IDENTIFIKASI EKSPRESI WAJAH BERBASIS CITRA MENGGUNAKAN ALGORITMA *CONVOLUTIONAL NEURAL NETWORK* (CNN)

SKRIPSI

OLEH:
HAMDANI MUBAROK
NIM. 13650096

JURUSAN TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM
MALANG
2019

IDENTIFIKASI EKSPRESI WAJAH BERBASIS CITRA MENGGUNAKAN ALGORITMA *CONVOLUTIONAL NEURAL NETWORK* (CNN)

SKRIPSI

Diajukan kepada:
Universitas Islam Negeri (UIN) Maulana Malik Ibrahim Malang
Untuk Memenuhi Salah Satu Persyaratan Dalam
Memperoleh Gelar Sarjana Komputer (S.Kom)

Oleh :
HAMDANI MUBAROK
NIM. 13650096

JURUSAN TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM MALANG
2019

LEMBAR PERSETUJUAN

IDENTIFIKASI EKSPRESI WAJAH BERBASIS CITRA MENGGUNAKAN ALGORITMA *CONVOLUTIONAL NEURAL NETWORK* (CNN)

SKRIPSI

OLEH:
HAMDANI MUBAROK
NIM. 13650096

Telah Diperiksa dan Disetujui untuk Diuji
Tanggal : Juni 2019

Dosen Pembimbing I	Dosen Pembimbing II
Irwan Budi Santoso, M.Kom	Dr. Cahyo Crysdian
NIP. 19790103 201101 1 004	NIP. 19740424 200901 1 008

Mengetahui,
Ketua Jurusan Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang

Dr. Cahyo Crysdian
NIP. 19740424 200901 1 008

HALAMAN PENGESAHAN

IDENTIFIKASI EKSPRESI WAJAH BERBASIS CITRA MENGGUNAKAN ALGORITMA *CONVOLUTIONAL NEURAL NETWORK* (CNN)

SKRIPSI

Oleh :
HAMDANI MUBAROK
NIM. 13650096

Telah Dipertahankan di Depan Dewan Penguji Skripsi dan
Dinyatakan Diterima Sebagai Salah Satu Persyaratan untuk
Memperoleh Gelar Sarjana Komputer (S.Kom)
Tanggal : Juni 2019

Susunan Dewan Penguji		**Tanda Tangan**
Penguji Utama	: Dr. M. Amin Hariyadi, MT NIP. 19670118 200501 1 001	( )
Ketua Penguji	: Roro Inda Melani, MT., M.Sc NIP. 19780925 200501 2 008	( )
Sekretaris Penguji	: Irwan Budi Santoso, M.Kom NIP. 19770103 201101 1 004	( )
Anggota Penguji	: Dr. Cahyo Crysdian NIP. 19740424 200901 1 008	( )

Mengetahui dan Mengesahkan,
Ketua Jurusan Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang

Dr. Cahyo Crysdian
NIP. 19740424 200901 1 008

PERNYATAAN KEASLIAN TULISAN

Saya yang bertanda tangan dibawah ini :

Nama : Hamdani Mubarok

NIM : 13650096

Jurusan : Teknik Informatika

Fakultas : Sains dan Teknologi

Judul : **IDENTIFIKASI EKSPRESI WAJAH BERBASIS CITRA MENGGUNAKAN ALGORITMA *CONVOLUTIONAL* NEURAL NETWORK (CNN)**

Menyatakan dengan sebenarnya bahwa skripsi yang saya tulis ini benarbenar merupakan hasil karya saya sendiri, bukan merupakan pengambil alihan data, tulisan atau pikiran orang lain yang saya akui sebagai hasil tulisan atau pikiran saya sendiri, kecuali dengan mencantumkan sumber cuplikan pada daftar pustaka. Apabila di kemudian hari terbukti atau dapat dibuktikan skripsi ini hasil jiplakan, maka saya bersedia menerima sanksi atas perbuatan tersebut.

Malang, Juni 2019

Yang Membuat Pernyataan

Hamdani Mubarok

NIM. 13650096

MOTTO

"Push Yourself To The Limit, Never Quit"

"Dan orang-orang yang bersungguh-sungguh untuk (mencari keridhaan) Kami, benar-benar akan Kami tunjukkan kepada mereka jalan-jalan Kami. Dan sesungguhnya Allah benar-benar beserta orang-orang yang berbuat baik."

(QS. Al-'Ankabut [29]: 69)

HALAMAN PERSEMBAHAN

Alhamdulillah puji syukur kehadirat Allah SWT yang memberikan kekuatan kepada saya hingga bisa sampai menyelesaikan kuliah S1 di kampus UIN MAULANA MALIK IBRAHIM MALANG. Sholawat serta salam kepada Nabi Muhammad SAW, yang membawa petunjuk kepada seluruh umat manusia

Alhamdulillah, terima kasih kepada kedua orang tuas saya, yang telah membesarkan dan mendidik saya dari kecil hingga sekarang bisa menyelesaikan kuliah saya, yang tiap hari mendoakan saya tiada henti, dan selalu mendukung saya dalam keadaan apapun. Tak lupa terimakasih untuk kedua kakak saya yang selalu memberi semangat dan dukungan yang tiada henti sampai pengerjaan skrispi ini selesai.

Terima kasih kepada dosen-dosen yang telah sabar dan ikhlas dalam mendidik saya hingga mampu melewati seluruh ujian dari semua mata kuliah yang saya tempuh, terutama kepada Ibu Roro Inda Melani, MT., M.Sc sebagai dosen wali yang selalu sabar dalam membuka wawasan saya. Tidak lupa juga saya ucapkan terimakasih kepada Bapak Irwan Budi Santoso, M.Kom dan Bapak Cahyo Crysdian selaku dosen pembimbing I dan pembimbing II, yang selalu sabar membimbing saya dalam pengerjaan skrispi ini, semoga ilmu yang beliau amalkan berguna bagi seluruh mahasiswa.

Terima kasih kepada seluruh teman seperjuangan yang menemani dan banyak menbantu selama kuliah. Teknik Informatika angkatan 2013, khususnya kepada Dimas Ari Setiawan, Dhofir Alibi, Ahmad Dzulfikri yang banyak memberi masukan dalam pengerjaan skripsi. Semoga kita semua mampu mewujudkan segala cita-cita yang kita impikan. Amin Allahumma Amin.

KATA PENGANTAR

Assalamu'alaikum Wr.Wb.

Segala puji bagi Allah SWT tuhan semesta alam, karena atas segala rahmat dan karunia-Nya sehingga penulis mampu menyelesaikan skripsi dengan baik dan lancar. Shalawat serta salam selalu tercurah kepada tauladan terbaik Nabi Muhammad SAW yang telah membimbing umatnya dari zaman kebodohan menuju islam yang *rahmatan lil alamiin.*

Dalam menyelesaikan skripsi ini, banyak pihak yang telah memberikan bantuan baik secara moril, nasihat dan semangat maupun materiil. Atas segala bantuan yang telah diberikan, penulis ingin menyampaikan doa dan ucapan terimakasih yang sedalam-dalamnya kepada:

1. Prof. Dr. Abdul Haris, M.Ag, selaku Rektor Universitas Islam Negeri Maulana Malik Ibrahim Malang.
2. Dr. Sri Harini, M.Si selaku Dekan Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang.
3. Dr. Cahyo Crysdian, selaku Ketua Jurusan Teknik Informatika Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang.
4. Irwan Budi Santoso, M.Kom., selaku dosen pembimbing I yang telah meluangkan waktu untuk membimbing, memotivasi, dan mengarahkan dan memberi masukan kepada penulis dalam pengerjaan skripsi ini hingga akhir.

5. Dr. Cahyo Crysdian, selaku dosen pembimbing II yang senantiasa memberi masukan dan nasihat serta petunjuk dalam penyusunan skripsi ini.
6. Segenap dosen Teknik Informatika yang telah memberikan bimbingan keilmuan selama masa studi.
7. Teman-teman seperjuangan Teknik Informatika angkatan 2013.

Berbagai kekurangan dan kesalahan mungkin pembaca temukan dalam penulisan skripsi ini, untuk itu penulis menerima segala kritik dan saran yang membangun dari pembaca sekalian. Semoga apa yang menjadi kekurangan bisa disempurnakan oleh peneliti selanjutnya dan semoga karya ini senantiasa dapat memberi manfaat. Amin.

Wassalamualaikum Wr.Wb

Malang, Mei 2019

Penulis

DAFTAR ISI

HALAMAN JUDUL ..... **i**
LEMBAR PERSETUJUAN ..... **ii**
HALAMAN PENGESAHAN ..... **iii**
PERNYATAAN KEASLIAN TULISAN ..... **iv**
MOTTO ..... **v**
HALAMAN PERSEMBAHAN ..... **vi**
KATA PENGANTAR ..... **vii**
DAFTAR ISI ..... **ix**
DAFTAR GAMBAR ..... **xi**
DAFTAR TABEL ..... **xiii**
ABSTRAK ..... **xiv**
ABSTRACT ..... **xv**
ملخص ..... **xvi**
BAB I PENDAHULUAN **1**
1.1. Latar Belakang 1
1.2. Masalah Penelitian 6
1.3. Tujuan Penelitian 7
1.4. Manfaat Penelitian 7
1.5. Batasan Penelitian 7
BAB II TINJAUAN PUSTAKA **8**
2.1. *Face Recognition* 8
2.1.1. Metode *Recognition* 9
2.2. *Machine Learning* 10
2.2.1. *Artificial Neural Network* 13
2.2.2. Karakteristik *Neural Network* 15
2.2.3. Arsitektur *Neural Network* 18
2.2.4. *Backpropagation* 21
2.3. *Deep learning* 24
2.4. *Convolutional Neural Network* 25
2.4.1. *Convolution layer* 27

2.4.2.*Pooling Layer* 32
2.4.3.*Fully connected layer* 33
2.5. *Related Research* 34
BAB III METODE PENELITIAN 38
3.1. Alur Penelitian 38
3.2. Pengumpulan Data 40
3.3. Desain Sistem 43
3.3.1.*Input* Gambar 45
3.3.2.*Preprocessing* Citra 45
3.3.3.Implementasi CNN 49
3.3.3.1. Proses *Training* 50
3.3.3.2. Proses *Testing* 57
BAB IV HASIL DAN PEMBAHASAN 60
4.1. Langkah-langkah Uji Coba 60
4.2. Hasil Uji Coba 60
4.2.2.Hasil Proses *Training* 60
4.2.3.Hasil Proses *Testing* 62
4.2.4.Pengaruh Parameter *Learning* Terhadap Akurasi 66
4.2.4.1. Pengaruh Jumlah *Epoch* 66
4.2.4.2. Pengaruh Jumlah Nilai *Learning Rate* 67
4.2.4.3. Pengaruh Ukuran Citra 67
4.3. Pembahasan 68
BAB V PENUTUP 73
5.1. Kesimpulan 73
5.2. Saran 73
DAFTAR PUSTAKA 75

DAFTAR GAMBAR

Gambar 2. 1 Machine learning 9
Gambar 2. 2 Deep learning 10
Gambar 2. 3 Algoritma machine learning 13
Gambar 2. 4 Neuron pada otak 14
Gambar 2. 5 Karakteristik neural network. 15
Gambar 2. 6 Grafik fungsi aktivasi sigmoid & tanh 17
Gambar 2. 7 Grafik fungsi aktivasi ReLu 18
Gambar 2. 8 Single Layer Neural Net. 19
Gambar 2. 9 Multilayer Neural Net 20
Gambar 2. 10 Backpropagation 21
Gambar 2. 11 Backpro tahap pertama 23
Gambar 2. 12 Arsitektur CNN 27
Gambar 2. 13 Urutan lapisan cnn 27
Gambar 2. 14 Konvolusi untuk membentuk kedalaman 28
Gambar 2. 15 Stride konvolusi 29
Gambar 2. 16 Penambahan padding 30
Gambar 2. 17 Proses konvolusi 1&2 32
Gambar 2. 18 Max pooling 33

Gambar 3. 1 Alur penelitian 38
Gambar 3. 2 Ekspresi netral 41
Gambar 3. 3 Ekspresi senyum 41
Gambar 3. 4 Ekspresi marah 42
Gambar 3. 5 Proses pengambilan foto 42
Gambar 3. 6 Source code pemisahan dataset 43
Gambar 3. 7 Dataset sesuai kelas 43
Gambar 3. 8 Desain sistem 44
Gambar 3. 9 Cropping image 46
Gambar 3. 10 Source code grayscale image 47
Gambar 3. 11 Citra grayscale 47
Gambar 3. 12 Augmentasi gambar 48

Gambar 3. 13 Source code augmentasi citra 48
Gambar 3. 14 Hasil augmentasi citra 49
Gambar 3. 15 Skema training dan testing 50
Gambar 3. 16 Arsitektur CNN 51
Gambar 3. 17 Simulasi CNN model 52
Gambar 3. 18 Source code model CNN 55
Gambar 3. 19 Source code compile model 56
Gambar 3. 20 Source code model fitting 57
Gambar 3. 21 Source code pemanggilan model 57
Gambar 3. 22 Output klasifikasi 58
Gambar 3. 23 Source code membandingkan bobot 59
Gambar 4. 1 Grafik akurasi dan loss 62
Gambar 4. 2 Source code perhitungan akurasi 62
Gambar 4. 3 Hasil feature learning tahap 1 69

DAFTAR TABEL

Tabel 3. 1 Inisialisasi parameter..... 51
Tabel 4. 1 Hasil prediksi proses training..... 61
Tabel 4. 2 Hasil uji coba identifikasi ekspresi wajah..... 63
Tabel 4. 3 Hasil prediksi proses testing 65
Tabel 4. 4 Pengaruh epoch..... 66
Tabel 4. 5 Pengaruh nilai learning rate 67
Tabel 4. 6 Pengaruh ukuran citra 68

ABSTRAK

Mubarok, Hamdani. 2019. **Identifikasi Ekspresi Wajah Berbasis Citra Menggunakan Algoritma *Convolution*al Neural Network (CNN)**. Skripsi. Jurusan Teknik Informatika Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang.
Pembimbing (I) Irwan Budi Santoso, M. Kom. (II) Dr. Cahyo Crysdian

Kata Kunci : *convolution, neural nets, preprocessing, grayscaled image, data augmentation, training, testing.*

Identifikasi ekspresi wajah merupakan materi penelitian yang saat ini terus dikembangkan. Detailnya ciri yang mesti ditangkap membuat para peneliti berlomba-lomba menemukan metode yang paling cocok untuk melakukan identifikasi. Algoritma *Convolutional Neural Network* (CNN) menjadi salah satu algoritma yang saat ini paling diunggulkan dalam bidang klasifikasi dan identifikasi objek saat ini. Dengan menggabungkan 2 metode dalam satu rangkaian yaitu *convolution* untuk ekstraksi ciri dan *neural nets* untuk klasifikasinya membuat algoritma ini lebih mudah digunakan. Pada penelitian ini, ekspresi yang akan diidentifikasi merupakan foto ekspresi yang diambil langsung dari mahasiswa UIN Malang. Data sebanyak 687 foto dibagi menjadi 2 bagian yaitu sebagai data *training* dan data *testing* dengan porsi 90:10. Kemudian untuk mempermudah dalam ekstraksi ciri dari fitur yang akan diidentifikasi, peneliti melakukan 3 tahap *preprocessing* yaitu *cropping, grayscaled image* dan *data augmentation*. Tujuan penelitian ini adalah untuk mengetahui akurasi algoritma CNN dalam mengidentifikasi ekspresi wajah berbasis citra. Setelah algoritma dijalankan dan model sudah terbentuk, maka didapatkan akurasi *training* sebesar 99,6% dan akurasi *testing* sebesar 88.89%.

ABSTRACT

Mubarok, Hamdani. 2019. **Identification of Image-Based Facial Expressions Using Convolutional Neural Network (CNN) Algorithm**. Undergraduate Thesis. Informatics Enginering Departement. Faculty of Science and Technology. State Islamic University of Maulana Malik Ibrahim Malang.
Advisor : (I) Irwan Budi Santoso, M. Kom. (II) Dr. Cahyo Crysdian

Keywords : convolution, neural nets, preprocessing, grayscaled image, data augmentation, training, testing.

The identification of facial expressions is a research material that is currently being developed. The details of the traits that must be captured make the researchers vying to find the most suitable method for identification. The Convolutional Neural Network (CNN) algorithm is one of the most favored algorithms in the field of classification and identification of current objects. By combining 2 methods in one series, convolution for feature extraction and neural nets for classification makes this algorithm easier to use. In this study, the expression to be identified is a photo of expression taken directly from UIN Malang students. Data of 687 photos are divided into 2 parts, namely as training data and testing data with a portion of 90:10. Then to facilitate the extraction of features from the features to be identified, the researcher conducted 3 preprocessing stages, namely cropping, grayscaled image and data augmentation. The purpose of this study was to determine the accuracy of the CNN algorithm in identifying image-based facial expressions. After the algorithm is run and the model has been formed, the training accuracy is 99.6% and the testing accuracy is 88.89%.

ملخص

مبروك, حمداني. 2019. تحديد تعبيرات الوجه المستندة إلى الصور باستخدام الخوارزميات **Convolutional Neural Network (CNN)**. البحث العلوم والتكنولوجية بجامعة موالن مالك إبراهيم اإلسالمية الحكومية مالنج.
حت اإلشراف : اروان بودي سانتوسو, الدكتور كاهي كريسديان

كلمت البحث : زيادة البيانات, صورة رمادية, تجهيزها, شبكات العصبية, التفاف, اختبارات, تدريب

تحديد تعبيرات الوجه هي مادة بحثية يجري تطويرها حاليًا. إن تفاصيل السمات التي يجب التقاطها تجعل الباحثين يتنافسون لإيجاد أنسب طريقة لتحديد الهوية. خوارزمية *Convolutional Neural Network* (CNN) تصبح واحدة من أكثر الخوارزميات المفضلة في مجال تصنيف وتحديد الأشياء الحالية. من خلال الجمع بين 2 طرق في سلسلة هذا هو *convolution* لاستخراج الميزة و *neural nets* للتصنيف يجعل هذه الخوارزمية أسهل في الاستخدام. في هذه الدراسة ، التعبير المراد تحديده هو صورة للتعبير مأخوذة مباشرة من الطالب. تنقسم بيانات 687 صورة إلى قسمين وهي بيانات التدريب وبيانات الاختبار مع جزء من 90:10. ثم لتبسيط استخراج الميزة من الميزات التي سيتم تحديدها ، أجرى الباحث 3 مراحل *preprocessing* هذا هو *cropping, grayscaled image* و *data augmentation*. الغرض من هذه الدراسة هو لمعرفة دقة الخوارزمية CNN في تحديد تعبيرات الوجه المستندة إلى الصور. بعد تشغيل الخوارزمية وتم تشكيل النموذج ، ثم دقة التدريب كبيرة مثل 99.6% ودقة الاختبار كبيرة مثل 88.89%.

BAB I

PENDAHULUAN

1.1. Latar Belakang

Saat ini ilmu teknologi pengolahan citra digital (*Digital Image Processing*) sudah sangat berkembang pesat semenjak adanya teknologi komputer yang sanggup mengolah algoritma pada tahun 1960. Banyak sekali masalah-masalah yang bisa diselesaikan oleh salah satu fokus keilmuan di bidang informatika ini. Bukan hanya itu saja, teknologi pengolahan citra digital saat juga mulai berkembang dan diterapkan pada berbagai bidang ilmu lainnya, mulai dari bidang hiburan hingga kesehatan.

Berkembangnya teknologi pengolahan citra ke berbagai bidang ini membuat semakin banyaknya masalah-masalah yang bisa diselesaikan. Bukan hanya dapat menyelesaikan masalah, tapi teknologi pengolahan citra ini juga membuka inovasi-inovasi baru dalam suatu sistem. Secara tidak langsung teknologi pengolahan citra ini mulai menggeser sistem-sistem yang masih menggunakan teknologi lama.

Jika dilihat dari perkembangannya, teknologi pengolahan citra ini semakin berkembang bukan hanya karena kebutuhan manusia akan teknologi yang lebih sederhana saja tapi juga kebutuhan sistem juga. Contoh kecil yang bisa dilihat di sekitar kita adalah sistem absensi. Dahulu sistem absensi yang masih manual, mulai berkembang dengan memanfaatkan pengolahan citra yaitu dengan adanya finger print. Dari sisi teknologi, finger print sangatlah membantu karena cukup mudah digunakan dan tidak perlu operator untuk mengisi absensi. Sedangkan dari kebutuhan sistem, untuk mengenali sidik jari orang yang absen dan mencocokkan

dengan data yang sebelumnya sudah ada juga memerlukan metode yang bagus. Hal ini sangatlah penting karena jika terjadi kesalahan baik dalam mengenali atapun mencocokkan data maka akan terjadi masalah dan kerugian oleh beberapa pihak. Oleh karena itu, saat ini bukan hanya teknologi dari pengolahan citra saja yang berkembang tetapi metode-metode *recognition* yang melatar belakangi teknologi pengolahan citra juga berkembang.

Seperti yang sudah disampaikan sebelumnya, metode *recognition* ini sangatlah penting perannya ketika diaplikasikan sesuai dengan kebutuhannya. Karena pasti akan banyak sekali masalah yang akan terselesaikan ketika metode *recognition* ini berhasil diaplikasikan apalagi jika memilki performa yang bagus. Tentunya hal ini juga harus didukung oleh perangkat komputasi yang bagus agar metode *recognition* bekerja dengan baik. Salah satu pengaplikasian metode *recognition* yang akan bisa sangat membantu manusia yaitu jika diaplikasikan pada bidang robotika.

Saat ini banyak sekali peneliti yang berlomba-lomba untuk membuat robot yang mampu menggantikan manusia. Dan kemampuan *recognition* di bidang robotika ini menjadi salah satu kemampuan penting yang harus dimiliki ketika mengembangkan robot. Oleh karena itu, penelitian tentang kemampuan penglihatan terutama untuk mengenali objek menjadi salah satu penelitian utama pada komunitas robotika. Selain itu memungkinkan untuk dikembangkan dengan kecerdasan yang tinggi, sehingga mampu menyelesaikan pekerjaan sulit yang membutuhkan kemampuan penglihatan manusia (Sheng & Enrico, 2008). Salah satu kemampuan yang saat ini dikembangkan adalah kemampuan robot untuk

mengenali wajah seseorang (*face recognition*). Hal ini tentunya akan sangat membantu berbagai pekerjaan jika robot sudah memiliki kemampuan tersebut.

Selain pada bidang robotika, kemampuan *face recognition* ini juga sangatlah dibutuhkan pada bidang keamanan. Seperti yang sudah diketahui banyak orang, saat ini sistem keamanan sudah banyak terbantu dengan adanya kamera CCTV. Tetapi kemampuan CCTV yang terpasang saat ini masihlah sangat terbatas hanya untuk merekam sudut-sudut yang tertangkap oleh kamera tersebut. Kini timbul pertanyaan, bagaimana caranya membuat kamera CCTV bukan hanya merekam saja, tapi juga bisa untuk mengawasi, mengenali objek, mengenali persepsi ancaman dan pencegahan pencurian (Anand, 2017).

Tentunya itu bukanlah hal yang mudah, membutuhkan kemampuan *machine learning* yang sangat bagus untuk mewujudkannya. Karena jika kebutuhan sistem sampai pada taraf untuk mengenali persepsi ancaman, maka dibutuhkan sebuah metode pengenalan yang tidak hanya mampu mengenali wajah saja, tapi juga bisa mengenali ekspresi wajah dari seseorang. Hal ini tentunya akan menjadi tantangan tersendiri, karena untuk mengenali wajah hingga taraf ekspresi tentu membutuhkan sebuah algoritma yang mampu mengenali ciri suatu citra secara detail.

Bukan hanya itu saja, jika berpikir lebih luas lagi, kita tentu tahu jika ekspresi dari setiap orang berbeda-beda. Oleh karena itu, dibutuhkanlah *dataset* besar yang berisi ekspresi yang berbeda-beda pada setiap kategorinya. Supaya algoritma yang digunakan nanti memiliki variasi dan perbandingan data yang cukup banyak sehingga algoritma tidak kesulitan ketika menemukan ekspresi-ekspresi baru yang belum ada pada *database*nya. Disinilah tantangan yang harus dipecahkan oleh para peneliti untuk menemukan sebuah algoritma yang bukan hanya mampu mengenali

ciri pada wajah secara detail tapi juga mampu untuk menampung dan mempelajari banyak data sekaligus.

Demi menjangkau itu semua dibutuhkan sebuah *machine learning* yang mampu mempelajari pekerjaan itu secara lebih dalam. Oleh karena itu, saat ini dikembangkan metode pembelajaran terbaru yang mampu memepelajari suatu pekerjaan secara lebih dalam yaitu *deep learning*. *Deep learning* adalah bagian dari *machine learning* yang memungkinkan komputer mempelajari berdasarkan pengalaman masala lalu dan memahami perintah berdasarkan konsep yang diberikan (Goodfellow *et al*., 2016). Hal yang membedakan *deep learning* dengan *machine learning* adalah pada struktur pembelajarannya yang lebih dalam berdasarkan representasi data yang ingin dipelajari. *Deep learning* saat ini menjadi topik hangat yang sering dibicarakan terkait penelitian tentang *object recognition*, *object detection*, *speech recognition* dan berbagai macam pattern *recognition* lainnya (Javier Ruiz *et al*., 2018).

Deep learning ini telah mengubah paradigma penelitian pattern *recognition* yang sebelumnya masih memisahkan anatar *feature extraction* bersama dengan metode klasifikasi secara terpisah. Dengan adanya *deep learning* ini menggunakan keduanya dalam satu struktur bisa dilakukan bersama (Albani *et al*., 2017). Salah satu algoritma yang menerapkan metode *deep learning* adalah *deep neural network* atau lebih dikenal dengan *Convolutional Neural Network* (CNN).

Ide dasar algoritma *Convolutional Neural Network* (CNN) ini adalah meniru struktur algoritma *neural network* yang melakukan proses pembelajaran melalui beberapa lapisan. Selain itu jika dilihat dari latar belakang adanya algoritma *neural network* ini adalah keinginan untuk meniru kemampuan berfikir manusia. Oleh

karena itu kemampuan untuk mempelajari suatu data secara lebih dalam sangat cocok diterapkan pada algoritma *neural network* ini. Meskipun begitu, secara struktural algoritma CNN ini cukup berbeda dari algoritma *neural network*. Lebih tepatnya *neural network* digunakan pada sebagian kecil struktur algoritma CNN ini, yaitu pada bagian klasifikasinya. Sementara untuk proses *feature learning*, sesuai dengan namanya, algoritma ini menggunakan teori konvolusi untuk proses *feature learning*.

Dalam beberapa penelitian *deep learning* telah menunjukan performa yang luar biasa. Hal ini sebagain besar dipengaruhi oleh kemampuannya untuk memeplajari *dataset* yang besar dan teknik melatih jaringan yang lebih dalam. Selain itu dengan dukungan perangkat komputasi yang lebih kuat membuat *deep learning* kini menjadi banyak digunakan (Goodfellow *et al*., 2016). Kemampuan CNN di klaim sebagai model terbaik untuk memecahkan permasalahan *object recognition* dan *face recognition*. Namun dalam CNN, seperti model *deep learning* lainnya, memiliki kelemahan yaitu proses komputasi model yang cukup lama. Tetapi dengan perkembangan hardware yang semakin pesat, hal tersebut dapat diatasi menggunakan teknologi *Graphical Procesing Unit* (GPU) dan PC yang memiliki spesifikasi tinggi.

Berdasarkan latar belakang diatas, penelitian ini akan mencoba untuk mengimplementasikan algoritma CNN ini. Pada penelitian ini berfokus terhadap bagaimana cara mengimplementasikan algoritma CNN untuk mengidentifikasi jenis ekspresi pada wajah laki-laki. Karena ini merupakan penelitian untuk mengidentifikasi jenis ekspresi, maka dibutuhkan objek yang akan digunakan sebagai bahan penelitian. Oleh karena itu objek yang akan dijadikan bahan

penelitian kali ini adalah foto ekspresi wajah mahasiswa laki-laki UIN Malang. Jadi ekspresi wajah mahasiswa UIN Malang nantinya akan foto kemudian diolah untuk dijadikan sebagai objek untuk penelitian ini.

Penelitian ini akan sangat bermanfaat nantinya ketika dikembangkan lebih lanjut khususnya jika dikembangkan dalam bidang keamanan. Sebagai contoh jika mampu diterapkan pada perangkat CCTV sebagai kamera pengawas di tempat parkir. Hal ini selaras dengan sabda rasulullah ﷺ yang menganjurkan umatnya untuk menjadi seseorang yang bermanfaat untuk lainnya. Rasulullah ﷺ bersabda:

خَيْرُ النَاسِ أَنْفَعُهُمْ لِلنَاسِ

"Sebaik-baik manusia adalah yang paling bermanfaat bagi manusia" (HR. Ahmad, ath-Thabrani, ad-Daruqutni. Hadits ini dihasankan oleh al-Albani di dalam Shahihul Jami' no:3289).

Menjadi seseorang yang bermanfaat untuk orang lain adalah salah satu karakter yang harus dimiliki sorang muslim. Karena pada dasarnya, ketika kita memberikan atau melakukan sesuatu yang bermanfaat untuk orang lain, maka manfaatnya akan kembali untuk kebaikan diri kita sendiri. Hal ini dijelaskan dalam firman Allah ﷻ pada surah al-isra ayat 7 :

إِنْ أَحْسَنْتُمْ أَحْسَنْتُمْ لِأَنْفُسِكُمْ

"Jika kalian berbuat baik, sesungguhnya kalian berbuat baik bagi diri kalian sendiri" (QS. Al-Isra:7).

1.2. Masalah Penelitian

Berdasarkan latar belakang masalah yang telah dipaparkan sebelumnya, maka pertanyaan yang akan diangkat pada penelitian ini adalah seberapa besar akurasi algoritma CNN dalam mengidentifikasi ekspresi wajah berbasis citra?

1.3. Tujuan Penelitian

Secara umum tujuan utama diadakannya penelitian ini adalah mengenali suatu objek. Adapun maksud dan tujuan yang di dapat dari penelitian ini adalah untuk mengetahui seberapa besar akurasi algoritma CNN mengidentifikasi ekspresi wajah berbasis citra.

1.4. Manfaat Penelitian

Manfaat yang diharapkan dari sistem identifikasi ekspresi wajah berbasis citra menggunakan metode *Convolutional Neural Network* (CNN) ini adalah :

1. Penelitian ini bisa dikembangkan untuk mendeteksi ekspresi secara *real time* pada CCTV.
2. Penelitian ini bisa dikembangkan lagi pada bidang robotika untuk menentukan tindakan berdasarkan ekspresi manusia.

1.5. Batasan Penelitian

Agar penelitian ini terarah dan permasalahan yang dihadapi tidak terlalu luas serta sesuai dengan tujuan penulis, maka ditetapkan batasan terhadap masalah yang sedang diteliti. Adapun batasan masalahnya, yaitu:

1. Data yang digunakan pada penelitian ini adalah data berupa foto ekpresi dari mahasiswa dalam negeri yang kuliah di UIN Malang
2. Ekspesi yang diteliti ada 3 yaitu senyum, netral dan marah.

BAB II

TINJAUAN PUSTAKA

2.1. *Face Recognition*

Face recognition adalah metode biomedika yang digunakan untuk mengidentifikasi atau memverifikasi seseorang dengan membandingkan dan menganalisa pola pada kontur wajah seseorang. Metode biomedika seperti ini biasanya menggunakan sifat-sifat biologi untuk mengidentikasi seseorang. Secara alami, mata manusia mampu untuk mengenali seseorang secara sempurna dengan melihatnya. Tetapi ketika dihadapkan pada pekerjaan yang membutuhkan pengawasan jangka panjang atau untuk menemukan seseorang pada ribuan foto, mata manusia akan kesulitan melakukan itu. Oleh karena itu, metode *face recognition* ini dikembangkan untuk melakukan pekerjaan penting pada pengawasan, keamanan dan forensik.

Saat ini, teknologi *face recognition*, sudah digunakan di berbagai bidang seperti kepolosian hingga pemerintahan. Dibidang kepolisian, *face recognition* digunakan untuk identifikasi forensik, sedangkan diperusahaan metode ini digunakan untuk akses keamanan pada area tertentu. Bahkan baru-baru ini, pemerintahan *United States* menggunakan *face recognition* untuk mencari dan mengidentifikasi anak yang hilang pada internet.

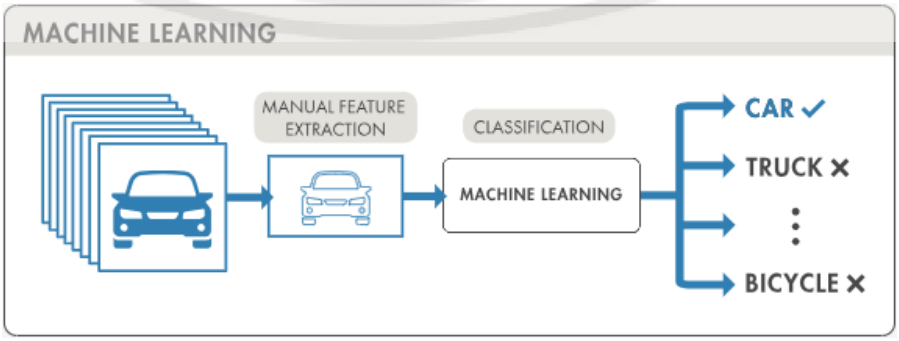
Cara kerja metode *face recognition* ini yaitu dengan mengambil gambar wajah seseorang, mengekstraksi gambar tersebut, kemudian membandingkannya dengan gambar yang ada pada *database* dan mengidentifikasi apakaha cocok atau tidak. Karena metode ini menggunakan komputasi dalam pemrosesannya, maka

tidak bisa disamakan dengan mata manusia. Oleh karena itu, hasilnya sangat bergantung pada kualitas gambar dan algoritma *recognition* yang digunakan.

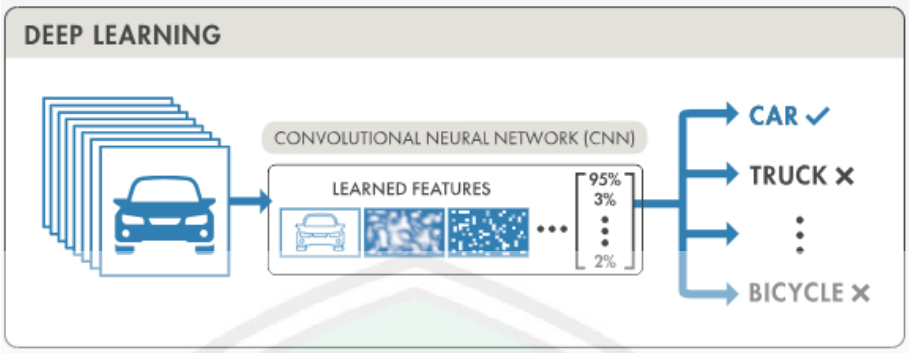
Saat ini banyak peneliti yang berlomba-lomba untuk menemukan algoritma *recognition* yang mampu memberikan hasil terbaik. Salah satu tantangan utama dari *face recognition* ini adalah kemampuan dan akurasi algoritma dalam mengidentifikasi dan memverifikasi seseorang dengan ribuan *database* yang memiliki banyak sudut pandang kamera, pencahayaan, *non-rigrid deformation* dan kondisi pencitraan. Masalah tentang akurasi dalam *face recognition* itulah yang saat ini masih terus dicari solusinya, demi mendapatkan akurasi terbaik yang mendekati kemampuan visual manusia (encyclopedia.com, 2019).

2.1.1. Metode *Recognition*

Ada beberapa metode yang bisa digunakan dalam *face recognition*. Ada berbagai pendekatan yang bisa dilakukan untuk mengenali wajah seseorang. Baru-baru ini, beberapa metode yang termasuk teknik pada *machine learning* dan *deep learning* telah menjadi pendekatan yang paling populer digunakan untuk masalah pengenalan objek ini. Metode pada kedua teknik tersebut sama-sama teknik untuk mempelajari dan mengidentifikasi objek pada gambar, tapi mereka berbeda dalam pelaksanaannya.



Gambar 2. 1 *Machine learning* (mathworks.com, 2016)



Gambar 2. 2 *Deep learning* (mathworks.com, 2016)

Gambar diatas merupakan gambaran perbedaan dari teknik *machine learning* dan *deep learning*. Pada teknik *machine learning*, membutuhkan pendefinisian gambar terlebih dahulu agar dapat dibaca oleh komputer. Sedangkan pada teknik *deep learning*, metode yang dijalankan lebih sederhana tapi proses pembelajaran lebih panjang, tapi mampu menghasilkan akurasi yang lebih tinggi.

2.2. *Machine Learning*

Machine learning adalah teknik untuk menganalisis data yang mengajarkan komputer untuk melakukan kemampuan alami manusia dan hewan (pembelajaran dari pengalaman). Algoritma *machine learning* menggunakan metode komputasi untuk "mempelajari" informasi dari data tanpa bergantung pada persamaan yang telah ditentukan sebagai model. Algoritma akan beradaptasi untuk meningkatkan kinerja mereka ketika jumlah sampel atau data *training* yang tersedia untuk pembelajaran meningkat.

Dengan populernya istilah big data, teknik *machine learning* ini merupakan teknik kunci untuk mengatasi berbagai permasalahan tersebut. Seperti:

- Sitem keuangan, untuk *credit scoring*, *trading* dan berbagai masalah akuntansi.

- *Image processing* dan *computer vision*, untuk pengenalan wajah, deteksi gerakan dan pengenalan objek.
- Biomedika, untuk mendeteksi tumor, penyesuaian DNA, pengenalan sidik jari dan untuk berbagai bidang lainnya.

Dalam proses pembelajaran, *machine learning* ini umumnya menggunakan 2 skenario pembelajaran yaitu *supervised learning* dan *unsupervised learning*. Perbedaan keduanya ada pada bagaimana cara mereka belajar (matworks.com, 2016).

- *Supervised Learning*

  Pada skenario supervised *learning*, algoritma yang digunakan seolah-olah akan dilatih terlebih dahulu untuk melakukan prediksi maupun klasifikasi menggunakan data latih hingga mencapai target yang ditetapkan oleh *user*. Biasanya algoritma akan dilatih dengan beberapa macam data latih berbeda untuk mencapai beberapa target tertentu sesuai dengan kualifikasi yang diberikan *user*. Ketika algoritma sudah mencapai atau sangat mendekati target, maka algoritma sudah siap untuk di uji.

  Sebagai contoh untuk klasifikasi buah mangga matang dan mangga busuk berbasis gambar. Sebelum malakukan klasifikasi, perlu disiapkan data latih berupa beberapa foto mangga matang dan mangga busuk. Kedua data latih tersebut kemudian dilatih menggunakan algoritma *neural network* dengan 2 target yang berbeda. Setelah data latih yang di tentukan mencapai target, maka algoritma tersebut siap digunakan untuk menguji gambar.

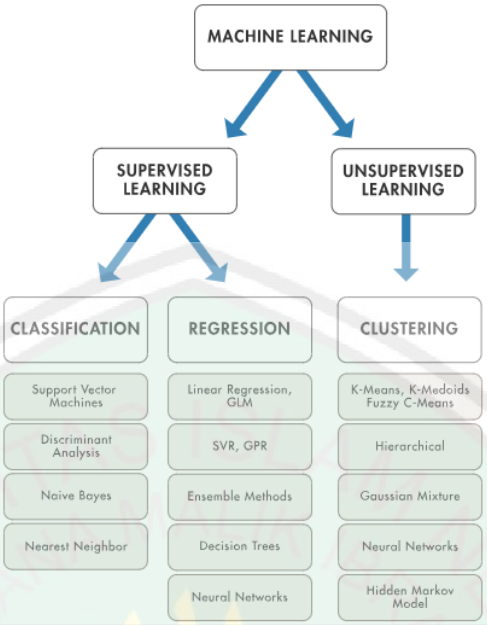
  Supervised *learning* menggunakan 2 tehnik untuk membangun sebuah model, yaitu model klasifikasi dan model regresi. Ada beberapa algoritma

yang termasuk dalam supervised *learning* ini, seperti ; *neural network, naïve bayes classifier, decision tree*, regresi linier berganda, analisis deret waktu dan beberapa algoritma lainnya.

- *Unsupervised Learning*

  Berbeda dengan teknik supervised, teknik *unsupervised* ini tidak membutuhkan data latih unruk melatih algoritma. Pendekatan yang digunakan oleh teknik *unsupervised learning* ini adalah dengan melakukan pembelajaran pada pola tersembunyi atau struktur yang terkandung pada data tersebut. Setelah pola pada data tersebut terbentuk, algoritma akan mengelompokkan data sesuai dengan pola data lain yang sejenis. Berdasarkan model matematikanya, teknik *unsupervised* ini tidak memiliki target variabel.

  Pemodelan yang paling sering digunakan oleh teknik *unsupervised* ini adalah *clustering model*. Model ini digunakan untuk menganalisis suatu data untuk menemukan pola tersembunyi atau mengelompokkan suatu data. Model clustering ini biasanya diaplikasikan untuk market research, tes psikologi dan lainnya. Algoritma yang biasa digunakan untuk *unsupervised learning* ini seperti ; *fuzzy c-means, k-means, hierarchial clustering* dsb.



Gambar 2. 3 Algoritma *machine learning* (mathworks.com, 2016)

Dalam kasus *object recognition*, saat ini proses *learning* sebenarnya bisa melakukan keduanya, tergantung ekstraksi seperti apa yang dijalankan. Tapi dalam beberapa penelitian sebelumnya, teknik *unsupervised* dengan metode klasifikasi memberikan hasil dengan akurasi tinggi. Salah satu metode klasifikasi *unsupervised learning* yang cukup sering digunakan untuk *object recognition* adalah model klasifikasi menggunakan algoritma *neural network* (Jaringan Syaraf Tiruan). Algoritma *neural network* ini sering digunakan untuk pengenalan pola seperti kasus *object recognition* ini karena kemampuannya yang unik, yaitu kemampuan untuk meniru proses pembelajaran seperti otak manusia (Adrian Rosebrock, 2017).

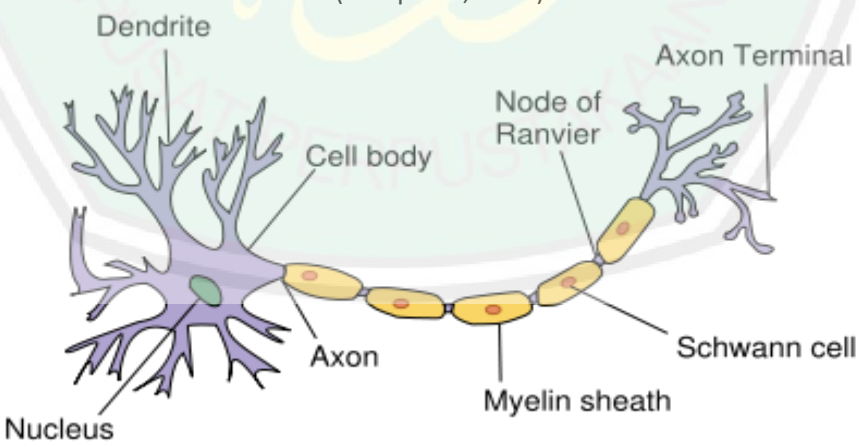
2.2.1. *Artificial Neural Network*

Algoritma *Artificial neural network* (ANN) muncul pertama kali ketika manusia ingin menggambungkan antara kemampuan otak manusia yang mampu untuk belajar dan komputer yang memiliki kemampuan untuk memproses banyak data dan menyimpannya. Karena banyak sekali masalah yang tidak mampu

dirumuskan oleh algoritma biasa. Contohnya untuk menentukan harga sebuah perumahan *atau real estate*. Otak manusia mampu untuk memperkirakan kalkulasi harga yang sesuai, tapi algoritma komputer tidak mampu melakukan hal yang sama seperti itu. Sementara, saat ini segala hal sudah di komputerisasi, untuk meringankan beban manusia. Untuk itulah, algoritma *Artificial Neural Network* (ANN) kini banyak digunakan diberbagai bidang, baik itu untuk klasifikasi, pengenalan pola ataupun untuk prediksi.

Artificial Neural Network (ANN) merupakan suatu model komputasi paralel yang meniru fungsi dari sistem jaringan syaraf biologi otak manusia. Dalam otak manusia terdiri dari milyaran *neuron* yang saling berhubungan. Hubungan ini disebut dengan *Synapses*. Komponen *neuron* terdiri dari satu inti sel yang akan melakukan pemrosesan informasi, satu akson (*axon*) dan minimal satu *dendrit*. Informasi yang masuk akan diterima oleh *dendrit*. Selain itu, dendrit juga menyertasi *akson* sebagai keluaran dari suatu pemrosesan informasi.

(Wikipedia, 2019)



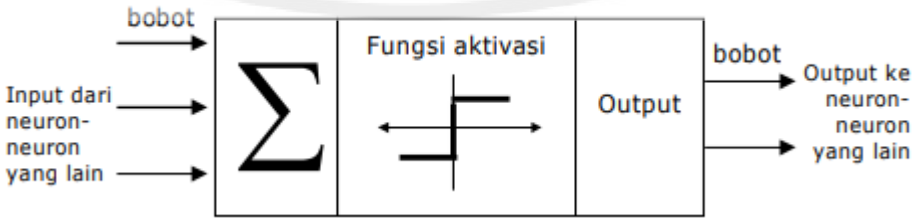
Gambar 2. 4 *Neuron* pada otak (wikipedia.com, 2019)

Struktur pada gambar diatas adalah bentuk standard satuan unik jaringan otak manusia yang telah disederhanakan (satu *neuron*). Bentuk standard ini mungkin akan berubah dikemudian hari jika ada ilmuwan lain yang menemukan

bentuk standard yang lebih baik ataupun memperbaiki bentuk standard ini. Cara kerja dari satuan sistem syaraf diatas adalah bermula pada sinyal masuk melalui dendrit yang kemudian diproses pada *cell body* dengan fungsi tertentu. Informasi tersebut kemudian akan dikirimkan ke *neuron* lain melewati sinapsis (pertemuan antar dua sel). jika sinyal hasil proses memenuhi nilai ambang batas (*treshold*) maka informasi tersebut akan diterima. Pada kasus tersebut neuran bisa dikatakan sudah diaktivasi. ANN merupakan sistem adatif yang dapat mengubah strukturnya untuk memecahkan suatu masalah berdasarkan informasi internal maupun eksternal. Seperti halnya otak manusia, selalu memiliki kemampuan untuk belajar dengan melakukan adaptasi (Adrian Rosebrock, 2017).

2.2.2. Karakteristik *Neural Network*

Neural network memiliki beberapa tipe yang berbeda, akan tetapi hampir semua komponen dan karakteristik yang dimiliki sama. Sama halnya dengan jaringan otak manusia, *neural network* juga terdiri dari beberapa *neuron* dan antar *neuron* juga berhubungan antar satu sama lain. *neuron-neuron* tersebut akan melakukan transformasi informasi yang diterima melalui sambungan keduanya menuju *neuron* yang lain berdasarakan bobot yang dimiliki antar *neuron*. Berikut komponen umum yang ada pada *neural network*.



Gambar 2. 5 Karakteristik *neural network* (elektronika-dasar.web.id, 2019)

Penyelesaian masalah dengan jaringan syaraf tiruan tidak memerlukan pemrograman. Jaringan syaraf tiruan menyelesaikan masalah melalui proses

belajar dari contoh-contoh pelatihan yang diberikan. Biasanya pada jaringan syaraf tiruan diberikan sebuah himpunan pola pelatihan yang terdiri dari sekumpulan contoh pola. Proses belajar jaringan syaraf tiruan berasal dari serangkaian contoh-contoh pola yang diberikan. Metode pelatihan yang sering dipakai adalah metode belajar terbimbing. Selama proses belajar itu pola masukan disajikan bersama-sama dengan pola keluaran yang diinginkan. Jaringan akan menyesuaikan nilai bobotnya sebagai tanggapan atas pola masukan dan sasaran yang disajikan tersebut (Hermawan, 2006).

- Faktor Bobot

  Bobot merupakan suatu nilai yang mendefinisikan tingkat atau kepentingan hubungan antara suatu *node* dengan *node* yang lain. Semakin besar bobot suatu hubungan menandakan semakin pentingnya hubungan kedua *node* tersebut.

  Bobot merupakan suatu hubungan berupa bilangan *real* maupun *integer*, tergantung dari jenis permasalahan dan model yang digunakan. Bobot-bobot tersebut bisa ditentukan untuk berada didalam interval tertentu. selama proses pelatihan, bobot tersebut dapat menyesuaikan dengan pola-pola *input*. Jaringan dengan sendirinya akan memperbaiki diri terus-menerus karena adanya kemampuan untuk belajar. Setiap ada suatu masalah baru, jaringan dapat belajar dari masalah baru tersebut, yaitu dengan mengatur kembali nilai bobot untuk menyesuaikan karakter nilai (Puspaningrum, 2006).

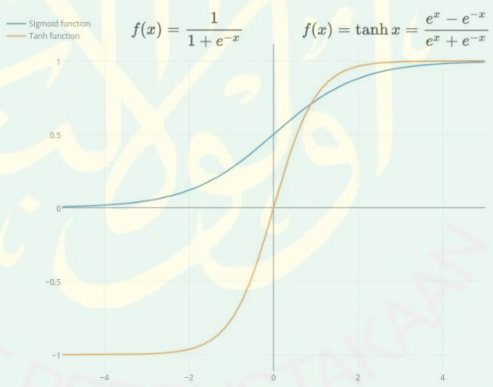
- Fungsi Aktivasi

  Fungsi aktivasi merupakan fungsi yang menggambarkan hubungan antara tingkat aktivitas internal (*summation function*) yang mungkin berbentuk

linear ataupun non-linear. befungsi untuk menentukan apakah *neuron* tersebut harus "aktif" atau tidak berdasarkan dari weighted sum (bobot dari hasil penjumlahan) dari *input*. Secara umum terdapat 2 jenis *activation function*, linear dan non-linear *activation function*. Secara umum, menurut artikel yang ditulis oleh Samuel sena (medium.com, 2017) ada 3 *activation function* yang seringdigunakan dalan jaringan *neural network*, diantaranya:

1) Fungsi Sigmoid and Tanh (Non-Linear)

Fungsi *sigmoid* dan fungsi *tanh* merupakan salah satu fungsi yang sering digunakan untuk fungsi aktivasi pada *neural network* multi *layer*. Kedua fungsi aktivasi ini, sama-sama fungsi aktivasi non-linear yang fungsi ini biasanya digunakan untuk klasifikasi 2 class atau kelompok data.

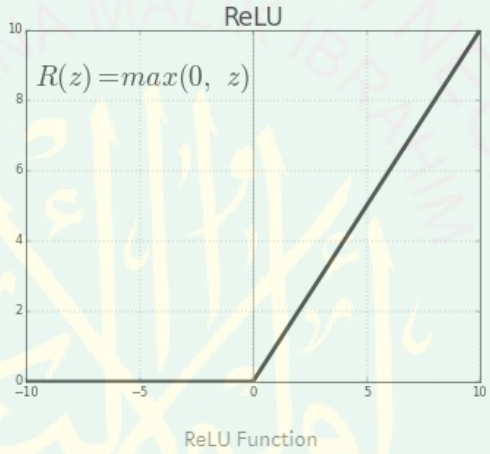


Gambar 2. 6 Grafik fungsi aktivasi *sigmoid & tanh* (Samuel Sena, 2018)

Fungsi *sigmoid* mempunyai rentang antara 0 hingga 1 sedangkan rentang dari fungsi *tanh* adalah -1 hingga 1. Kedua fungsi ini hampir sama, hanya saja fungsi tanh merupakan pengembangan dari fungsi *sigmoid*. Tapi kedua fungsi ini memiliki kelemahan yaitu, dapat mematikan *gradient*, ketika aktivasi dari *neuron* mengeluarkan nilai yang berada pada *range* 0 atau satu, dimana gradient di wilayah ini hampir bernilai 0. Kemudian *output* dari *sigmoid* tidak *zero-centered*.

2) Fungsi ReLu (non-linear)

ReLU atau *Rectified Linear Unit* menjadi salah satu *activation function* yang popular belakangan ini, Vincent Vanhoucke dalam course *deep learning*nya di udacity mengatakan bahwa ReLU merupakan *activation function* favorit para *engineer* yang malas. Karena ReLU pada intinya hanya membuat pembatas pada bilangan nol, artinya apabila $z \leq 0$ maka $z = 0$ dan apabila $z > 0$ maka $z = z$.



Gambar 2. 7 Grafik fungsi aktivasi *ReLu* (Samuel Sena, 2018)

ReLU bisa diimplementasikan hanya dengan membuat pembatas (*threshold*) pada bilangan nol. Fungsi aktivasi ini memiliki kelebihan yaitu dapat mempercepat proses konvigurasi yang dilakukan dengan *Stochastic Gradient Descent* (SGD) jika dibandingkan dengan fungsi *sigmoid* dan *tanh*. Namun aktivasi ini juga memiliki kelemahan yaitu aktivasi ini bisa menjadi rapuh pada proses *training* dan bisa membuat unit tersbut mati.

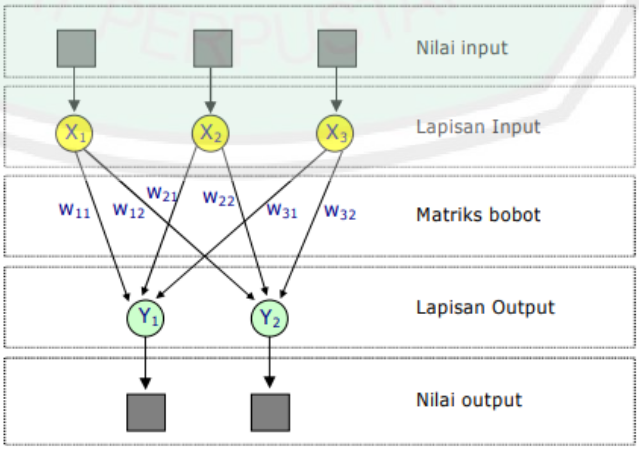
2.2.3. Arsitektur *Neural Network*

Pada *neural network*, *neuron-neuron* yang ada pada lapisan yang sama memiliki keadaan yang sama. Terdapat faktor penting dalam menentukan sifat suaru *neuron* yaitu bobot (*weight*) dan penggunaan fungsi aktivasi dari *neuron*

tersebut. Arsitektur yag dapat dibentuk oleh ANN bermacam-macam. Dari yang paling sederhana terdiri satu *neuron* (*single neuron*) sampai yang paling rumit menjadi multi *neuron* (*multiple neuron*) dalam satu lapis (*single layer*). Ada juga arsitektur dengan jaringan *multiple neuron* dalam *multiple layers*. Beberapa jaringan tersebut memiliki kemampuan yang berbeda-beda. Semakin rumit suatu jaringan, maka persoalan yang dapat diselesaikan menjadi lebih luas. Namun terdapat kelemahan yaitu kerumitan tersebut dapat menimbulkan persoalan tersendiri pada kebutuhan proses trainning dan simulasi (*testing*) yang akan memerlukan waktu lebih lama. Secara umur, algoritma *neural network* dibagi menjadi 3 arsitektur bedasarkan jumlah lapisannya, yaitu (David, 2005):

1) *Singel Layer Neural Net*

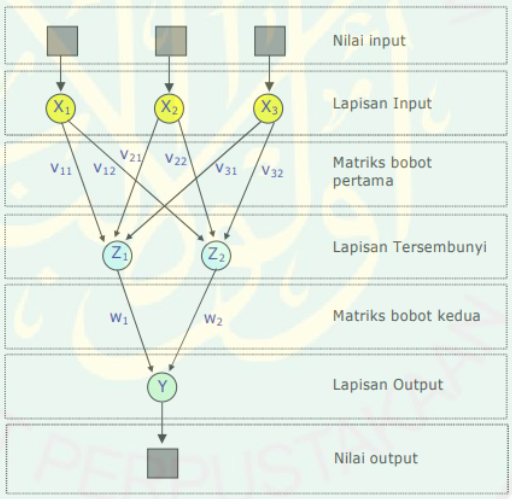
   Dari namanya bisa diketahui bahwa arsitektur ini hanya terdiri dari satu lapisan *input* dan satu lapisan *output*. Setiap *neuron* yang terdapat di dalam lapisan *input* selalu terhubung dengan setiap *neuron* yang terdapat pada lapisan *output*. Jaringan ini hanya menerima *input* kemudian secara langsung akan mengolahnya menjadi *output* tanpa harus melalui lapisan tersembunyi :



Gambar 2. 8 *Single Layer Neural Net* (Miracle *et al*., 2018)

2) *Multi Layer neural network*

Arsitektur model ini sering disebut dengan *Multi Layer Perceptron* (MLP) yang terdiri dari banyak lapisan, umumnya memiliki satu atau lebih lapisan yang terletak diantara lapisan *input* dan lapisan *output*, yang dikenal dengan lapisan tersembunyi (*hidden layer*). Umumnya, ada lapisan bobot- bobot yang terletak antara 2 lapisan yang bersebelahan. Arsitektur *neural network* dengan banyak lapisan ini umunya digunakan untuk menyelesaikan permasalahan yang lebih sulit dengan pembelajaran yang lebih rumit. Dengan arsitektur seperti ini, terbukti mampu untuk menyelesaikan banyak kasus dengan tingkat akurasi yang lebih tinggi.



Gambar 2. 9 *Multilayer Neural Net* (Miracle *et al*., 2018)

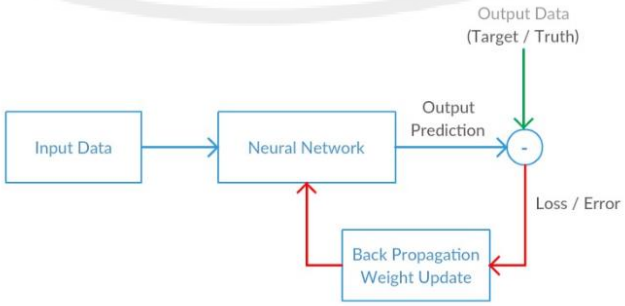
Pada tiap koneksi memiliki nilai bobot yang nilai dari setiap bobot itu berbeda-beda. Kemudian pada *hidden layer* dan *output layer* juga memiliki tambahan nilai "*input*" yang biasa disebut dengan bias. Proses untuk mendapatkan nilai *output* pada *neural network* bisa dihitung menggunakan rumus

$$dot_j = w_{ji}x_i + b_j \quad (2.1)$$

Persamaan diatas merupakan persamaan dari proses perkalian bobot pada satu *node* ouput pada arsitektur *neural network* yang menggunakan fungsi aktivasi ReLu. Dimana *i* adalah *node* pada *input layer*, kemudian *j* adalah *node* pada hidden *layer* atau *output layer* dan *b* adalah bias pada hidden *layer* atau *output layer*.

2.2.4. *Backpropagation*

Neural network merupakan suatu model komputasi yang sistemnya mengikuti syaraf otak manusia. Dan salah satu ciri khas dari otak manusia adalah melakukan pembelajaran berdasarkan pengalaman. *neural network* juga begitu, model ini membutuhkan proses pembelajaran untuk mengenali pola dari data yang dipelajarinya. Pembelajaran ini bertujuan untuk melakukan suatu proses dalam menentukan nilai bobot (*weight*) yang tepat untuk masing-masing *input*. Proses untuk melakukan pembalajaran ini biasanya terjadi pada saat proses pelatihan data (*training* data) biasanya disebut dengan *backpropagation*. *Backpropagation* merupakan salah satu dari metode pelatihan pada jaringan syaraf, dimana ciri dari metode ini adalah meminimalkan *error* pada *output* yang dihasilkan oleh jaringan (Puspita & Eunike, 2007). Jadi *backpropagation* ini bekerja dengan cara melakukan *update* nilai bobot pada *neuron* di *layer* sebelumnya.



Gambar 2. 10 *Backpropagation* (Samuel Sena, 2018)

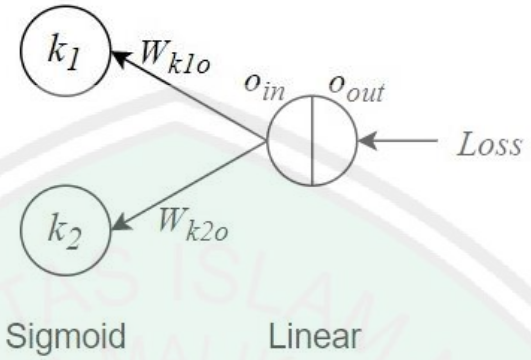
Seperti pada gambar diagram diatas tugas dari metode *backpropagation* adalah dengan melakukan *update* bobot berdasarkan nilai *error* yang didapatkan dari perbandingan antara nilai *output* dan target *output*. Proses *training* menggunakan *backpropagation* secara garis besar terdiri dari 2 tahap ; tahap maju (*forward pass*) dan tahap mundur (*backward pass*) (Ciresan *et al*., 2011).

Pada tahap maju, akan dilakukan operasi dot antar nilai bobot pada *node input* dengan *node* pada *layer* di depannya. Operasi ini dilakukan untuk mendapatkan nilai ouput, proses ini disebut dengan forward pass. Kemudian hasil prediksi *output* akan dibandingakan dengan target dengan menggunakan sebuah fungsi yang biasa disebut dengan *loss function*. Secara sederhana *loss function* adalah fungsi yang digunakan untuk mengukur seberapa bagus performa dari *neural network* kita dalam melakukan prediksi terhadap target (Lisa, 2015). Ada berbagai macam *loss function* yang bisa digunakan, namun yang paling sering digunakan adalah *squared error*, *loss function* yang sering digunakan untuk linier regresi. Sedangkan untuk klasifikasi dengan beberapa kelas yang biasa digunakan adalah *Cross Entropy* (Katarzyna & Wojciech, 2017). Secara sederhana rumus yang digunakan untuk mencari *loss function* menggunakan squared *error* adalah seperti ini

$$Loss = (prediction - target)^2 \tag{2.2}$$

Kemudian pada tahap kedua akan dilakukan propagasi mundur (backward pass). Tahapannya hampir sama dengan *forward pass*, pada *backward pass* nilai *loss* yang didapatkan akan mengalir dan dijadikan sebagai bobot pengali menuju semua *node* pada *layer* sebelumnya untuk dicari *gradient* nya. Semisal ingin melakukan *update* pada parameter W_{k1o}, pertama akan dicari seberapa besar

perubahan *loss* berdasarkan *output* dengan cara mencari turunan parsial dari *loss function* terhadap *output*.



Gambar 2. 11 *Backpropagation* tahap pertama (medium.com, 2019)

Setelah mendapatkan nilai *gradient loss* dari parameter yang dituju, maka akan diberlakukan *update* nilai dari parameter tersebut menggunakan algoritma *Stochastic Gradient Descent* (SGD). Secara sederhana, algoritma SGD ini bertujuan untuk menurunkan nilai *loss* pada parameter dengan cara mengurangi bobot asli dari parameter tersebut dengan "sebagian" dari nilai *gradient loss* yang sudah didapatkan sebelumnya. Sebagian disini diwakili oleh nilai dari *learning rate* (Samuel sena, 2017). Jadi SGD ini akan menurunkan nilai *loss* pada parameter secara terus-menerus hingga mencapai titip minimum secara optimal sebanyak iterasi yang terjadi. Rumus untuk melakukan *update* parameter menggunakan SGD sebagai berikut

$$W'_{k1o} = W_{k1o} - \alpha\left(\frac{\partial Loss}{\partial W_{k1o}}\right) \quad (2.3)$$

Dari persamaan diatas, untuk melakukan *update* pada parameter W_{k1o} maka nilai bobot asli dari parameter W_{k1o} akan dikurangi dengan sebagian dari nilai gradient yang didapat. Sebagian disini adalah nilai dari *learning rate* (α) yang dikalikan dengan nilai *gradient loss* yang sudah dicari sebelumnya. Proses ini

akan dilakukan terus-menerus pada semua parameter yang terhubung hingga sampai pada *node input*. Setelah itu akan dilakukan proses *forward pass* dan *backward pass* lagi. Proses ini dilakukan terus-menerus hingga proses iterasi selesai.

2.3. *Deep learning*

Teknik *Deep learning* adalah teknik *machine learning* yang mengajarkan komputer agar memiliki kemampuan menyerupai kemampuan alami manusia, yaitu belajar dari pengalaman. Dengan adanya teknologi *deep learning*, produk-produk seperti *self-driving car*, *face recognition* atau voice *recognition* dapat dihasilkan. Ini adalah kunci untuk mengontrol suara di perangkat konsumen seperti ponsel, tablet, TV, dan dan lainnya. Teknik *deep learning* akhir-akhir ini mendapatkan banyak perhatian dari para peneliti karena pencapaian hasilnya yang bagus.

Komputer akan memodelkan pembelajaran untuk melakukan klasifikasi langsung dari gambar, teks, atau suara ketika menggunakan *deep learning*. Penggunakan *deep learning* ini dapat mencapai akurasi ketepatan yang tinggi, terkadang melebihi kinerja tingkat manusia pada beberapa kasus. Model dilatih dengan menggunakan sejumlah data yang besar dan dipadukan dengan arsitektur *neural network* yang mengandung banyak lapisan.

Meskipun istilah *deep learning* pada *neural network* sudah ada sejak 2006, tapi baru beberapa tahun kebelakang istilah ini dimunculkan lagi. Hal itu dikarenakan 2 hal, yang pertama, metode *deep learning* membutuhkan *dataset* dalam jumlah besar, dengan begitu kemampuan akurasi yang tinggi dari *deep learning* ini baru terlihat benar-benar nyata. Alasa kedua, *deep learning* dengan *dataset* yang besar membutuhkan perangkat kompuasi dengan performa yang tinggi

untuk meningkatkan kecepatan pemrosesan. Alasan kedua inilah yang menjadi sebab metode *deep learning* kini menjadi sangat digemari oleh peneliti, karena saat ini perangkat komputasi seperti GPU dengan performa tinggi dan *cloud computing* yang semakin berkembang, membuat metode ini cepat berkembang (Hafizan & Dina, 2018).

Salah satu metode *deep learning* yang memanfaatkan deep *neural network* adalah Convolusional *neural network* (CNN atau CovNets). Nama tersebut diambil dari operasi terpenting yang ada pada metode tersebut yaitu *Convolutional*, yang kemudian digabung dengan metode *neural network*. CNN ini merupakan model terbaru dari algoritma deep *neural network* yang dikembangkan untuk memproses data dalam bentuk 2D data, seperti gambar. Berbeda dari algoritma pengolahan citra lainnya, algoritma CNN ini menghilangkan ekstraksi fitur secara manual. Jadi tidak perlu untuk mengekstraksi fitur untuk mengklasifikasikan gambar tersebut. Fitur-fitur yang relevan tidak perlu dilatih terlebih dahulu, mereka akan mempelajari fitur dari sebuah gambar ketika jaringan tersebut dijalankan. Ekstraksi fitur otomatis ini membuat model *deep learning* memiliki akurasi tinggi untuk melakukan tugas *computer vision* seperti klasifikasi objek.

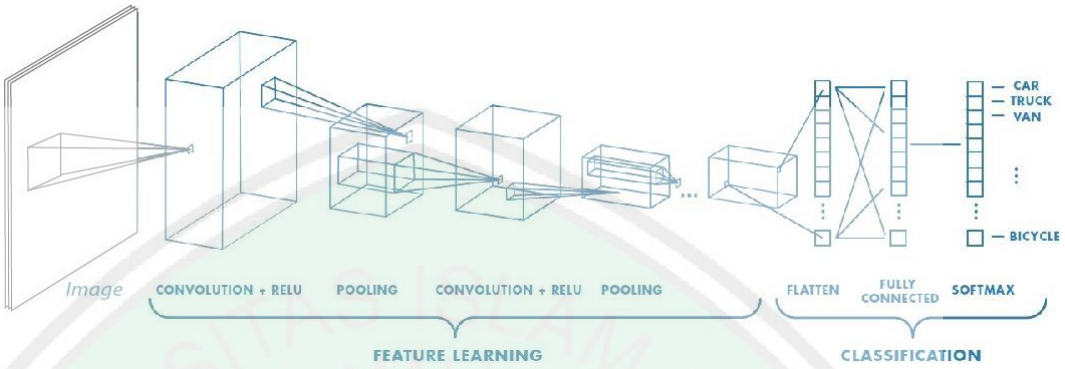
2.4. *Convolutional Neural Network*

Convolutional Neural Network (CNN) adalah salah satu algoritma paling populer digunakan untuk *deep learning*, sebuah *machine learning* yang model pembelajarannya dikhususkan untuk melakukan klasifikasi langsung pada media 2 dimensi seperti gambar, video, teks atau suara. Algoritma CNN akan sangat berguna khususnya ketika digunakan untuk mencari pola pada suatu gambar kemudian mengenali objek pada gambar tersebut. Bukan hanya pada objek atau

benda saja, CNN ini sebenarnya juga bisa digunakan untuk mengenali wajah yang selama ini perlu segmentasi untuk meningkatkan akurasinya. Penelitian awal yang mendasari penemuan CNN ini pertama kali dilakukan oleh Hubel dan Wiesel (Hubel & Wiesel, T, 1968) mengenai visual *cortex* pada indera penglihatan kucing. Pada dasarnya klasifikasi citra menggunakan MLP sudah bisa dilakukan, akan tetapi ketika digunakan untuk melakukan klasifikasi data dalam jumlah banyak, akurasi yang didapatkannya pun menurun. Oleh karena itu, algoritma CNN ini dikembangkan karena algoritma ini mampu untuk mempelajari langsung data yang ada pada gambar, kemudian menggunakan pola yang didapatkan untuk mengklasifikasi.

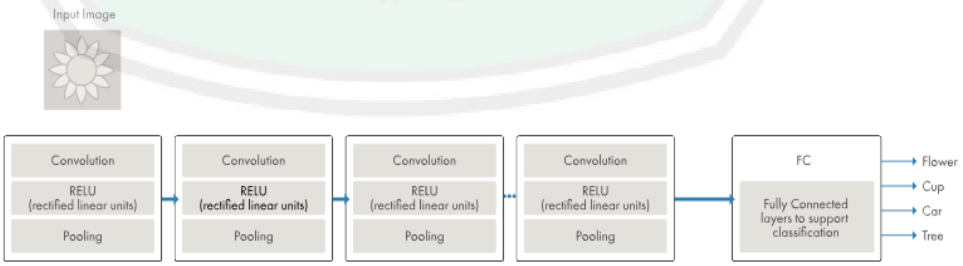
Berbeda dengan arsitektur MLP, arsitektur CNN ini sebenarnya lebih kompleks dan memiliki proses yang cukup panjang sebelum masuk tahap klasifikasi. Namun secara garis besar ada 2 tahapan pemrosesan yang dilakukan oleh algoritma CNN ini, yaitu tahap *feature learning* dan tahap classification. Tahap *feature learning* merupakan tahap dimana gambar yang di*input*kan akan diekstraksi untuk dipelajari value dari gambar tersebut. Proses inilah yang membedakan algoritma CNN dengan MLP. Jika MLP, hanya bisa menggunakan satu proses ekstraksi dalam sekali *input* untuk melakukan klasifikasi. Sedangkan CNN ini, untuk melakukan klasifikasi bisa menggunakan banyak sekali ekstraksi dalam sekali *input*. Banyaknya ekstraksi ini disimpan dalam bentuk kedalaman gambar (*depth*). Proses *feature learning* ini sangat bergantung pada kedalaman suatu gambar. Semakin dalam suatu gambar maka semakin banyak ekstraksi yang didapatkan sehingga pola yang didapat juga semakin jelas terbentuk (Hui Li *et al*., 2018). Value inilah yang nantinya akan dikonversi menjadi vektor dan kemudian

masuk pada tahap kalsifikasi. Pada tahap klasifikasi ini, model *neural network* akan digunakan untuk melakukan klasifikasi objek berdasarkan kelasnya.



Gambar 2. 12 Arsitektur CNN

Berdasarkan gambar diatas, arsitektur dari algoritma dibagi menjadi 2 tahapan pemrosesan. Pada tahap *feature learning*, secara umum ada 3 lapisan proses ekstraksi fitur. Lapisan-lapisan ini sering disebut dengan *covolution layer*, *activation* dan *pooling layer*. Lapisan-lapisan ini akan melakukan operasi khusus untuk membentuk kedalaman data agar mendapatkan pola secara spesifik. Ketiga *layer* tersebut memiliki urutan proses yang tidak harus selalu sama, dalam artian prosesnya bisa dimodifikasi sesuai dengan kebutuhan. Tapi umumnya proses *feature learning* ini diawali dengan melakukan proses konvolusi antara matriks *input* dengan *kernel* ukuran tertentu.



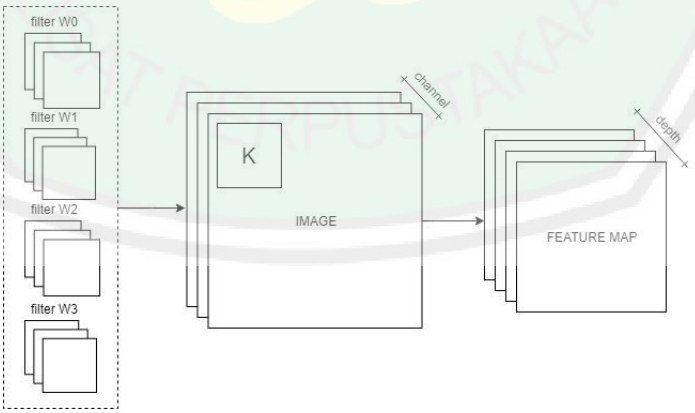
Gambar 2. 13 Urutan lapisan CNN (mathworks.com, 2019)

2.4.1. *Convolution layer*

Convolution layer merupakan proses pertama yang harus dilalui dalam tahapan *feature learning*. Pada *convolution layer*, akan dilakukan operasi

konvolusi antara matriks *input* dengan *kernel* yang ada pada matriks *filter*. Konvolusi adalah operasi perkalian antara dua matriks yang kemudian hasilnya dijumlahkan (Adrian Rosebrock, 2017). Hasil dari proses konvolusi pada algoritma CNN ini disebut dengan *feature map*.

Convolution layer ini adalah bagian terpenting dalam membentuk kedalaman (depth) data pada suatu *feature*. Sebagai *input*-an, kedalaman suatu gambar didefinisikan dengan banyaknya channel gambar tersebut. Sebagai contoh, jika gambar tersebut berukuran 32x32x3, angka 3 yang menotasikan jumlah *layer* warna pada gambar tersebut juga bisa dikatakan sebagai ukuran kedalaman dari gambar tersebut. Karena proses *learning* pada algoritma CNN ini tergantung pada kedalaman data, untuk membuat data *input*an lebih dalam lagi maka *input*an akan dikonvolusi dengan sejumlah matrik K (*kernel*) yang disebut dengan *filter* pada tahap *convolution layer* ini. Dimana semua *filter* memiliki ukuran yang hampir selalu persegi dengan kedalaman sesuai dengan yang ditentukan.

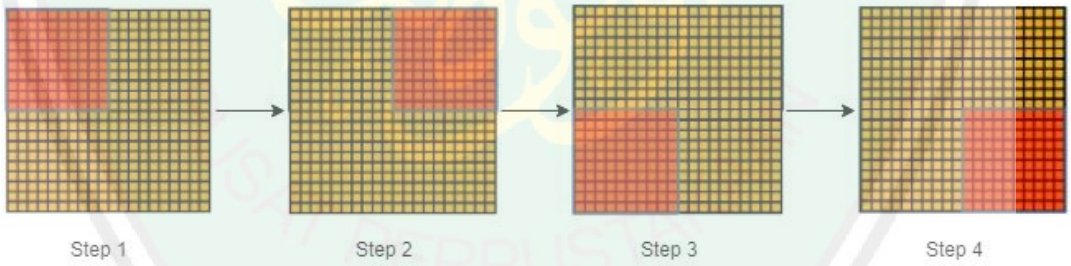


Gambar 2. 14 Konvolusi untuk membentuk kedalaman

Proses konvolusi antara gambar *input* dengan sejumlah *filter* akan menghasilkan satu *set feature map* yang berisi berbagai variasi ekstraksi seperti *edge detection*, *smoothing*, *sharpening* dan sebagainya.

- *Stride*

Berdasarkan gambar diatas, *feature map* didapatkan dari proses konvolusi antara gambar dengan sejumlah K *filter* . Proses perhitungan untuk mendapatkan *feature map* tersebut adalah dengan cara melakukan operasi perkalian antara matriks pada gambar dengan matriks yang kecil pada K *filter*. Proses perkalian ini dimulai dari bagian pojok kiri atas matriks pada gambar. Hasil perkalian dari setiap elemen pada matriks tersebut kemudian akan dijumlahkan untuk menghasilkan satu elemen baru dari hasil perhitungan tersebut. Setelah itu matriks K *filter* tersebut akan digeser ke kanan, sejumlah "*stride*" yang ditentukan. *Stride* (*S*) adalah parameter yang menentukan berapa jumlah pergeseran *filter* yang akan digunakan. Jika nilai *stride* adalah 1, maka K *filter* akan bergeser sebanyak 1 piksel secara horizontal ke kanan hingga selesai, kemudian baru bergeser lagi secara vertikal ke bawah.



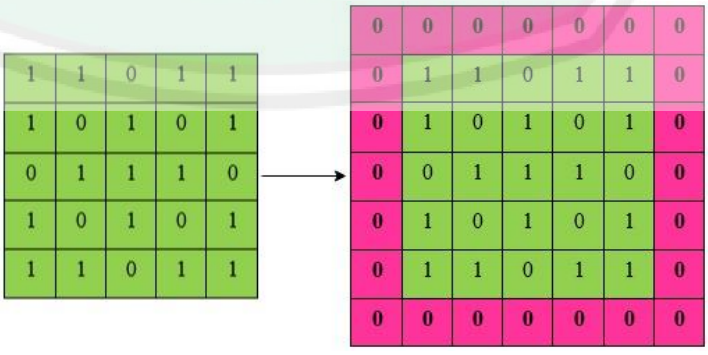
Gambar 2. 15 *Stride* konvolusi

Seperti pada gambar diatas, bagian pada matriks gambar yang sedang terjadi proses konvolusi ditandai oleh blok yang berwarna merah. Pada konvolusi pertama, sebagian dari matriks gambar akan dikonvolusi dengan *kernel* seluas volume *kernel* tersebut dan konvolusi ini diawali dari bagian pojok kiri matriks gambar. Setelah hasil konvolusi selesai dijumlahkan, *kernel* akan digeser ke kanan sebesar jumlah *stride* yang ditentukan seperti pada step 2. Dalam menentukan jumlah *stride* ini juga tidak boleh sembarangan, harus

memastikan jumlah *stride* yang ditentukan tersebut sesuai atau pas dengan luas matriks gambar. Jika ternyata *stride* yang dientukan ternyata tidak pas, bisa menambahkan *padding* pada setiap sisi matriks, atau mengurangi jumlah *stride*.

- *Padding*

  Padding adalah parameter yang menentukan jumlah piksel (berisi nilai 0) yang akan ditambahkan disetiap sisi dari matriks *input*. *Padding* ini digunakan untuk memanupulasi dimensi *output* dari *feature map*. *Padding* (P) ini bisa dibilang salah satu parameter yang cukup penting untuk mendapatkan hasil konvolusi yang cukup dalam. Karena dimensi *output* dari proses konvolusi ini selalu lebih kecil dari dimensi *input*nya. Sementara *output* akan digunakan lagi pada *convolution layer* berikutnya, otomatis informasi pada gambar akan makin banyak terbuang, apalagi jika *filter size* yang digunakan cukup besar. Dengan menggunakan *padding*, dimensi *output* dapat dimanipulasi agar memiliki dimensi yang sama dengan *input* atau setidaknya tidak berkurang secara drastis. Sehingga bisa menggunakan *convolution layer* yang lebih deep dan mendapatakan lebih banyak ekstraksi dari *input*.



Gambar 2. 16 Penambahan *padding*

Seperti gambar diatas, matriks gambar yang awalnya berukuran 5x5 kemudian ditambah dengan *padding* 1 (P =1) dengan begitu pada setiap sisi-sisi matriks akan ditambahkan 1 piksel dengan nilai 0 pada setiap sisinya. Penggunaan *padding* ini selain untuk memanipulasi *feature map* nantinya, teknik ini juga bermanfaat untuk meningkatkan performa *convolution layer*. Karena dengan menambahkan *padding*, maka *filter* akan fokus pada informasi yang berada diantara *padding* tersebut.

Seperti yang sudah dijelaskan sebelumnya dalam menentukan *stride* haruslah pas dengan volume matriks gambar. Salah satu cara untuk melihat bahwa *stride* yang akan digunakan sesuai atau tidak, bisa menggunakan rumus untuk menghitung dimensi dari *feature map* ini. Jika hasil *output* berupa bilangan integer, maka *stride* yang digunakan sudah sesuai.

$$output = \frac{W-N+2P}{S} + 1 \quad (2.4)$$

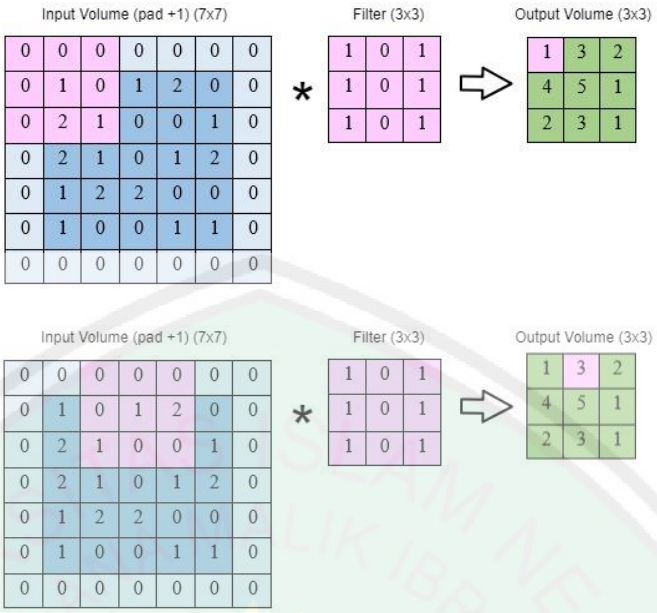
W = panjang/tinggi *input*

N = panjang/tinggi *filter*

P = *padding*

S = *stride*

Sebagai contoh dari proses konvolusi pada *convolution layer* ini, misalnya memiliki *input* dengan ukuran 4x4 yang akan di konvolusi dengan *kernel* berukuran 3x3 dengan *stride* 2 dan *padding* 1. Jika dihitung menggunakan rumus diatas maka *feature map* yang didapatkan adalah 2, berarti *feature map* yang didapatkan adalah matriks berukuran 2x2 seperti berikut ini



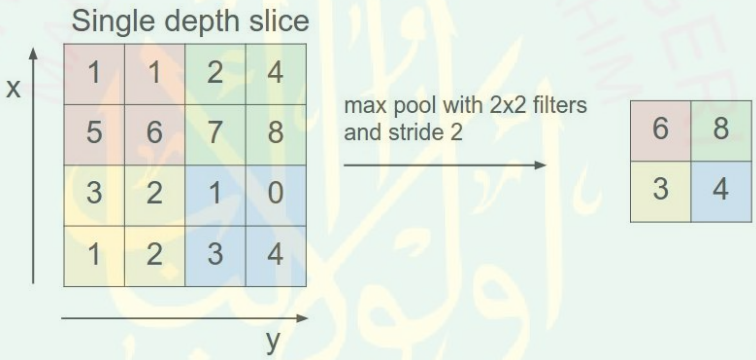
Gambar 2. 17 Proses konvolusi 1&2

Setelah semua matriks pada *input* selesai dikonvolusi, proses selanjutnya adalah menentukan bobot dari *output* tersebut harus "aktif" atau tidak menggunakan fungsi tertentu. Pada kasus CNN ini, fungsi akivasi yang sering digunakan untuk mengaktifkan hasil dari proses konvolusi adalah ReLu (cs231n, 2017).

2.4.2. *Pooling Layer*

Ada 2 metode untuk mengurangi ukuran dari volume *input*, yang pertama *convolution layer* dengan *stride* >1 seperti contoh diatas dan yang kedua adalah *pooling layer*. Merujuk pada arsitektur CNN diawal pembahasan tadi, tahap *pooling layer* ini terletak setelah *convolution layer*. Pada dasarnya *pooling layer* ini terdiri dari sebuah *filter* dengan ukuran tertentu yang akan dioperasikan dengan *stride* tertentu ke *feature map* hasil *convolution layer*. Ada dua metode *pooling* yang biasa digunakan di *pooling layer* ini yaitu *average pooling* dan max *pooling*. Perbedaan dari 2 metode ini adalah pada nilai yang diambil. Jika *average pooling*, nilai yang diambil adalah nilai rata-rata dari matriks yang mengalami operasi

pooling. Sedangkan max *pooling* adalah dengan mengambil nilai tertinggi. Lapisan *pooling* yang dimasukkan diantara lapisan konvolusi secara berturut-turut dalam arsitektur model CNN dapat secara progresif mengurangi ukuran volume *output* pada *feature map*, sehingga mengurangi jumlah parameter dan perhitungan di jaringan, untuk mengendalikan Overfitting. Lapisan *pooling* bekerja di setiap tumpukan *feature map* dan melakukan pengurangan pada ukurannya. Sebagai contoh menggunakan max *pooling* 2x2 dengan *stride* 2, maka pada setiap pergeseran *filter* , nilai maximum pada area 2x2 piksel tersebut yang akan dipilih seperti contoh berikut :



Gambar 2. 18 *Max pooling*

Berdasarkan ilustrasi diatas, pada *feature map* hasil konvolusi dilakukan oprasi max *pooling*. Operasi untuk mengambil nilai tertinggi pada batasan ukuran *filter* tertentu. Dari proses *max pooling* dengan *filter* 2x2 dan *stride* 2 tersebut didapatkan *feature map* baru dengan ukuran yang lebih kecil yaitu 2x2 (cs231n, 2017).

2.4.3. *Fully connected layer*

Fully connected layer merupakan lapisan dimana semua *neuron* dari lapisan-lapisan sebelumnya dijadikan satu untuk dilakukan proses klasifikasi menggunakan *neural network*. Pada dasarnya lapisan ini sama halnya dengan

lapisan *neural network* biasa, bisa dalam bentuk *single net* ataupun MLP. Tapi sebelum dilakukan proses klasifikasi, *feature map* yang dihasilkan dari *feature learning* tersebut masih berbentuk *multi dimensional array*, sehingga perlu mengubahnya menjadi bentuk *vector*, tehnik ini sebut dengan *flatten*. *Flatten* merupakan tehnik untuk *reshape feature map* menjadi sebuah vector agar bisa digunakan sebagai *input* dari *fully connected layer*. Jadi *input* dari *fully connected layer* terdiri dari satu *neuron* hasil *reshape feature map* tadi menjadi vektor. Setelah dilakukan *flatten*, semua bobot tersebut akan diklasifikasi sesuai dengan banyaknya kelas.

Pada *fully connected layer* ini, tidak ada ketentuan pasti untuk menggunakan jaringan *single net* ataupun MLP. Karena proses ekstraksi dan *feature learning* sudah dilakukan pada *layer* sebelumnya, beban jaringan *neural network* untuk melakukan proses klasifkasi relatif lebih ringan. Tapi dalam beberapa penelitian sebelumnya, menunjukkan dengan menggunakan jaringan MLP dapat meningkatkan akurasi klasifikasi meskipun perbedaannya tidak terlalu signifikan (Adrian Rosebrock, 2017).

2.5. *Related Research*

Beberapa tahun ini penelitian tentang *deep learning* mulai banyak bermunculan dan sedang hangat-hangatnya dibahas khususnya algoritma CNN. Bahkan beberapa instansi penyedia *image dataset* pun banyak memberikan tantangan untuk menemukan model dengan akurasi paling tinggi. Penelitian awal yang mencoba model CNN ini untuk *object recognition* pada data dengan skala dan jumlah yang besar dilakukan oleh Alex Krizhevsky *et al*., (2012). Peneliti melakukan klasifikasi pada 1,2 juta gambar *high resolution* (224x2224) dengan

1000 kelas berbeda yang disediakan oleh ImageNet. Pada penelitian tersebut, peneliti menggunakan 5 *convolution layer* yang diikuti dengan fungsi aktivasi ReLu dan max *pooling*. Kemudian pada *fully connected layer*, peneliti mennunakan 3 *layer* dengan 4096 neuran pada 2 hidden *layer* dan 1000 *neuron* sofmax. Dalam pemrosesannya, peneliti juga menggunakan GPU untuk meningkatkan pemrosesan gambar. Dari penelitian tersebut hasilnya memuaskan, dalam *challenge* yang diadakan *Image*Net tersebut modelnya mendapatkan peringkat pertama dengan *error* rate 15,3%. Model CNN ini kemudian dikenal dengan nama AlexNet.

Setelah penelitian tersebut muncul lagi penelitian yang dilakukan oleh Karen & Andrew (2015). Penelitian ini juga dilatarbelakangi oleh *challenge* yang diadakan *Image*Net 2104. Arsitektur CNN ini kemudian lebih dikenal dengan nama VGGC-16. Arsitektur yang digunakan oleh peneliti sebenarnya hampir sama dengan arsitektur AlexNet, hanya saja berbeda pada volume dan kedalaman *filter* yang digunakan. Jika AlexNet menggunakan volume *filter* 11x11 dengan *stride* 4, VGGC-16 menggunakan volume yang sangat kecil yaitu 3x3 dengan *stride* 1. Hasil yang didapatkanpun meningkat cukup pesat, jika *error* rate AlexNet sekitar 15%. VGCC-16 memiliki *error* rate 7.1% yang berarti VGCC-16 memiliki tingkat akurasi diatas 90% dengan dengan data 1,2 juta gambar hight resolution. Selain menggunakan model tersebut peneliti juga melakukan penelitian pada 5 model lain dengan jumlah *layer* yang berbeda. Tapi dari 5 model tersebut jaringan VGCC-16 (menggunakan 16 *layer*) yang memiliki akurasi paling tinggi, meskipun ada model dengan jumlah *layer* lebih banyak.

Melihat akurasinya yang cukup tinggi, algoritma CNN inipun terus dikembangkan penggunaanya. Hyeonseob & Bohyung (2016) mengembangkan

algoritma CNN ini melakukan pembelajaran pada objek multi-domain tepatnya untuk visual tracking. Pada penelitian tersebut gambar *input*-an memiliki volume lebih kecil daripada 2 penelitian sebelumnya yaitu 107x107x3. Model arsitektur yang digunakannyapun juga lebih sederhana, hanya menggunakan 5 *layer* yang terdiri dari 3 *layer convolution layer* dan 2 *fully connected layer*. Sedangkan volume *filter* yang digunakan yaitu 3x3 dengan *stride* 2 dan menggunakan 512 *neuron* pada 2 hidden *layer*. Hasil dari penelitian tersebut memiliki *error* rate sebesar 2,29%.

Penelitian lainnya juga dikembangkan untuk mengidentifikasi ekspresi wajah. Penelitian tersebut dilakukan oleh Liu Kuang *et al*., (2016), dengan judul "Facial Expression Recognition with CNN Ensemble". Pada penelitian tersebut, para peneliti melakukan pengenalan ekspresi pada wajah menggunakan algoritma CNN. Data ekspresi yang digunakan merupakan dataset yang diambil dari google image berdasarkan keyword tertentu yang kemudian dirubah menjadi grayscale dengan ukuran 48x48. Pada paper tersebut, mereka menggunakan 3 model arsitektur untuk membandingkan arsitektur mana yang paling bagus hasilnya. Tiga arsitektur yang dibangun tersebut merupakan arsitektur dengan 3 layer, 4 layer dan 5 layer konvolusi. Sementara bagian FC layer menggunakan 3 layer dengan 4096 neuron pada layer 1 & 2. Model tersebut kemudian ditraining mulai dari 45-100 epoch. Hasil yang didapatkan dari penelitian tersebut, arsitektur ketiga menghasilkan akurasi paling tinggi dengan nilai akurasi rata-rata sebesar 79,8%. Sedangkan hasil testing pada single expression, ekspresi terkegut memiliki tingkat akurasi paling tinggi yaitu sebesar 81,2%.

Berawal dari penelitian Liu Kuang *et al*., tersebut, penelitian tentang facial expression recognition dikembangkan lagi oleh Andre Lopes *et al*., (2016) untuk

mendapatkan hasil yang lebih baik. Pada penelitian tersebut mereka mencoba melakukan beberapa preprocessing image sebelum image diproses menggunakan algoritma CNN. Preprocessing yang dilakukan berupa cropping image pada area yang menunjukkan ekspresi. Kemudian merubah gambar menjadi grayscale, dan yang terakhir adalah melakukan augmentasi data. Sedangkan dataset yang digunakan pada penelitian tersebut merupakan public dataset yang diambil dari CK+ database, JAFFE database and BU-3DFE database dengan 6 kelas. Dan untuk arsitektur yang digunakan, mereka hanya mengaplikasikan satu model arsitektur saja dengan 2 layer konvolusi dan 2 FC layer. Dengan ukuran image 32x32, mereka berhasil meningkatkan akurasi rata-rata yang didapatkan menjadi 98,9% pada 6 ekspresi.

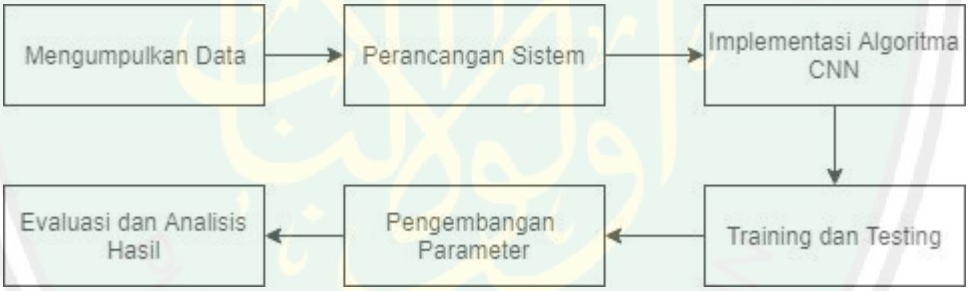
BAB III

METODE PENELITIAN

Pada bab ini akan dijelaskan tentang analisa dan perancangan sistem dari penelitian ini. Terdapat beberapa tahapan, yaitu tahapan penelitian yang dilakukan, kebutuhan sistem yang akan dibuat dan penyelesaian masalah identifikasi ekspresi wajah menggunakan algoritma *Convolutional Neural Network* (CNN.

3.1. Alur Penelitian

Dalam melakukan penelitian ini, untuk mempermudahnya maka dijabarkan langkah-langkah apa saja yang akan diambil dalam melakukan penelitian ini. Alur dari penelitian ini direprentasikan pada gambar dibawah ini.



Gambar 3. 1 Alur penelitian

Berdasarkan pada gambar dibawah ini dapat dilihat langkah-langkah apa saja yang harus dilakukakan untuk mendapatkan hasil dari penelitian ini. Langkah pertama yang dilakukan adalah mengumpulkan data untuk penelitian, data ini berupa data *training* dan data *testing*. Bab tinjauan pustaka disajikan bukan hanya sebagai informasi atau materi tambahan saja, tetapi itu juga sebagai dasar penelitian ini dijalankan sekaligus memuat perbandingan antara penelitian ini dengan penelitian lainnya. Sehingga hal tersebut bisa sebagai pembelajaran untuk

membantu berjalannya penelitian ini. Langkah selanjutnya yang harus dikerjakan adalah proses perancangan sistem, pada proses ini sistem yang akan dibangun pada penelitian ini akan didefinisikan jalannya sistem yang akan dibangun pada penelitian ini. Setelah bagian ini selesai didefinisikan, berikutnya bagaimana mengimplementasikan metode dan algoritma yang akan digunakan pada sistem ini. Seperti pada judul penelitian ini akan menggunakan salah satu algoritma *deep learning* yaitu *Convolutional Neural Network* (CNN). Pada penelitian ini algoritma *Convolutional Neural Network* (CNN) akan diimplementasikan untuk mengenali objek benda yang di*input*kan pada sistem. Algoritma CNN ini dipilih karena merupakan salah satu algoritma *deep learning* yang memiliki tingkat akurasi yang cukup tinggi. Oleh karena itu penulis berharap dengan mengunakan metode ini, dapat membangun sistem yang memiliki tingkat kecerdasan untuk mengenali benda dengan akurasi tinggi layaknya kemampuan manusia.

Setelah metode berhasil diimplemenasikan pada sistem yang dibangun ini, maka akan dilakukan *training* dan *testing* pada sistem. *Training* pada sistem ini bertujuan untuk memeperkenalkan dan melatih sistem yang telah dibangun tersebut untuk mengenali benda yang yang akan di ujikan nanti. Dengan melakukan *training* dengan sejumlah data *training* yang sudah disiapkan, sistem kemudian akan siap untuk dilkukan uji coba untuk mengenali benda. Pada tahap *testing*, penulis berharap sistem akan mampu untuk mengenali benda yang dimaksudkan dan menampilkan akurasi ketepatan pengenalan benda. Akurasi yang didapatkan akan dijadikan sebagai acuan utama untuk tahap selanjutnya, yaitu tahap pengembangan parameter. Pada tahap ini penulis akan mencoba untuk mengembangkan beberapa parameter yang mungkin bisa mempengaruhi akurasi sistem. Hal ini dilakukan

bertujuan untuk membandingkan beberapa parameter dan mendapatkan akurasi yang paling tinggi. Evaluasi dan analisis dari hasil ujicoba ini diperlukan untuk membahas dan mengetahui seberapa besar kemampuan sistem dalam melakukan identifikasi ekspresi.

3.2. Pengumpulan Data

Pada penelitian kali ini, untuk keperluan identifikasi ekspresi wajah maka diperlukan objek berupa foto ekspresi wajah untuk diidentifikasi. Seperti yang sudah dijelaskan pada latar belakang, pada penelitian kali ini studi kasus yang akan digunakan adalah foto ekspresi wajah mahasiswa UIN Malang. Objek penelitian ini nantinya akan difoto secara manual menggunakan kamera smartphone. Sedangkan untuk ekspresi yang akan diidentifikasi pada penelitian ini dibagi menjadi 3 kategori ekspresi, yaitu:

1. Netral
2. Senyum
3. Marah

Jadi mahasiswa diminta untuk berekspresi senatural mungkin untuk memperlihatkan ketiga ekspresi tersebut. Untuk memvalidasi bahwa ekspresi yang ditampilkan sudah benar dan sesuai maka peneliti manentukan beberapa ciri yang sesuai dengan ekspresi tersebut. Pada penelitian yang dilakukan oleh Tekalp (2013) dijelaskan deskripsi tesktual dari beberapa kondisi wajah ketika berekspresi. Berikut deskripsi tekstual ciri dari 3 ekspresi yang digunakan pada penelitian kali ini:

a. Ekspesi Netral, merupakan ekspresi dasar ketika kondisi wajah tidak mengalami emosi apapun.

- Seluruh otot wajah dalam kondisi rileks
- Kelopak mata bersinggungan dengan retina
- Bibir atas dan bawah saling bersentuhan
- Mulut tertutup

Dari deskripsi ciri tersebut kemudian peneliti meminta mahasiswa untuk berekspresi sesuai dengan ciri tersebut dan berikut hasilnya.

Gambar 3. 2 Ekspresi netral

b. Ekspesi Bahagia

- Posisi alis mata rileks
- Posisi mulut terbuka dan sedikit terlihat gigi atas dan bawah
- Ujung mulut tertarik ke arah telinga

Dari deskripsi ciri tersebut kemudian peneliti meminta mahasiswa untuk berekspresi sesuai dengan ciri tersebut dan berikut hasilnya

Gambar 3. 3 Ekspresi senyum

c. Ekspesi Marah

- Posisi alis mata bagian dalam tertarik kedalam dan hampir menyatu
- Mata terbuka lebar
- Birbir atas dan bawah saling mengerut dan menekan

Dari deskripsi ciri tersebut kemudian peneliti meminta mahasiswa untuk berekspresi sesuai dengan ciri tersebut dan berikut hasilnya.

Gambar 3. 4 Ekspresi marah

Berdasarkan deskripsi tekstual tersebut, peneliti nantinya akan meminta kepada mahasiswa untuk berekspresi sesuai dengan 3 kategori ekspresi tersebut dan kemudian difoto dari 10 sudut yang berbeda dimulai dari depan samping kanan/kiri dan geser ke arah sebaliknya. Tujuannya adalah untuk menadapatkan variasi gambar pada satu ekspresi.

Gambar 3. 5 Proses pengambilan foto

Dari 10 foto yang diambil tentu tidak semua dipilih untuk dijadikan data primer. Hasil foto dipilih mana yang paling terlihat bagus dan terlihat paling natural ekspresinya. Setelah diseleksi didapatkan 687 foto yang layak untuk digunakan yang terbagi menjadi 3 ekspresi.

Data berupa foto tersebut dibagi menjadi 2 bagian yaitu sebagai data *training* dan data *testing* dengan porsi kurang lebih 90%:10%. Dari komposisi porsi tersebut, kemudian *dataset* dipisah menggunakan kode berikut

```
train_generator = train_datagen.flow_from_directory(
        'E:/tes/data/baru_gray/train',
        target_size=(img_width, img_height),
        batch_size=batch_size,
        class_mode='categorical')
```

```
valid_generator = valid_datagen.flow_from_directory(
        'E:/tes/data/baru_gray/valid',
        target_size=(32, 32),
        batch_size=16,
        class_mode='categorical')
```

Gambar 3. 6 *Source code* pemisahan *dataset*

Kode diatas merupakan merupakan kode untuk memanggil dan kemudian memisahkan data sesuai dengan kelasnya. Ketika dijalankan maka sistem akan otomatis membaca semua gambar yang ada pada *path* yang sudah ditentukan, sekaligus membaca juga merubah setiap folder yang ada pada path tersebut menjadi kelas yang disusun dalam bentuk array class.

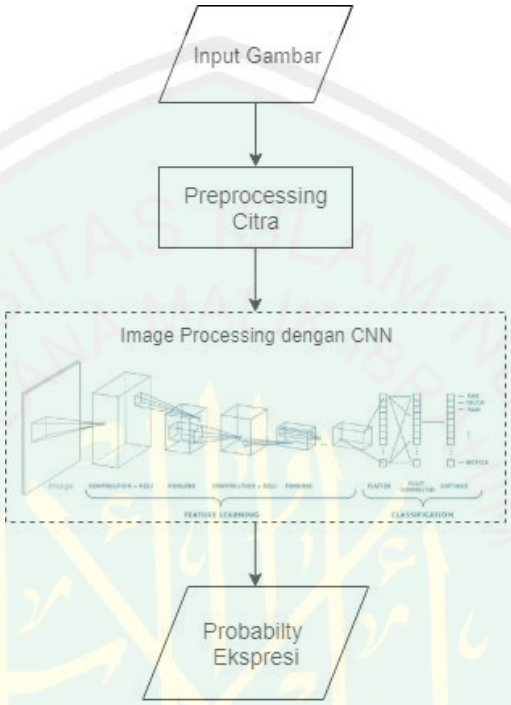
```
Found 570 images belonging to 3 classes.

570
{'marah': 0, 'normal': 1, 'senyum': 2}
```

Gambar 3. 7 *Dataset* sesuai kelas

3.3. Desain Sistem

Sebelum sistem pada penelitian ini dibangun, perlu adanya sebuah desain dari sistem yang akan dibangun terlebih dahulu. Desain sistem ini akan menjadi gambaran besar seperti apa jalannya sistem yang akan dibangun nantinya. Dari

desain sistem itu akan terlihat pada bagian mana algoritma yang CNN nanti akan diimplementasikan. Secara sederhana desain dari sistem identifikasi ekspresi wajah ini tergambar pada gambar dibawah ini.



Gambar 3. 8 Desain sistem

Secara sederhana pada penelitian ini, sistem yang akan dibangun nantinya terdiri dari 3 bagian utama, yaitu :

1. *Input* gambar
2. *Preprocessing* Citra
3. Implementasi CNN
4. *Output*

Seperti yang dijelaskan pada subab sebelumnya yaitu pengumpulan data, data yang sudah dikumpulkan akan menjadi masukan yang akan digunakan. *Dataset* tersebut nantinya akan diproses pada bagian *image processing* dengan algoritma CNN. Baru setelah pemrosesan selesai, akan menghasilkan *output* berupa hasil identifikasi ekspresi wajah. Dari 3 bagian utama sistem ini, pada masing-masing

bagian terdapat pemrosesan sendiri-sendiri. Berikut akan dijabarkan pemrosesan yang terjadi pada setiap bagian.

3.3.1. *Input* Gambar

Gambar objek yang sebelumnya sudah dikumpulkan, nantinya akan digunakan sebagai masukan pada sistem yang akan dibangun ini. Masukan gambar pada sistem ini dibagi menjadi 2 masukan, yang pertama masukan gambar sebagai data *training* dan masukan gambar sebagai data *testing*.

3.3.2. *Preprocessing* Citra

Merujuk pada penelitian yang dilakukan oleh Andre Lopes *et al.*, (2017) tentang identifikasi ekspresi dengan data yang sedikit, maka penelitian kali ini akan sedikit banyak merujuk kepada penelitian yang mereka lakukan. Dalam penelitian yang mereka lakukan, sebelum citra dijadikan masukan untuk pelatihan, citra diolah terlebih dahulu agar lebih memudahkan algoritma CNN untuk melakukan pelatihan dan menemukan ciri dari citra yang dimasukkan.

Ada 3 tahap *preprocessing* citra yang dilakukan sebelum citra tersebut diolah oleh algoritma CNN :

a. Cropping

Cropping citra dilakukan untuk mendapatkan lebih banyak objek tampak wajah dibandingkan objek selain wajah pada citra. Proses *cropping* menjadi hal yang cukup penting, karena jika tidak dilakukan *cropping*, maka semua objek yang ada pada citra akan dianalisis oleh komputer, padahal objek yang ingin diteliti adalah berupa ekspresi yang ada pada wajah saja. Sehingga, peneliti melakukan *cropping* pada bagian *background* dan objek selain wajah. Jadi hanya mengambil citra pada bagian wajah saja. Untuk proses *cropping*

ini masih dilakukan secara manual, karena ada beberapa ekspresi wajah yang gambarnya diambil dari posisi samping.

Gambar 3. 9 *Cropping image*

b. Grayscaled Image

Agar algoritma CNN nantinya lebih mudah dalam mendapatkan ciri dari citra yang sedang dilatih, maka citra training yang akan dijadikan masukan nantinya akan lakukan perbaikan pada fitur warna. Perbaikan pada ftur warna yang akan dilakukan adalah dengan mengubah citra RGB menjadi citra *grayscale*. Dengan begitu komputasi algoritma CNN yang dilakukan lebih sedikit dan ciri pada setiap gambar lebih mudah untuk didapatkan. Pada bahasa pemrograman python, untuk mengubah citra RGB menjadi citra *grayscale*, bisa menggunakan perintah `convert('L')` yang tersedia pada *library* `Image`.

```
1     from PIL import Image
2
3     for i in range(1,229):
4           t = i
5           img_name = 'file'+str(t)+'.JPG'
6           img = Image.open(img_name).convert('L')
7           citragray = img.save("gray"+str(t)+"jpg")
8       print (t)
```

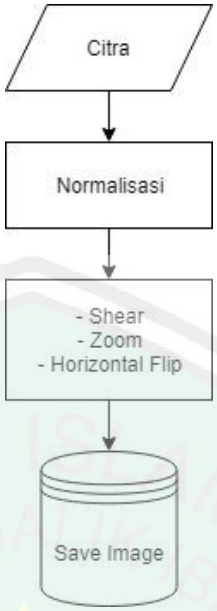
Gambar 3. 10 *Source code grayscale image*

Hasil yang didapatkan dari proses perbaikan citra pada fitur warna adalah berupa citra *grayscale* 8-bit. Seperti ini hasil yang didapatkan dari perubahan warna tersebut

Gambar 3. 11 Citra *grayscale*

c. Augmentasi Citra

Setalah citra dirubah menjadi citra *grayscale*, kemudian dilakukan proses data augmentasi. Seperti yang diketahui, untuk mendapatkan performa optimal, *deep learning* membutuhkan data dengan jumlah yang banyak. Karena variasi objek yang ada di lokasi penelitian dan kemampuan pengambilan data cukup terbatas, maka dilakukan data augmentasi untuk memperbanyak variasi data. Data augmentasi adalah sebuah teknik memanipulasi sebuah data tanpa kehilangan inti atau esensi data data tersebut. Augmentasi yang dilakukan *rescale*, *rotate*, *zoom* dan *flip*.



Gambar 3. 12 Augmentasi gambar

Augmentasi citra pada python bisa dilakukan dengan menggunakan *library* `ImageDataGenerator` yang sudah disediakan oleh keras. Setelah melakukan import package, fungsi baru bisa `ImageDataGenerator` dipanggil, kemudian menentukan augmentasi apa saja yang akan diaplikasikan ke gambar.

```
1    train_datagen = ImageDataGenerator(
2          rescale=1. / 255,
3          shear_range=0.2,
4          zoom_range=0.2,
5          rotation=30,
6          horizontal_flip=True
```

Gambar 3. 13 *Source code* augmentasi citra

Kode diatas merupakan kode untuk memanggil fungsi yang digunakan untuk melakukan augmentasi citra. Augmentasi citra yang dilakukan pada gambar adalah berupa:

a. *Rescale* Citra: fungsi dari `rescale=1. / 255` adalah untuk menormalisasi citra yaitu dengan membagi piksel terkecil dengen piksel terbesar. Fungsi

ini akan dikenakan pada semua gambar sebelum melakukan augmentasi dengan fungsi lainnya.

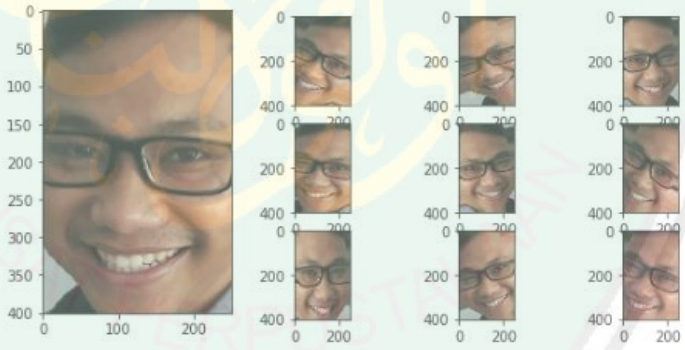
b. *Shear*: fungsi dari `shear_range=0.2` adalah untuk menggeser citra searah jarum jam dengan pergeseran sebanyak 0.2 derajat.

c. *Zoom*: fungsi dari `zoom_range=0.2` adalah untuk memperbesar citra dengan perbesaran sebanyak 1+0.2 dari luas gambar.

d. *Rotation*: fungsi dari `rotation=30` adalah untuk memutar gambar dengan sudut 30 derajat secara acak.

e. *Flip*: fungsi dari `horizontal_flip=True` adalah untuk membalik gambar secara horizontal.

Semua fungsi augmentasi citra tersebut akan dikombinasikan dan diaplikasikan secara acak kepada semua citra.



Gambar 3. 14 Hasil augmentasi citra

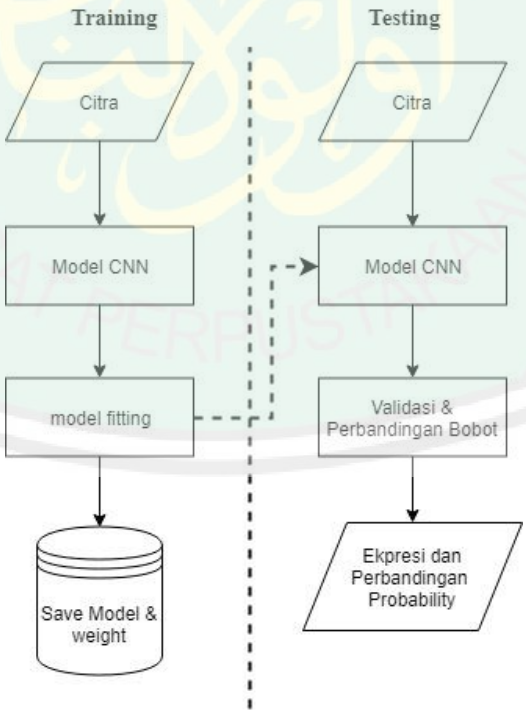
3.3.3. Implementasi CNN

Sebelum algoritma CNN diimplementasikan untuk mengidentifikasi ekspresi, algoritma CNN perlu dilatih terlebih dahulu untuk menemukan ciri yang bisa dikenali pada gambar tersebut dan mengaktifkan *neuron* untuk klasifikasi. Pada tahap *training* pun, pengolahan gambar dibagi menjadi 2 model yaitu, model

training dan model *testing* Setelah model dilatih, arsitektur CNN tersebut nantinya disimpan untuk kemudian dipanggil pada tahap *testing*.

3.3.3.1. Proses *Training*

Dalam sistem identifikasi ekspresi wajah ini, untuk mendapatkan akurasi pengenalan objek yang tinggi maka algoritma perlu dilatih terlebih dahulu dengan sejumlah data *training*. Tujuan dari dari melatih algoritma ini adalah untuk menemukan ciri dari setiap gambar kemudian menandai *neuron-neuron* mana yang akan diaktifkan ketika gambar diklasifikasi. Oleh karena itu, perlu dibuat skema atau model untuk melakukan pelatihan pada algoritma agar ketika dilakukan pengujian *object recognition*, algoritma sudah terlatih. Berikut skema pembentukan model yang akan dilakukan untuk mendapatkan pola menggunakan algoritma CNN.



Gambar 3. 15 Skema *training dan testing*

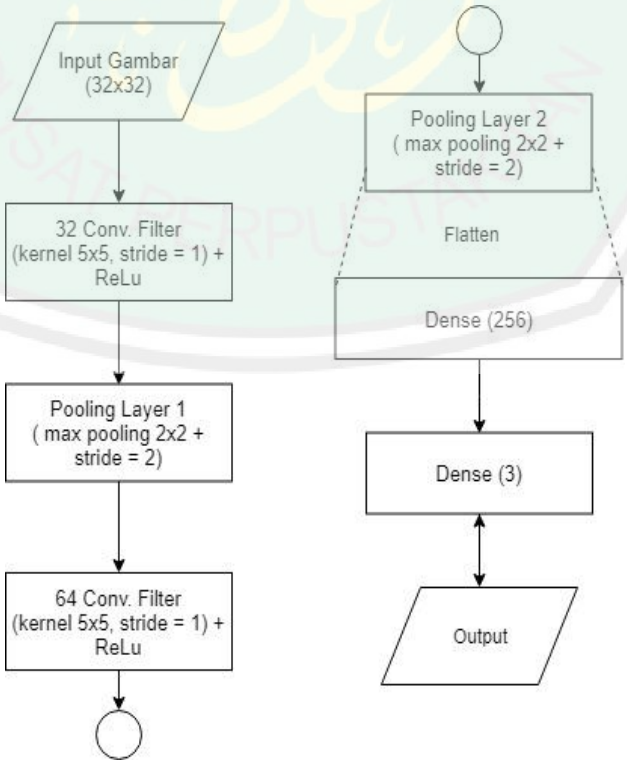
Sebelum melakukan pemrosesan gambar menggunakan algortima CNN, data *training* yang dibutuhkan harus dipanggil terlebih dahulu. Data *training* yang

dipanggil ini merupakan data yang sudah dilakukan augmentasi data untuk memperluas data *training*. Sebalum itu, perlu adanya inisialisasi beberapa parameter *learning*. Parameter yang perlu diinisialisasi untuk proses *training* ini adalah *learning rate*, *batch size* dan *epoch*

Tabel 3. 1 Inisialisasi parameter

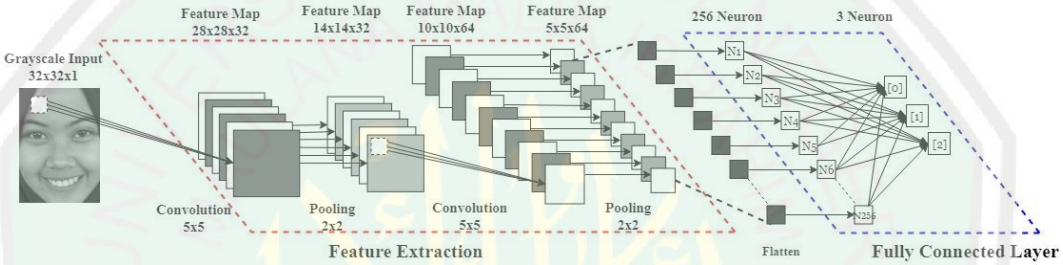
Parameter	Fungsi
Learning rate	Parameter gradient descent yang digunakan untuk meng*update* bobot pada parameter yang akan *training* pada setiap kali iterasi dilakukan
Batch size	Jumlah sampel data yang akan disebarkan ke *neural network* dalam satu kali *epoch*
Epoch	Banyaknya putaran yang dilakukan mulai dari awal *dataset* pertama hingga akhir

Setelah parameter yang dibutuhkan sudah diinisialisasi kemudian algoritma CNN diimplementasikan untuk melakukan *feature learning* dan klasifikasi gambar. Tapi sebelum itu, arsitektur dari algoritma CNN ini perlu dirancang terlebih dahulu. Berikut rancangan arsitektur CNN pada penelitian ini :



Gambar 3. 16 Arsitektur CNN

Blok diagram arsitektur diatas merupakan model algoritma CNN yang dikembangkan oleh Andre Lopes *et al*., (2017). Arsitektur tersebut merupakan arsitektur pengembangan dari arsitektur yang sudah pernah dibuat sebelumnya oleh Liu Kuang *et al*., (2016). tersebut. Perbedaannya adalah Andre Lopes *et al*., (2017) ini berusaha untuk membentuk model CNN yang bagus dengan jumlah data yang lebih sedikit. Berdasarkan model pada blok diagram diatas, arsitektur CNN dapat disimulasikan seperti pada gambar berikut :



Gambar 3. 17 Simulasi CNN model

Agar lebih mudah memahami arsitektur CNN pada blok diagram diatas, bisa dijelaskan seperti ini:

1. Sebelum gambar di-*input*-kan, gambar akan diaugmentasi seperti yang sudah dijelaskan pada bagian *input* gambar diatas. Hasil augmentasi tersebut akan disimpan dalam bentuk variabel.
2. Variabel yang berisi hasil augmentasi citra kemudian dipanggil untuk dijadikan *input* pada model CNN ini. *Input* citra yang akan digunakan berukuran 32x32x1. Angka 1 pada ukuran citra tersebut menandakan *channel image* yang merupakan citra *grayscale*.
3. Setelah itu citra yang sudah di*input*kan tersebut masuk pada proses konvolusi pertama. Pada konvolusi yang pertama, gambar akan dikalikan dengan *kernel* berukuran 5x5 dengan *filter* sebanyak 32 *filter*. Proses perkalian dilakukan dilakukan dengan menggeser *kernel* sebanyak 1 *stride*.

Dari proses konvolusi pertama ini *feature map* yang dihasilkan adalah 28x28x32. Setelah itu *feature map* yang dihasilkan akan akan dilakukan aktivasi fungsi menggunakan ReLu.

4. Proses *pooling*, *pooling* merupakan proses subsampling pada *feature map* hasil konvolusi. Pada dasarnya *pooling* ini adalah melakukan pengurangan ukuran *feature map* dengan menggunakan *kernel* ukuran tertentu yang akan secara digeser pada seluruh area *feature map*. Pada penelitian ini menggunakan max-*pooling* dengan ukuran *kernel* 2x2 dan *stride* sebesar 2 piksel.
5. Setelah dilakukan *pooling*, *feature map* hasil *pooling* akan dijadikan sebgai *input*an kembali untuk proses konvolusi kedua. Pada proses kovolusi kedua ukuran *kernel* yang digunakan masih sama yaitu 5x5, tetapi *filter* yang digunakan bertambah menjadi 64 *filter*. Penambahan jumlah *filter* ini dilakukan karena pada proses *pooling* informasi yang dibuang semakin banyak, oleh karena itu penambahan *filter* dilakukan agar variasi informasi yang diperoleh dari informasi yang tersedia semakin banyak. Pada proses konvolusi kedua ini akan dihasilakn *feature map* berukuran 10x10x64. Sama seperti sebelumnya, proses konvolusi kedua ini juga menggunakan fungsi aktivasi ReLu.
6. Proses selanjutnya masuk ke proses *pooling* yang kedua, proses ini hampir sama dengan proses *pooling* yang pertama, yaitu dengan menggunakan *kernel* berukuran 2x2 dengan *stride* 2 piksel. Dari proses *pooling* yang kedua ini dihasilkan *feature map* berukuran 5x5x64.

7. Sebelum masuk pada proses *fully connected layer*, *feature map* yang berupa 3D array akan dirubah menjadi vector 1-D *list* terlebih dahulu, proses ini sering dikenal *flatten*. Dari *feature map* berukuran 5x5x64 maka akan didapatkan nilai vector sebesar 1,250 piksel. Hasil ini akan dijadikan sebagai *input*an pada proses *fully connected layer*.
8. Setelah tahap flatten, maka akan diteruskan ke jaringan *fully connected layer* (FC *Layer*). Pada jaringan FC *Layer* ini menggunakan jaringan MLP dengan 4 *layer* yang terdiri dari *input layer*, outpur *layer* dan 2 hidden *layer*. Pada *input layer* terdiri dari 1 *neuron* hasil flatten dan pada *output layer* terdiri dari 10 *neuron*. Karena *dataset* nantinya terdiri dari 10 kelas, maka 10 *neuron* ini merupakan jumlah klasifikasi citra untuk setiap kelasnya. Sedangkan pada 2 jaringan hidden *layer* memiliki jumlah *neuron* yang berbeda, hidden *layer* pertama terdiri dari 120 *neuron*. Kemudian pada hidden *layer* kedua terdiri dari 84 *neuron*.

Proses pembentukan arsitektur CNN bisa dilakukan dilakukan dengan mengimport library yang sudah disediakan yaitu `keras`. Kemudian mendefinisikan *layer-layer* yang diperlukan untuk membentuk arsitektur CNN.

```
#model
model = Sequential()

#CNN Layer
model.add(Conv2D(filter s=32, kernel_size=5,
      strides=(1,1),
      padding ='same', input_shape=(img_width,
      img_height, 3), 7          activation='relu'))
model.add(BatchNormalization())

#Maxpooling
model.add(MaxPooling2D(pool_size=2))

model.add(Conv2D(filter s=64, kernel_size=5,
      strides=(1,1), padding ='same',
      activation='relu'))
model.add(BatchNormalization())

model.add(MaxPooling2D(pool_size=2))
model.add(Dropout(0.5))

#Flatten
model.add(Flatten())

#FC Layer
model.add(Dense(256, activation='relu'))

model.add(Dropout(0.5))
model.add(Dense(3, activation='softmax'))
```

Gambar 3. 18 *Source code* model CNN

Agar model diatas berjalan sesuai dengan parameter learning yang ditentukan, maka model CNN yang sudah disusun tersebut perlu untuk dikonfigunarsi menggunakan *method* `model.compile()`.

```
1     model.compile(optimizer=sgd(lr=0.001, momentum=0.9,
2           decay=0.0), loss='categorical_crossentropy',
3           metrics=['accuracy'])
```

Gambar 3. 19 *Source code compile* model

Kode diatas merupakan kode untuk menyusun algoritma yang sudah dibuat agara melakukan *training* sesuai dengan parameter *learning* yang diberikan. Parameter *learning* yang diinisialisasi pada kode diatas adalah parameter *learning rate* dengan optimizer *Stochastic Gradient Descent* (SGD). SGD adalah metode optimisasi *machine learning* yang berguna untuk melakukan *update* bobot pada setiap parameter yang di *training*. Karena banyaknya parameter yang harus di *training* pada jaringan *FC Layer*, maka optimizer SGD dirasa cocok untuk digunakan. Selain itu merujuk pada penelitian yang dilakukan oleh Ashia Wil (Ashia C. Wilson, 2018)son *et al.*, (2018) ketika melakukan komparasi antara beberapa metode optimisasi *machine learning*, menunjukkan bahwa SGD memiliki performa paling bagus diantara optimisasi lainnya.

Dari arsitektur CNN diatas maka *dataset* akan dilakukan proses *training* sesuai dengan *training* model yang sudah dibuat sebelumnya. Proses untuk melakukan pelatihan model ini sering disebut dengan *model fitting*. Proses *model fitting* ini bisa dilakukan dengan cara memanggil method `model.fit()`, kemudian memasukkan variabel data dan parameter iterasi yang harus dijalankan selama proses model *fitting* tersebut.

```
modelfit = model.fit_generator(train_generator,
      steps_per_epoch=nb_train_samples // 16,
      epochs=1000,
      validation_data=validation_generator,
      validation_steps=nb_validation_samples // 16,
      callbacks=[callbacks])
```

Gambar 3. 20 *Source code* model *fitting*

Ketika model *fitting* ini dijalankan, maka arsitektur CNN yang sudah dibentuk akan langsung bekerja untuk mengeksekusi data yang sudah dipersiapkan sebelumnya. Proses pelatihan dilakukan sejumlah *epoch* yang sudah ditentukan. Setiap menjalankan iterasi, sistem akan langsung otomatis menampilkan *accuracy value* dan *loss value* dari data *training*. Setelah proses training selesai, model yang terbentuk dari proses *training* ini kemudian disimpan yang kemudian dipanggil saat proses *testing*.

3.3.3.2. Proses *Testing*

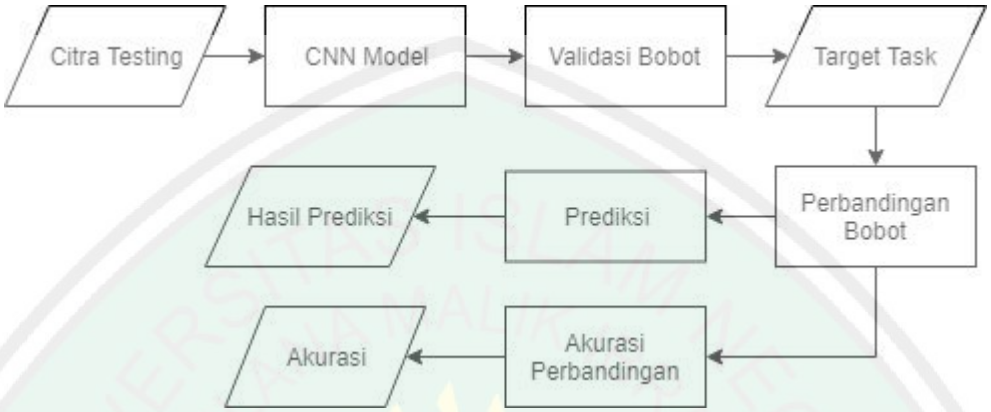
Setelah model CNN melalui prose *training*, model tersebut akan di *testing* untuk menguji seberapa baik kinerja dari model yang sudah dilatih sebelumnya. Proses pengujian ini dilakukan dengan cara mengujikan image baru yang belum pernah dilatih sebelumnya oleh model tersebut. Dengan begitu, akan terlihat seberapa baik kinerja model yang sudah dibuat untuk mengidentifikasi. Step pertama yang dilakukan adalah dengan memanggil model yang sudah dibentuk sebelumnya pada saat pelatihan.

```
model_path = "../training folder/new_dataset2.h5"
model = load_model(model_path)
```

Gambar 3. 21 Source code pemanggilan model

Proses selanjutnya adalah memunculkan hasil prediksi dari image yang diujikan. Untuk memunculkan hasil prediksi, sistem akan membandingkan bobot

antara *image* yang akan diprediksi dengan bobot *output* pada model yang sudah dipanggil tersebut. Setelah bobot *image* tervalidasi, maka *image* tersebut akan diklasifikasikan ke *output* dengan bobot terdekat.



Gambar 3. 22 *Output* klasifikasi

Hasil dari pengujian ini nantinya berupa hasil klasifikasi dan nilai *probability* dari 2 bobot *output* yang terdekat dengan image yang diuji. Source code untuk proses pengujian seperti pada gambar 3.23

```
1     def convert_to_array(img):
2           im = cv.imread(img)
3           img = Image.fromarray(im, 'RGB')
4           image = img.resize((32, 32))
5           return np.array(image)
6
7     def predict_label(file):
8           ar1=convert_to_array(file)
9           ar=ar1/255
10          a=[]
11          a.append(ar)
12          a=np.array(a)
13          label_dict = {0: 'Marah', 1: 'Netral', 2:
14                'Senyum'}
15          score=model.predict(a)
16          ekspresi_top = score[0].argpartition(-2)
17                [-2:][::-1]
18          label_index=np.argmax(score[0]
19                [ekspresi_top[0]])
20          label_index_2=np.argmax(score[0]
21                [ekspresi_top[1]])
22          percent_high = np.around(100*score[0]
23                [ekspresi_top[0]],decimals=2)
24          percent_secondhigh = np.around(100*score[0]
25                [ekspresi_top[1]],decimals=2)
```

Gambar 3. 23 *Source code* membandingkan bobot

BAB IV

HASIL DAN PEMBAHASAN

4.1. Langkah-langkah Uji Coba

Langkah-langkah uji coba untuk mendeteksi ekspresi berbasis citra menggunakan algoritma CNN dapat dilihat sebagai berikut:

a. Proses *training*, dimana tahap ini akan dilakukan pelatihan pada data yang sudah disiapkan sebelumnya menggunakan model CNN. Model yang sudah terlatih nantinya, kemudian digunakan untuk mengukur seberapa bagus algoritma dalam melakukan identifikasi
b. Proses *testing*, tahap ini merupakan tahap yang cukup penting karena pada tahap ini program yang sudah dibangun dan di *training* sebelumnya dilakukan pengujian. Data yang sudah disiapkan untuk di uji kemudian diujikan menggunakan model yang sudah dilatih sebelumnya.
c. Pengembangan parameter *learning*, pada tahap ini akan ditampilkan perbandingan akurasi yang didapatkan dengan merubah beberapa parameter *learning* seperti *epoch*, *learning* rate dan ukuran citra.

4.2. Hasil Uji Coba

Berikut hasil uji coba pada sistem identifikasi ekspresi wajah berbasis citra menggunakan algoritma CNN.

4.2.2. Hasil Proses *Training*

Salah satu bagian terpenting dari berhasilnya proses identifikasi ekspresi berbasis citra ini adalah bagusnya hasil dari proses *training* ini. Baiknya hasil dari

proses *training* akan memberikan dampak yang sangat tinggi terhadap hasil yang didapatkan pada proses uji coba nantinya.

Setelah arsitektur dibentuk dan dilakukan proses model *fitting*, maka algoritma akan langsung melakukan pelatihan pada data yang sudah disiapkan sebelumnya. Seperti yang sudah dibahas pada sub bab sebelumnya, data yang akan digunakan untuk proses *training* adalah sebesar 90% data. Data tersebut akan dieksekusi oleh algoritma CNN untuk diekstraksi yang dipelajari fiturnya.

Parameter iterasi yang dilakukan pada proses *training* ini sebanyak 1000 *epoch* dengan nilai *batch size* sebesar 16. Jadi proses pelatihan akan berlangsung dan diulang-ulang sebanyak 1000 kali untuk memperolah ekstrasi ciri dari fitur yang dibutuhkan. Kemudian untuk nilai *learning* rate yang digunakan pada proses *training* ini adalah 0.001. Nilai *learning* rate ini digunakan untuk melakukan *update* bobot setiap kali algoritma melakukan proses *backward-pass*.

Setelah proses *training* ini selesai, kemudian dilakukan perhitungan akurasi untuk mendapatkan nilai akurasi dari hasil *training* ini. Dalam menghitung akurasi, dilakukan dengan memasukkan hasil prediksi pada tabel hasil prediski seperti pada tabel dibawah ini.

Tabel 4. 1 Hasil prediksi proses *training*

Matriks		*Predict Class*		
		Marah	Netral	Senyum
Actual Class	Marah	205	0	0
	Netral	1	204	0
	Senyum	0	1	204

Dari tabel diatas bisa diketahui bahwa akurasi dari proses *training* sebesar 99.6%. Nilai akurasi ini bisa dinyatakan pada grafik hasil proses *training* setelah berjalan 1000 *epoch* menggunakan *tensorboard*.

Gambar 4. 1 Grafik akurasi dan *loss*

4.2.3. Hasil Proses *Testing*

Proses *testing* menggunakan data uji sebanyak 72, dan setiap kelas jenis ekspresi sebanyak 24 citra. Proses untuk mendapatkan akurasi dilakukan dengan 2 cara, yaitu menggunakan *method* `model.evaluate_generator()` dan dengan menggunakan perhitungan *confusion matrix*.

Proses penghitungan akurasi bisa dilakukan langsung menggunakan *method* `model.evaluate_generator()` dengan cara memanggil *directory* dari folder validasi, kemudian dieksekusi dengan *method* tersebut.

```
# menentukan masukan tiap data testing
steps = nb_validation_samples/1

# proses penghitungan akurasi
evaluate= model.evaluate_generator
      (validation_generator, steps)

# menampilkan akurasi
print('Test accuracy =
      {:.2f}'.format(evaluate[1]*100)+'%')
```

Gambar 4. 2 *Source code* perhitungan akurasi

Hasil dari proses penghitungan akurasi menggunakan *method* `model.evaluate_generator()` ini, didapatkan nilai akurasi sebesar 85,77%

dengan nilai *loss* sebesar 0,48. Dari proses penghitungan ini bisa diketahui bahwa hasil dari proses *testing* ini, lebih dari 80% data yang diuji coba hasilnya benar.

Sebagai penguat dari hasil penghitungan akurasi menggunakan *method*, proses *testing* dilakukan dengan menguji semua data *testing* satu persatu. Hasil dari pengujian yang dilakukan bisa dilihat pada tabel dibawah ini. Dibawah ini merupakan hasil identifikasi ekspresi marah.

Tabel 4. 2 Hasil uji coba identifikasi ekspresi marah

No	Input	Output	Status Identifikasi
1	marah1.JPG	Marah	Benar
2	marah2.JPG	Marah	Benar
3	marah3.JPG	Marah	Benar
4	marah4.JPG	Marah	Benar
5	marah5.JPG	Marah	Benar
6	marah6.JPG	Marah	Benar
7	marah7.JPG	Marah	Benar
8	marah8.JPG	Marah	Benar
9	marah9.JPG	Marah	Benar
10	marah10.JPG	Marah	Benar
11	**marah11.JPG**	**Netral**	**Salah**
12	marah12.JPG	Marah	Benar
13	marah13.JPG	Marah	Benar
14	marah14.JPG	Marah	Benar
15	marah15.JPG	Marah	Benar
16	marah16.JPG	Marah	Benar
17	**marah17.JPG**	**Netral**	**Salah**
18	marah18.JPG	Marah	Benar
19	marah19.JPG	Marah	Benar
20	marah20.JPEG	Marah	Benar
21	marah21.JPEG	Marah	Benar
22	marah22.JPEG	Marah	Benar
23	marah23.JPG	Marah	Benar
24	marah24.JPG	Marah	Benar

Kemudian dibawah ini adalah hasil uji coba identifikasi ekspresi netral.

Tabel 4. 3 Hasil uji coba identifikasi ekspresi netral

1	netral1.JPG	Netral	Benar
2	netral2.JPG	Netral	Benar
3	netral3.JPG	Netral	Salah
4	**netral4.JPG**	**Marah**	**Benar**
5	netral5.JPG	Netral	Benar
6	netral6.JPG	Netral	Benar
7	netral7.JPG	Netral	Benar
8	**netral8.JPG**	**Marah**	**Salah**
9	netral9.JPG	Netral	Benar
10	netral10.JPG	Netral	Benar
11	netral11.JPG	Netral	Benar
12	netral12.jpg	Netral	Benar
13	netral13.JPG	Netral	Benar
14	netral14.JPG	Netral	Benar
15	netral15.JPG	Netral	Benar
16	netral16.JPG	Netral	Benar
17	netral17.JPG	Netral	Benar
18	netral18.JPG	Netral	Benar
19	netral19.JPG	Netral	Benar
20	**netral20.JPG**	**Marah**	**Salah**
21	netral21.JPG	Netral	Benar
22	netral22.JPG	Netral	Benar
23	netral23.JPG	Netral	Benar
24	netral24.JPG	Netral	Benar

Kemudian dibawah ini adalah hasil uji coba identifikasi ekspresi senyum.

Tabel 4. 4 Hasil uji coba identifikasi ekspresi senyum

1	senyum1.JPG	Senyum	Benar
2	senyum2.JPG	Senyum	Benar
3	senyum3.JPG	Senyum	Benar
4	senyum4.JPG	Senyum	Benar
5	senyum5.JPG	Senyum	Benar
6	senyum6.JPG	Senyum	Benar
7	senyum7.JPG	Senyum	Benar
8	**senyum8.JPG**	**Netral**	**Salah**
9	senyum9.JPG	Senyum	Benar
10	senyum10.JPG	Senyum	Benar
11	senyum11.JPG	Senyum	Benar

12	senyum12.JPG	Senyum	Benar
13	senyum13.JPG	Senyum	Benar
14	senyum14.JPG	Senyum	Benar
15	senyum15.JPG	Senyum	Benar
16	senyum16.JPG	Senyum	Benar
17	senyum17.JPG	Netral	Benar
18	**senyum18.JPG**	**Marah**	**Salah**
19	**senyum19.JPG**	**Netral**	**Salah**
20	senyum20.JPEG	Senyum	Benar
21	senyum21.JPG	Senyum	Benar
22	senyum22.JPG	Senyum	Benar
23	senyum23.JPG	Senyum	Benar
24	senyum24.JPG	Senyum	Benar

Dari tabel-tabel diatas bisa diketahui bahwa ada 2 data image yang salah pada kategori marah, 3 data *image* yang salah pada kategori netral dan 3 data image yang salah pada kategori senyum. Dalam mengidentifikasi jenis ekspresi wajah yang terdapat pada image tersebut. Dari tabel diatas kemudian dimasukkan kedalam tabel prediksi yang lebih ringkas seperti dibawah ini.

Tabel 4. 5 Hasil prediksi proses *testing*

Matriks		*Predict Class*		
		Marah	Netral	Senyum
Actual Class	Marah	22	2	0
	Netral	3	21	0
	Senyum	0	3	21

Berdasarkan tabel hasil prediksi diatas, hasil prediksi dari model terhadap data *testing* menunjukan hasil yang baik. Pada kelas marah dari 24 data yang diuji hanya ada 2 data yang memiliki prediksi salah. Sedangkan pada kelas netral dan senyum ada 3 ekspresi yang memiliki prediksi yang salah. Dari tabel tersebut bisa dihitung akurasinya seperti berikut

$$Accuracy = \frac{All\ True\ Positive}{Total\ Number\ Testing\ Entries} x100\% \quad (4.1)$$

$$Accuracy = \frac{64}{72} x100\% = 88{,}89\% \quad (4.2)$$

Dari perhitungan tersebut didapatkan nilai akurasi identifikasi ekspresi menggunakan algoritma CNN ini sebesar 88,89%. Hasil yang didapatkan dari perhitungan akurasi menggunakan *confusion matrix* ternyata lebih tinggi daripada perhitungan menggunakan *method*.

4.2.4. Pengaruh Parameter *Learning* Terhadap Akurasi

Salah satu hal yang mempengaruhi tinggi rendahnya akurasi yang didapatkan dalam proses pembentukan model adalah parameter *learning*. Dalam menentukan model terbaik, salah satu harus dicari nilai terbaik parameter *learning* dalam model CNN. Parameter *learning* yang dimaksud adalah pengaruh jumlah *epoch*, pengaruh nilai *learning* rate dan pengaruh ukuran *input* citra. Tujuan dari penentuan parameter model ini ingin membandingkan model mana yang paling terbaik dengan memperhatikan nilai parameternya.

4.2.4.1. Pengaruh Jumlah *Epoch*

Dalam melakukan pembelajaran, ketika seluruh *dataset* sudah melalui proses *training* pada Neural Network sampai dikembalikan ke awal dalam satu putaran itulah yang disebut dengan *epoch*. Hingga saat ini belum ada penelitian yang melakukan klaim tentang jumlah *epoch* yang paling bagus dalam melakukan proses *training*. Namun jika merujuk pada pada penelitian yang dilakukan oleh Andre Lopes *et al*., (2017), mereka melakukan percobaan antara 1000 hinggan 2000 *epoch*. Tapi disini peneliti mencoba untuk melakukan variasi percobaan lain seperti berikut ini.

Tabel 4. 6 Pengaruh epoch

Epoch	Akurasi	*Loss value*	Waktu
500	81.2%	0.53	1 jam 20 menit

1000	85.8%	0.48	2 jam 41 menit
1500	83.8%	0.8	3 jam 15 menit

Berdasarkan tabel diatas dengan menggunakan nilai *learning rate* 0.001 dan dimensi 32x32 didapatkan akurasi yang paling bagus ada pada *epoch* 1000 yaitu 85.8%. Jika dilihat dari tabel tersebut, ketika *epoch* ditambah 500, akurasi malah menurun dan nilai *loss* juga bertambah.

4.2.4.2. Pengaruh Jumlah Nilai *Learning Rate*

Dalam melakukan pembelajaran dan *update* bobot pada *neuron* di *fully connected layer*, dibutuhkan metode optimasi seperti *Stochastic Gradient Descent* (SGD). Dengan menggunakan metode ini, proses *update* bobot bisa lebih optimal dengan adanya parameter *learning* seperti nilai *learning rate*. Pada penelitian kali ini, salah satu yang menjadi analisis oleh peneliti adalah pengaruh nilai *learning rate* terhadap akurasi yang didapatkan.

Tabel 4. 7 Pengaruh nilai *learning rate*

Learning Rate	Akurasi	*Loss value*	Waktu
0.01	33.2%	11.8	2 jam 45 menit
0.001	85.8%	0.48	2 jam 41 menit
0.0001	84.2%	0.5	4 jam 12 menit

Jika dilihat dari tabel analisis diatas, dengan jumlah *epoch* 1000 dan dimensi 32x32 perubahan nilai *learning* juga cukup mempengaruhi nilai akurasi dan *loss value*. Sedangkan untuk waktu eksekusi dengan jumlah *epoch* yang sama, perubahan waktu tidak terlalu siknifikan. Berdasarkan tabel tersebut bisa diketahui bahwa nilai *learning* rate yang paling optimal adalah 0.001 dengan nilai akurasi 85.8% dan value *loss* sebesar 0.48.

4.2.4.3. Pengaruh Ukuran Citra

Salah satu parameter yang juga cukup mempengaruhi nilai akurasi adalah ukuran dari citra yang dijadikan masukan pada saat proses *training*. Karena semakin besar ukuran, otomatis semakin banyak nilai piksel yang perlu dilatih, hal itu membuat komputasi juga semakin meningkat. Selain itu, meningkatnya komputasi yang harus dilakukan membuat waktu pemrosesan juga semakin lama. Hal ini juga meningkatka resiko terbentuknya model yang *overfitting*. Berikut pengaruh ukuran citra terhadap akurasi.

Tabel 4. 8 Pengaruh ukuran citra

Ukuran Citra	Akurasi	*Loss value*	Waktu
32 x 32	85.6%	0.54	2 jam 45 menit
64 x 64	83.8%	0.48	3 jam 15 menit
128 x 128	81.2%	0.5	4 jam 25 menit

Proses pelatihan diatas dijalankan dengan nilai *epoch* 1000 dan nilai *learning* rate 0.001. Dari percobaan tersebut dapat dilihat nilai akurasi malah menurun ketika meningkatkan ukuran dari *input*-an. Selain itu, dengan meningkatkan ukuran citra juga membuat waktu pelatihan juga semakin lama. Dari skenario pelatihan yang dijalankan ini bisa dilihat bahwa ukuran yang sesuai dengan arsitektur yang dibuat ada citra dengan ukuran 32 x 32.

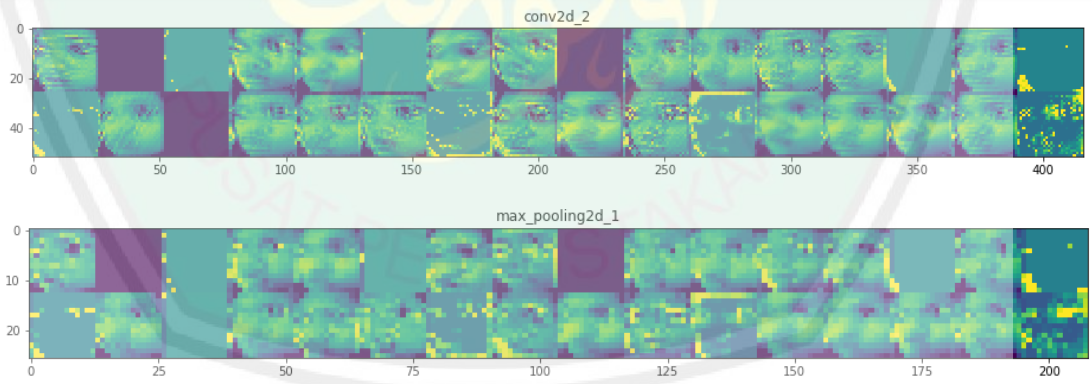
4.3. Pembahasan

Algoritma *Convolutional Neural Network* (CNN) merupakan salah satu algoritma *deep learning* yang fokus untuk mempelajari ciri dari suatu objek secara lebih mendalam. Salah satu yang menjadi bagian terpenting dari berhasilnya algoritma ini untuk melakukan klasifikasi terhadap suatu objek adalah banyaknya jumlah data. Semakin banyak jumlah data yang dilatih pada proses *training*, maka hasil yang didapatkannyapun akan semakin tinggi. Begitupun sebaliknya, algoritma

CNN ini tidak akan optimal ketika data yang dilatih sedikit. Oleh karena itu, perlu adanya operasi ekstraksi tambahan untuk meningkatkan hasilnya.

Pada penelitian ini algoritma CNN berhasil untuk melakukan identifikasi ekspresi pada wajah dengan akurasi 88,89%. Jika melihat dari penelitian yang dijadikan rujukan yaitu penelitian yang dilakukan oleh Andre Lopes *et al*., (2017). Dalam mengidentifikasi ekspresi, hasil yang didapatkan pada penelitian kali ini dinlai masih kurang optimal dan seharusnya bisa ditingkatkan lagi. Peneliti menganalisa sebab kurang optimalnya akurasi yang didapatkan ini adalah karena jumlah yang terlalu sedikit dan kualitas data yang kurang bagus.

Sebenarnya algoritma CNN ini merupakan algoritma yang sudah cukup bagus melakukan identifikasi pada ekspresi wajah. Ditambah lagi dengan kemandirian algoritma yang sebenarnya sudah tidak membutuhkan ekstraksi fitur lagi, karena didalamnya sudah ada proses *feature learning*.



Gambar 4. 3 Hasil *feature learning* tahap 1

Meskipun bagitu, karena tingkat komputasinya yang terlalu tinggi membuat algoritma *overfitting* sehingga akurasi yang didapatkan malah tidak setinggi yang diharapkan. Oleh karena itulah peneliti menambahkan ekstraksi fitur tambahan dengan merubah citra menjadi *grayscale* agar komputasinya lebih ringan.

Selain masalah diatas, masalah yang membuat hasil dari klasifikasi kali ini kurang maksimal adalah terbatasnya kemampuan perangkat yang peneliti gunakan. Hal itu membuat peneliti tidak mampu bereksperimen lebih untuk meningkatkan hasil dari penelitian kali ini.

Melihat dari pembahasan diatas bisa dilihat bahwa pentingnya untuk belajar dan mempelajari sesuatu secara mendalam. Ketika algoritma tidak mampu mempelajari suatu objek secara lebih dalam, maka hasil yang didapatkannyapun akan berkurang. Hal ini tidak beda dengan manusia, ketika kita tidak mau untuk belajar dan mempelajari sesuatu maka *output* dari diri kita pasti kurang baik.

Proses untuk selalu belajar dan mempelajari sesuatu ini selaras dengan wahyu pertama yang Allah turunkan kepada nabi Muhammad ﷺ yaitu QS al-'Alaq/96: 1-5

ٱقۡرَأۡ بِٱسۡمِ رَبِّكَ ٱلَّذِي خَلَقَ (١) خَلَقَ ٱلۡإِنسَٰنَ مِنۡ عَلَقٍ (٢) ٱقۡرَأۡ وَرَبُّكَ ٱلۡأَكۡرَمُ (٣) ٱلَّذِي عَلَّمَ بِٱلۡقَلَمِ (٤) عَلَّمَ ٱلۡإِنسَٰنَ مَا لَمۡ يَعۡلَمۡ (٥)

Artinya : *Bacalah dengan (menyebut) nama Tuhan Yang menciptakan, Dia telah menciptakan manusia dari segumpal darah. Bacalah, dan Tuhanmulah Yang Maha Pemurah, Yang mengajar (manusia) dengan perantaraan kalam. Dan mengajarkan kepada manusia apa yang tidak diketahuinya*.

Ayat di atas, mengandung pesan ontologis tentang belajar dan pembelajaran. Dalam hal ini, nabi Muhammad ﷺ, yang ummi (buta huruf aksara) beliau diperintahkan untuk tetap belajar. Hasil yang ditimbulkan dengan usaha belajar tersebut, dapat menghasilkan ilmu agama seperti fikih, tauhid, akhlak dan semacamnya yang Allah transferkan melalui lisan beliau. Sedangkan hasil yang ditimbulkan dengan usaha membaca ayat-ayat *kawniyah*, dapat menghasilkan sains

seperti fisika, biologi, kimia, astronomi, dan semacamnya. Dapat dirumuskan bahwa ilmu yang bersumber dari ayat-ayat *qur'aniyah* dan *kawniyah*, harus diperoleh melalui proses belajar membaca.

Timbul pertanyaan, mengapa kata *iqra'* atau perintah membaca sederatan ayat di atas terulang dua kali yakni pada ayat 1 dan 3. Jika merujuk pada kitab tafsir ibnu katsir rahimahullah, beliau berkata "Seseorang itu akan semakin mulia dengan ilmu diin yang ia miliki. Ilmu itulah yang membedakan bapak manusia, yaitu Adam dengan para malaikat. Ilmu ini terkadang di pikiran. Ilmu juga kadang di lisan. Ilmu juga terkadang di dalam tulisan tangan untuk menyalurkan apa yang dalam pikiran, lisan, maupun yang tergambarkan di pikiran."

Bisa disimpulkan bahwa kemampuan manusia untuk belajar dan mempelajari inilah yang membuat manusia semakin mulia disisi Allah SWT. Selain itu Allah juga mengisyaratkan bahwa Allah menciptakan tubuh kita ini agar dimanfaatkan untuk belajar. Hal ini juga diisyaratkan Allah pada QS An-Nahl (16): 78 berbicara tentang komponen pada diri manusia yang harus digunakan dalam kegiatan belajar dan pembelajaran:

وَٱللَّهُ أَخۡرَجَكُم مِّنۢ بُطُونِ أُمَّهَٰتِكُمۡ لَا تَعۡلَمُونَ شَيۡـٔٗا وَجَعَلَ لَكُمُ ٱلسَّمۡعَ وَٱلۡأَبۡصَٰرَ وَٱلۡأَفۡـِٔدَةَۙ لَعَلَّكُمۡ تَشۡكُرُونَ (٧٨)

Artinya: *Dan Allah mengeluarkan kamu dari perut ibumu dalam keadaan tidak mengetahui sesuatupun, dan Dia memberi kamu pendengaran, penglihatan dan hati, agar kamu bersyukur*

Ayat di atas mengisyaratkan adanya tiga komponen yang terlibat dalam teori pembelajaran, yaitu: al-sam'a, al-bashar dan al-fu'ad. Secara leksikal, kata al-sam'a berarti telinga yang fungsinya menangkap suara, memahami pembicaraan, dan selainnya. Penyebutan al-sama dalam Alquran seringkali dihubungkan dengan

penglihatan dan qalbu, yang menunjukkan adanya saling melengkapi antara berbagai alat itu dalam kegiatan belajar dan mempelajari sesuatu. Tidak sekedar akal pikiran saja, dalam mempelajari sesutu manusia juga perlu menggunakan berbagai komponen yang ada dalam tubuhnya agar mampu mempelajari sesuatu secara lebih mendalam.

Kemudian dilanjutkan lagi pada QS An-Nahl (16): 79 untuk mengamati kebesaran-kebesaran Allah SWT

أَلَمْ يَرَوْا۟ إِلَى ٱلطَّيْرِ مُسَخَّرَٰتٍ فِى جَوِّ ٱلسَّمَآءِ مَا يُمْسِكُهُنَّ إِلَّا ٱللَّهُ ۚ إِنَّ فِى ذَٰلِكَ لَءَايَٰتٍ لِّقَوْمٍ يُؤْمِنُونَ (٧٩)

Artinya : *Tidakkah mereka memperhatikan burung-burung yang dimudahkan terbang di angkasa bebas. Tidak ada yang menahannya selain daripada Allah. Sesungguhnya pada yang demikian itu benar-benar terdapat tanda-tanda (kebesaran Rabb) bagi orang-orang yang beriman.*

Setelah Allah mengisyaratkan bahwa Allah telah memberi pendengaran, penglihatan dan hati dengan suatu tujuan tertentu. Allah memerintahkan kita untuk memperhatikan berbagai hal disekitar kita. Dicontohkan dalam ayat tersebut adalah perintah untuk mengamati burung-burung yang terbang di angkasa. Secara eksplisit Allah memrintahkan kita untuk berfikir bahwa segala seuatu yang terjadi disekitar kita, bisa terjadi karena kekuasaan Allah SWT.

BAB V

PENUTUP

5.1. Kesimpulan

Dari hasil penelitian yang dikerjakan untuk mengidentifikasi ekspresi wajah berbasis citra menggunakan metode *Convolutional Neural Network* (CNN) ini, dapat ditarik kesimpulan sebagai berikut :

1. Pada dasarnya algoritma CNN merupakan algoritma yang mampu bisa melakukan klasifikasi pada objek tanpa membutuhkan tambahan ekstraksi fitur. Karena pada algoritma tersebut sudah ada tahap atau proses *feature learning*. Namun pada perjalanan penelitian ini, terlalu sedikitnya jumlah data yang dimiliki dan kurang bagusnya kualitas data yang didapatkan membuat peneliti menambahkan ekstraksi fitur tambahan agar lebih mempermudah algoritma dalam melakukan ekstraksi fitur pada image yang akan diproses. Dengan merubah citra RGB menjadi citra *grayscale*, peneliti berhasil mempertajam citra image untuk mempermudah proses ekstraksi.
2. Algoritma CNN sudah cukup baik dalam melakukan identifikasi ekspresi wajah berbasis citra ini. Dengan melakukan *training* pada 1000 *epoch* dan nilai *learning* rate 0.001 didapatkan akurasi *training* sebesar 99,6% dan akurasi *testing* menggunakan citra baru sebesar 88,89%. Hasil ini sudah cukup bagus melihat kualitas dan jumlah data yang didapatkan tidak begitu bagus dan jumlahnya yang kurang banyak.

5.2. Saran

Dalam pengembangan sistem identifikasi ekspresi wajah berbasis citra mengguanakan algoritma CNN ini diperlukan beberapa perbaikan untuk mencapai

hasil yang lebih maksimal, maka perlu dilakukan perbaikan pada beberapa hal diantaranya:

1. Mempebaiki kualitas foto ekpresi yang diambil agar terlihat lebih natural dan ekspresif. Selain itu memperbanyak jumlah data agar algortima mampu bekerja lebih maksimal
2. Menambah jenis ekspresi yang diidentifikasi agar kemampuan algoritma CNN lebih terukur.

DAFTAR PUSTAKA

Albani, Youssef, & Suriani. (2017). A Deep Learning Approach for Object Recognition with NAO Soccer Robots. *Sapienza University of Rome*.

Aliady, H., & Utari, D. T. (2018). GPU Based Image Classification using *Convolution*al Neural Network Chicken Dishes Classification. *Department of Statistics, Universitas Islam Indonesia*.

Andre Lopes, E. d. (2016). Facial Expression Recognition with Convolutional Neural Networks: Coping with Few Data and the Training Sample Order.

Ashia C. Wilson, ,. R. (2018). The Marginal Value of Adaptive Gradient Methods in Machine Learning. 31st Conference on Neural Information Processing System

Ciresan, D., Meier, U., Masci, J., Gambardella, L., & Schmidhuber. (2011). Flexible, High Performance *Convolution*al Neural Networks for Image Classification. *IDSIA Manno-Lugano, Switzerland*.

"Facial Recognition." World of Forensic Science. . Retrieved February 26, 2019 fromEncyclopedia.com: https://www.encyclopedia.com/science/encyclopedias-almanacs-transcripts-and-maps/facial-recognition/.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning (Adaptive Computation and Mechine Learning Series). *The IMT Press*.

Hermawan. (2006). *Jaringan Syaraf Tiruan dan Aplikasinya.* Yogyakarta: Andi.

Hubel, & Wiesel. (1961). Receptive Fields, Binocular Interaction And *Functio*nal Architecture In The Cat's Visual Cortex. *Harvard Aledical School*.

Janocha, K., & Czarnecki, W. M. (2017). On *Loss Function*s for Deep Neural Networks in Classification. *Jagiellonian University, Krakow, Poland*.

Joshi , A. (2017). Real Time Monitoring of CCTV Camera Images Using Object Detectors and Scene Classification for Retail and Surveillance Applications. *Computer Science, Stanford University*.

Kriesel, D. (2005). A Brief Introduction to Neural Networks. *drkriesel.com*.

Krizhevsky, A., Sutskever, I., & Hinton, G. (2012). ImageNet Classification with Deep *Convolution*al Neural Networks. *University of Toronto*.

Kuang Liu, M. Z. (2016). Facial Expression Recognition with CNN Ensemble. *2016 International Conference on Cyberworlds*, 163.

LeCun, Y., Bottou, L., Bengio, Y., & Haffiner, P. (1998). Gradient-Based Learning Applied to Document Recognition. *IEEE*.

Li, H., Wang, P., You, M., & Shen, C. (2018). Reading car license plates using deep neural networks. *Elsevier*.

Lisa. (2015). Deep Learning Tutorial. *University of Montreal*.

Mathworks. (2016). *Machine Learning*. Retrieved from https://www.mathworks.com/discovery/machine-*learning*.html

Miracle, Budi, Sutrisno (2018). Implementasi Metode Backpropagation untuk Prediksi Harga Batu Bara. Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer, 6502

Nam, H., & Han, B. (2016). Learning Multi-Domain *Convolution*al Neural Networks for Visual Tracking. *Computer Science and Engineering, POSTECH, Korea*.

Nurfita, R. D. (2018). Implementasi Deep Learning Berbasis Tensorflow Untuk Pengenalan Sidik Jari. *Universitas Muhammadiyah Surakarta*.

Nurhikmat, T. (2018). Implementasi Deep Learning Untuk Image Classification Menggunakan Algoritma *Convolution*al Neural Network (Cnn) Pada Citra Wayang Golek. *Universitas Islam Indonesia*.

Puspita , & Eunike. (2007). Penggunaan Jaringan Syaraf Tiruan Metode Backpropagation untuk Memprediksi Bibir Sumbing. *Seminar Nasional Teknologi*.

Puspitaningrum, d. (2006). *Pengantar Jaringan Syaraf Tiruan.* Yogyakarta: Andi.

Rosebrock, A. (2017). *Deep Learning for Computer Vision with Python.* Pyimagesearch.

Sena, s. (2017, November 13). *Convolutional Neural Network*. Retrieved from medium.com: https://medium.com/@samuelsena/pengenalan-deep-*learning*-part-7-*convolution*al-neural-network-cnn-b003b477dc94/

Simonyan, K., & Zisserman, A. (2015). Very Deep *Convolution*al Networks for Large-Scale Image Recognition. *University of Oxford*.

Singhal, N., Srishti, & Kalaichelvi. (2017). Application of *Convolution*al Neural Network to Classify Sitting and Standing Postures. *WCECS San Francisco, USA*.

Solar, J. R., Loncomilla, P., & Soto, N. (2018). A Survey on Deep Learning Methods for Robot Vision. *Electrical Engineering Universidad de Chile*.

Tekalp, O. (2013). Face and 2-D mesh animation in MPEG-4.

Xie, S. Q., & Hämmerle, E. (2008). Vision-Guided Robot Control for 3D Object Recognition and Manipulation . *The University of Auckland*, 28.

Yordanova, K. (2014). Methods for Engineering Symbolic Human Behaviour Models for Activity Recognition. (p. 171). Rostock: Faculty of Computer Science and Electrical Engineering University of Rostock.

Zufar, M., & Setiyono, B. (2016). *Convolutiona*l Neural Networks untuk Pengenalan Wajah Secara Real-Time. *Fakultas MIPA, Institut Teknologi Sepuluh Nopember (ITS)*.