appl.*, Vol. 37, No. 12, hal. 7888–7894, 2010.

[9] Y. Kristian, M. Hariadi, dan M. H. Purnomo, "Ideal Modified Adachi Chaotic Neural Networks and Active Shape Model for Infant Facial Cry Detection on Still Image," *Proceedings of the International Joint Conference on Neural Networks*, 2014, hal. 2783–2787.

[10] Y. Kristian, H. Takahashi, I.K.E. Purnama, K. Yoshimoto, E.I. Setiawan, E. Hanindito, M.H. Purnomo, "A Novel Approach on Infant Facial Pain Classification using Multi Stage Classifier and Geometrical-Textural Features Combination," *IAENG Int. J. Comput. Sci.*, Vol. 44, No. 1, hal. 112-121, 2017.

[11] Y. Lecun, Y. Bengio, dan G. Hinton, "Deep Learning," *Nature*, Vol. 521, No. 7553, hal. 436–444, 2015.

[12] M. W. Gardner dan S. R. Dorling, "Artificial Neural Networks (The Multilayer Perceptron)—A Review of Applications in the Atmospheric Sciences," *Atmos. Environ.*, Vol. 32, No. 14–15, hal. 2627–2636, 1998.

[13] R. Hecht-Nielsen, "Theory of the Backpropagation Neural Network," *Proc. Int. Jt. Conf. Neural Networks*, 1989, Vol. 1, hal. 593–605.

[14] A. Krizhevsky, "Learning Multiple Layers of Features from Tiny Images," Sci. Dep. Univ. Toronto, Technical Report, hal. 1–60, 2009.

[15] Y. LeCun, L. Bottou, Y. Bengio, dan P. Haffner, "Gradient-Based Learning Applied to Document Recognition," *Proc. of the IEEE*, Vol. 86, No. 11, hal. 2278–2324, 1998.

[16] A. Krizhevsky, I. Sutskever, dan G. E. Hinton, "Imagenet Classification with Deep Convolutional Neural Networks," *Advances in Neural Information Processing Systems*, 2012, hal. 1097–1105.

[17] 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, hal. 167-173, 2018.

[18] W. Ouyang dan X. Wang, "Joint Deep Learning for Pedestrian Detection," *2013 IEEE International Conference on Computer Vision (ICCV)*, 2013, hal. 2056–2063.

[19] G. Levi dan T. Hassner, "Age and Gender Classification Using Convolutional Neural Networks," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2015, hal. 34–42.

[20] K. O'Shea dan R. Nash, "An Introduction to Convolutional Neural Networks," *arXiv Prepr.*, Vol. 1511, hal. 1–11, 2015.

[21] W. Wang, Y. Huang, Y. Wang, dan L. Wang, "Generalized Autoencoder: A Neural Network Framework for Dimensionality Reduction," *IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Work*, 2014, hal. 496–503.

[22] G. E. Hinton dan R. R. Salakhutdinov, "Reducing the Dimensionality of Data with Neural Networks," *Science (80-.).*, Vol. 313, No. 5786, hal. 504–507, 2006.

[23] P. Vincent dan H. Larochelle, "Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion Pierre-Antoine Manzagol," *J. Mach. Learn. Res.*, Vol. 11, hal. 3371–3408, 2010.

[24] S. Wold, K. Esbensen, dan P. Geladi, "Principal Component Analysis," *Chemom. Intell. Lab. Syst.*, Vol. 2, No. 1–3, hal. 37–52, 1987.

[25] J. Masci, U. Meier, D. Cireşan, dan J. Schmidhuber, "Stacked Convolutional Auto-Encoders for Hierarchical Feature Extraction," *International Conference on Artificial Neural Networks*, 2011, hal. 52–59.

[26] H. Sak, A. Senior, dan F. Beaufays, "Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling," *Proc. of the Annual Conference of the International Speech Communication Association (Interspeech 2014)*, 2014, hal. 338–342.

[27] S. Hochreiter, "The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions," *Int. J. Uncertainty, Fuzziness Knowledge-Based Syst.*, Vol. 6, No. 2, hal. 107–116, 1998.

[28] S. Hochreiter dan J. J. Schmidhuber, "Long Short-Term Memory," *Neural Comput.*, Vol. 9, No. 8, hal. 1–32, 1997.

[29] P. Viola dan M. J. Jones, "Robust Real-Time Face Detection," *Int. J. Comput. Vis.*, Vol. 57, No. 2, hal. 137–154, 2004.

[30] 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," *J. Nas. Tek. Elektro dan Teknol. Inf.*, Vol. 6, No. 3, hal. 333-341, 2017.

[31] V. Kazemi dan J. Sullivan, "One Millisecond Face Alignment with an Ensemble of Regression Trees," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, hal. 1867–1874.

[32] B. Xu, N. Wang, T. Chen, dan M. Li, "Empirical Evaluation of Rectified Activations in Convolutional Network," *arXiv Prepr. arXiv1505.00853*, hal. 1-5, 2015.

[33] Z. Wang, A. C. Bovik, H.R. Sheikh, dan E.P. Simoncelli, "Image Quality Assessment: From Error Visibility to Structural Similarity," *IEEE Trans. Image Process.*, Vol. 13, No. 4, hal. 600–612, 2004.

[34] 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.

[35] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, dan R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *J. Mach. Learn. Res.*, Vol. 15, No. 1, hal. 1929–1958, 2014.