DETEKSI PERGERAKAN ARAH MATA MENGGUNAKAN *CONVOLUTION NEURAL NETWORK* BERDASARKAN *FACIAL LANDMARK*

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Muhammad Amin Nurdin
NIM: 135150200111013

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2020

PENGESAHAN

DETEKSI PERGERAKAN ARAH MATA MENGGUNAKAN *CONVOLUTION NEURAL NETWORK* BERDASARKAN *FACIAL LANDMARK*

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Muhammad Amin Nurdin
NIM: 135150200111013

Skripsi ini telah diuji dan dinyatakan lulus pada
23 Juli 2020
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing 2
Randy Cahya Wihandika, S.ST., M.Kom.	Dr. Eng. Fitri Utaminingrum S.T., M.T.
NIK: 201405 880206 1 001	NIP: 19820710 200812 2 001

Mengetahui
Ketua Jurusan Teknik Informatika

Achmad Basuki, S.T., M.MG., Ph.D.
NIP: 19741118 200312 1 002

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar referensi.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 23 Juli 2020

Muhammad Amin Nurdin

NIM: 135150200111013

ABSTRAK

Muhammad Amin Nurdin, Deteksi Pergerakan Arah Mata Menggunakan *Convolution Neural Network* Berdasarkan *Facial Landmark*

Pembimbing: Randy Cahya Wihandika, S.ST., M.Kom. dan Dr. Eng. Fitri Utaminingrum S.T., M.T.

Pergerakan mata manusia dapat berguna di berbagai bidang, contohnya dalam sistem keamanan, kesehatan, transportasi dan desain antarmuka. Dalam desain sistem antarmuka, pergerakan mata digunakan sebagai sistem interaktif. Sistem dapat berinteraksi dan dapat memberikan tanggapan kepada pengguna dengan menggunakan gerakan mata. Metode *eye tracking* berbasis video memiliki keuntungan karena praktis dan nyaman selama proses deteksinya. Penelitian ini menggunakan algoritme *Convolution Neural Network* (CNN) karena akan memanfaatkan kelebihan dari metode CNN yaitu mampu mengklasifikasikan dan memiliki hasil paling signifikan dalam pengenalan objek. Hasil dari penelitian ini menunjukkan model CNN yang tepat digunakan dalam klasifikasi arah mata berdasarkan *facial landmark* adalah dengan 2 *layer* dengan 32 filter dan 64 filter, batch size 16 dalam augmentasi citra dengan 20 *fully connected layer* menghasilkan nilai loss 0,08, dengan akurasi 0,98 dan waktu pelatihan 8,62 detik. Hasil uji terhadap video yang diambil 50 *frame* secara acak sebanyak tiga kali, menghasilkan rata-rata akurasi sebesar 0,95.

Kata kunci: gerakan mata, facial landmark, convolution neural network.

ABSTRACT

Muhammad Amin Nurdin, *Eye Direction Movements Detection using Convolution Neural Network based on Facial Landmark.*

Supervisors: Randy Cahya Wihandika, S.ST., M.Kom. and Dr. Eng. Fitri Utaminingrum S.T., M.T.

The movement of the human eye can be useful in various fields, for example in security systems, health, transportation and design interface. In the design interface systems, eye movement used as an interactive system. The system can interact and responses to users by using eye movements. The video-based eye tracking method has the advantage of being practical and convenient during the detection process. This study uses the Convolution Neural Network (CNN) algorithm because it will utilize the advantages of the CNN method to classify and have the most significant results in object recognition. The results of this study indicate that the CNN model that good to use in the classification of eye direction based on facial landmarks is with 2 layers contain 32 filters and 64 filters, batch size 16 in image augmentation with 20 fully connected layers resulting loss value of 0.08, with an accuracy of 0.98 and 8.62 seconds in training time. Test results on videos taken 50 frames randomly three times, resulting in an average accuracy of 0.95.

Keywords: eye movement, facial landmark, convolution neural network

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Di masa sekarang perangkat komputer semakin mudah digunakan oleh berbagai macam pengguna. Berbagai penelitian di bidang *Human Computer Interaction* (HCI) memiliki peran dalam membuat desain dari suatu sistem yang membuat sebuah perangkat komputer dapat digunakan secara intuitif. Mata manusia merupakah salah satu objek yang menarik. Pandangan dari wajah menentukan garis penglihatan. Jadi, arah mata dapat mengekspresikan kepentingan pengguna dan tatapannya dapat digunakan untuk menafsirkan maksud dari pengguna sebagai media interaksi. Memasukkan gerakan mata ke dalam interaksi antara manusia dan komputer memberikan banyak manfaat potensial (Othera, et al., 2007). Pergerakan mata manusia juga berguna di berbagai bidang, contohnya dalam sistem keamanan, kesehatan, transportasi dan desain antarmuka. Dalam bidang keamanan, posisi mata dapat digunakan dalam membantu proses pengenalan jenis kelamin dan ekspresi wajah pengguna (Kroon, et al., 2009).

Pada bidang kesehatan, pergerakan mata digunakan untuk mendeteksi adanya suatu penyakit, contohnya adalah vertigo. Penyakit vertigo sering ditandai dengan terdapatnya pergerakan mata yang dilakukan secara tidak sadar (Pavlin-Premrl, et al,. 2015). Dalam bidang sistem transportasi cerdas, pergerakan dari mata dapat digunakan untuk mengukur tingkat perhatian dari pengemudi dan mengidentifikasi tingkat kantuk pada pengemudi (Edenborough, et al,. 2015). Sedangkan dalam desain sistem antarmuka, pergerakan mata digunakan sebagai sistem interaktif. Sistem dapat berinteraksi dan dapat memberikan tanggapan kepada pengguna dengan menggunakan gerakan mata (Duchowski, 2002).

Baru-baru ini teknik deteksi mata berbasis video telah banyak dipelajari. Dibandingkan dengan metode pelacakan tatapan berbasis *image*, metode *eye tracking* berbasis video memiliki keuntungan karena praktis dan nyaman selama proses deteksinya (Othera, et al., 2007). Kawato, et al. (2006) melakukan metode deteksi tatapan menggunakan empat titik referensi yang diletakkan di wajah dan tiga gambar kalibrasi. Matsumoto, et al. (2000) melakukan penelitian dengan menggunakan sistem *real-time stereo vision* untuk memperkirakan arah pandangan. Wang, et al., (2001) menggunakan metode untuk memperkirakan arah pandangan dengan mengukur perubahan kontur iris. Umumnya, deteksi berdasarkan pengukuran mata dengan akurasi tinggi biayanya mahal, karena memerlukan kamera *pan-tilt / zoom-in* dengan resolusi yang cukup tinggi untuk mengukur kontur iris atau pupil yang akurat(Wang,et al., 2001).

Metode yang menggunakan gerakan iris yang di-*capture* oleh kamera tunggal dengan resolusi rendah dengan kondisi pose kepala tertentu dilakukan oleh Hammal, et al. (2005) dan Benoit, et al. (2005). Metode ini menggunakan gradien luminance untuk mengekstrak lingkaran iris dan tepi mata. Oleh karena itu, perlu

untuk menentukan posisi kasar mata terlebih dahulu. Ekstraksi fitur iris sangat penting dalam algoritme yang mendeteksi pandangan dari gerakan iris. Ohtera, et al. (2006) mencoba menggunakan algoritme ekstraksi iris yang tepat berdasarkan transformasi *Hough*. Namun, metode tersebut tidak dapat mendeteksi pergerakan sub-piksel dan perubahan *Hough* memerlukan waktu operasi yang lama.

Metode lain yang banyak digunakan juga adalah metode *Convolution Neural Network* (CNN) yang termasuk salah satu algoritme dari Deep Learning. Jaringan ini dibuat dengan input berupa citra dan memiliki berbagai macam lapisan. Pada setiap lapisan ini, citra input akan menghasilkan pola-pola yang nantinya akan lebih mudah untuk diklasifikasikan. Menurut Suartika (2016), saat ini penggunaan CNN memberikan hasil yang baik dalam pengenalan citra karena CNN berusaha meniru sistem pengenalan citra pada manusia. Penelitian mengenai pengolahan citra yang lain dengan algoritme CNN menghasilkan nilai akurasi yang bagus, misalnya penelitian yang dilakukan oleh Abhirawa, (2017) tentang pengenalan wajah dengan menggunakan *Convolution Neural Network* memiliki tingkat akurasi 0,8973.

Berdasarkan uraikan diatas, peneliti mengusulkan penelitian yang berjudul "Deteksi Posisi dan Arah Mata Menggunakan Convolution Neural Network Berdasarkan Facial Landmark". Penelitian ini menggunakan algoritme *Convolution Neural Network* karena CNN mampu mengklasifikasikan dan memiliki hasil paling signifikan dalam pengenalan objek (Suartika, 2016).

Dalam penelitian ini, kami berkonsentrasi pada pendekatan berbasis video, dan mengembangkan sistem deteksi mata sederhana dengan hanya menggunakan sebuah kamera. Tujuan dari deteksi menggunakan *Convolution Neural Network* berdasarkan *Facial Landmark* adalah untuk mendeteksi dan mengevaluasi algoritme ini dalam mendeteksi arah pandangan mata agar dapat diketahui kelemahan sistem supaya dapat diperbaiki kemudian.

1.2 Rumusan Masalah

Berdasarkan uraian yang disampaikan dari latar belakang diatas, maka rumusan masalah dalam penelitian ini sebagai berikut:

1. Bagaimana membangun model *Convolution Neural Network* berdasarkan *Facial Landmark* dalam mengklasifikasikan arah mata pada citra?

1.3 Tujuan

Berdasarkan latar belakang dan rumusan masalah yang telah dijabarkan diatas, penelitian ini memiliki tujuan sebagaimana berikut

1. Membangun model *Convolution Neural Network* berdasarkan *Facial Landmark* untuk mengklasifikasikan arah mata pada citra menggunakan

1.4 Manfaat

Berdasarkan latar belakang, rumusan masalah dan tujuan yang telah dijabarkan, penelitian ini mempunyai manfaat sebagaimana berikut:

1. Memberikan referensi dalam pengembangan proses klasifikasi dalam bidang ilmu komputasi cerdas.

1.5 Batasan Masalah

Berdasarkan latar belakang dan rumusan masalah yang telah dijabarkan, supaya pembahasan yang dilakukan dalam penelitian ini dapat lebih fokus dan terarah, maka perlu ditetapkan beberapa batasan sebagai berikut

1. Pengambilan citra dilakukan di tempat yang memiliki cukup cahaya.
2. Hasil *capture* harus terdapat seluruh bagian wajah dan menghadap tegak lurus.
3. Jarak dari kamera antara 50-100 cm.

1.6 Sistematika Pembahasan

Sistematika dalam penulisan penelitian ini sebagai berikut:

BAB I Pendahuluan

Bab ini menjelaskan tentang latar belakang, rumusan masalah, tujuan, manfaat, batasan masalah dan sitematika penyusunan laporan.

BAB II Landasan Kepustakaan

Bab ini menjelaskan landasasan teori-teori yang berkaitan dengan penelitian. Bab ini juga menjelaskan tentang penelitian serupa yang sudah pernah dilakukan sebelumnya.

BAB III Metode Penelitian

Bab ini menjelaskan tentang uraian lengkap langkah-langkah dan tahapan penelitian demi mencapai tujuan penelitian.

BAB IV Perancangan

Bab ini membahas tentang perancangan dari sistem.

BAB V Implementasi

Bab ini menjelaskan tentang implementasi sistem yang terdiri dari penerapan metode yang digunakan.

BAB VI Pengujian dan Analisis

Bab ini membahas tentang hasi pengujian yang dilakukan apabila sistem sudah selesai dirancang dan diimplementasi. Pengujian tersebut dilakukan untuk melakukan uji coba terhadap sistem dan melakukan pembahasan terhadap hasilnya.

BAB VII Penutup

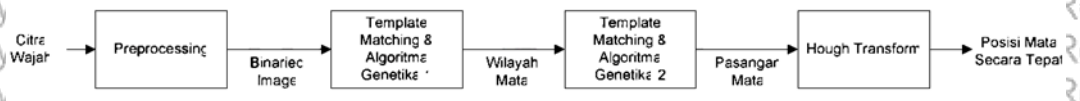
Bab ini berisi tentang saran dan kesimpulan dari hasil pengujian sistem yang dapat digunakan pada masa mendatang.

BAB 2 LANDASAN KEPUSTAKAAN

2.1 Kajian Pustaka

Tinjauan pustaka dilakukan dengan menganalisa dan membandingkan metode-metode yang digunakan pada penelitian sebelumnya untuk proses deteksi mata. Penelitian sebelumnya yang dilakukan Muhammad Syarif & Wijanarto (2015), menggunakan metode *Haar Cascade Classifier* untuk mendeteksi bagian mata. Hasil deteksi kemudian dianalisis *contour*-nya untuk menentukan kedipan mata.

Penelitian yang dilakukan Aji Atturmudzi (2008) tentang deteksi mata menggunakan *Template Matching*, Algoritme Genetika dan *Hough Transform*. Citra *input* merupakan citra wajah yang dilakukan preprocessing untuk dihasilkan citra biner. Template Matching dan Algoritme Genetika digunakan untuk mendeteksi wilayah mata dan *Hough Transform* digunakan untuk menentukan posisi mata secara tepat. Diagram blog sistem dapat dilihat pada Gambar 2.1.



Gambar 2.1 Diagram blok sistem penelitian Aji Atturmudzi

Sumber: Atturmudzi, et al. (2008)

Pada penelitian ini tahap *pre-processing* yang dilakukan menggunakan *homomorphic filtering* yang sangat berpengaruh pada pendeteksian wilayah mata menggunakan *template matching*. Algoritme genetika dikombinasikan dengan *template matching* untuk menghasilkan solusi yang lebih baik dan lebih cepat.

Penelitian yang dilakukan Danukusumo (2017) dalam implementasi *Convolution Neural Network* untuk melakukan klasifikasi pada gambar candi dengan menggunakan Graphic Processing Unit (GPU) untuk menganalisa kinerja pelatihan dari model yang sudah dibuat menghasilkan akurasi sebesar 0,9899 pada data *training* dan 0,8557 pada data uji. Sehingga dapat disimpulkan bahwa algoritme CNN dapat digunakan untuk klasifikasi citra candi dengan sangat baik.

Abhirawa (2017) melakukan penelitian menggunakan *Convolution Neural Network* untuk pengenalan wajah, hasilnya menunjukan tingkat akurasi pengenalan setinggi 0,8973 pada *data learning* dan akurasi pengenalan setinggi 0,7579 apabila dilakukan pengujian terhadap *data testing*.

Tabel 2.1 Kajian Pustaka

No	Judul	Objek	Metode Proses	Keterangan
1	Convolution Neural Networks Untuk Pengenalan Wajah Secara Real-Time	Wajah Manusia	Convolution Neural Network	Kinerja pengenalan wajah meraih rata-rata tingkat akurasi lebih dari 0,89 dalam 2 frame per detik.
2	Implementasi Metode Convolution Neural Network Untuk Klasifikasi Tanaman Pada Citra Resolusi Tinggi	Teknologi UAV (Unmanned Aerial Vehicle)	Convolution Neural Network	Penelitian menggunakan 5 kelas tanaman, yaitu padi, bawang merah, kelapa, pisang, cabai. Proses learning menghasilkan akurasi 1,00 terhadap data training. Pengujian validasi menghasilkan akurasi 0,93 dan terhadap data tes sebesar 0,82
3	Penerapan *Deep Learning* Menggunakan *Convolution Neural Network* Untuk Klasifikasi CItra Wayang Punakawan	Citra Wayang Punakawan	Convolution Neural Network	Hasil dari model yang diuji menghasilkan akurasi terhadap 0,916 terhadap citra uji.
4	Implementasi *Convolution Neural Networks* untuk Klasifikasi Citra Tomat Menggunakan Keras	Citra Tomat	Convolution Neural Network	Pengujian terhadap 100 sampel tomat menghasilkan akurasi sebesar 0,9 dan dinilai mampu melakukan identifikasi kelayakan tomat.
5	*Convolution Neural Networks for Handwritten Javanese Character Recognition*	Javanese Character	Convolution Neural Network	Hasil dari pengujian CNN mampu lebih baik dari model MLP dengan waktu latih yang lebih lama.

2.2 Landasan Teori

2.2.1 Citra Digital

Citra merupakan sebuah informasi visual yang diperoleh dari penangkapan intensitas cahaya yang dipantulkan oleh suatu objek. Sumber cahaya menerangi objek dan sebagian dari cahaya tersebut dipantulkan kembali. Cahaya yang dipantulkan tersebut kemudian ditangkap oleh benda optis yang dapat merekam intensitas cahaya, misalnya mata, kamera, dan sebagainya.

Citra digital merupakan representasi dari citra 2 dimensi yang merupakan kelompok data digital yang memiliki nilai data pada setiap piksel. Piksel

merupakan elemen terkecil dari suatu citra digital. Ukuran dari citra digital menunjukkan banyaknya informasi piksel. Data 2 dimensi pada citra digital dapat direpresentasikan dalam bentuk larik (*array*) yang berisi nilai real/kompleks dalam jangkauan deretan bit tertentu (Putra, 2010).

Citra digital merupakan fungsi *f(x,y),* dengan nilai *x* dan *y* adalah koordinat spasial dan nilai dari f menunjukkan tingat kecerahan dari suatu piksel

2.2.2 Jenis Citra

Suatu piksel dalan citra memiliki rentang nilai tertentu dan jangkauan nilainya berbeda tergantung jenisnya. Nilai jangkauan suatu piksel biasanya adalah 0-255. Berikut ini adalah jenis citra berdasarkan nilai tiap piksel.

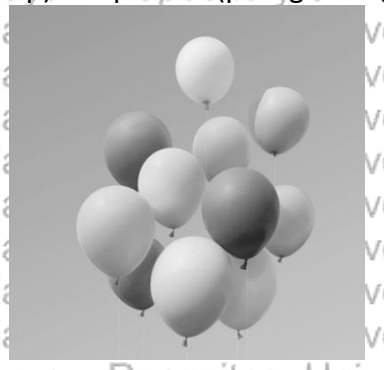
1. Citra warna (RGB)

   Citra berwarna tersusun dari 3 lapisan matriks yang menunjukkan nilai dari warna R (merah), G (hijau) dan B (Biru). Intensitas masing-masing lapisan memiliki rentang nilai 8 bit, sehingga jumlah keseluruhan menjadi 24-bit warna (Putra, 2010).

Gambar 2.2 Citra Warna RGB

2. Citra keabuan (*greyscale*)

   Citra *greyscale* adalah citra yang tersusun dari warna hitam, keabuan, dan putih. Citra ini hanya memiliki satu lapisan matrik. Warna keabuan memiliki rentang dari 0 (paling gelap) sampai 255 (paling terang) (Putra, 2010).



Gambar 2.3 Citra Warna Greyscale

3. Citra biner

Citra biner adalah citra yang memiliki dua kemungkinan warna, yaitu hitam dan putih. Citra biner hanya memiliki rentang nilai 1 bit untuk menyimpan informasi warna tiap pikselnya. Citra biner sering digunakan pada proses segmentasi dan morfologi (Putra, 2010).

Gambar 2.4 Citra Warna Biner

2.2.3 Machine Learning

Machine Learning adalah salah satu bagian dalam *Artificial Intelligence* yang sering dipakai untuk menirukan perilaku manusia dalam penyelesaian suatu masalah dan membuat melakukan otomatisasi dengan cara mencoba meneladani manusia dalam proses belajar dan menggeneralisasi sesuatu.

Dalam *machine learning* terdapat proses training dan *testing*. Kedua proses tersebut memerlukan data. Data *training* digunakan untuk melatih algoritme *machine learning*, sedangkan data *testing* digunakan pada data yang baru selain yang terdapat pada data training. Hasil dari data *testing* digunakan untuk mengetahui performa dan akurasi ketika menemukan data baru yang belum pernah dilihat.

Menurut Ahmad (2017) terdapat 2 aplikasi dalam *machine learning* yaitu, klasifikasi dan prediksi. Klasifikasi yang terdapat dalam machine learning digunakan untuk memilih/mengklasifikasikan berdasarkan ciri-ciri khusus yang terdapat dalam obyek. Prediksi digunakan oleh untuk memperkirakan *output* berdasarkan data *training*. Terdapat 4 jenis metode pembelajaran dalam machine learning, yaitu:

1. *Supervised Learning*
2. *Unsupervised Learning*
3. *Semi-supervised Learning*
4. *Reinforcement Learning*

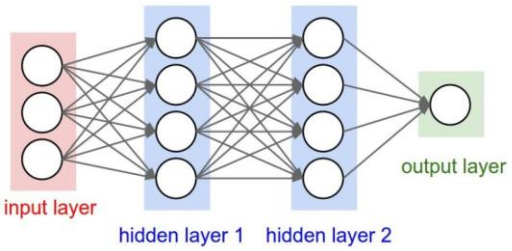
2.2.4 Deep Learning

Deep Learning adalah salah satu cabang dari *Machine Learning* yang berlandaskan pada jaringan saraf tiruan yang dapat membuat komputer untuk

melakukan pembelajaran dari contoh-contoh data yang dilatihkan. Dalam *Deep Learning*, komputer dapat belajar secara langsung dari gambar, suara, teks, dll.

Deep Learning tersusun dari input *layer*, *hidden layer* dan *output layer*. *Input layer* berisikan *node-node* yang menyimpan nilai data masukan. *Hidden layer* dapat dibentuk dengan beberapa lapisan untuk menemukan komposisi algoritme yang tepat supaya dapat mengurangi *error* pada *output*. Kemudian fungsi aktivasi pada hidden layer kemudian menghasilkan *output layer* yang datanya berasal dari *input layer*.

Arsitektur dalam *Deep Learning* memiliki 3 bagian, yaitu terdiri dari input layer, hidden layer, dan output layer yang ditunjukkan pada Gambar 2.6 dan disebut sebagai *Multi Layer Perceptron* (MLP).

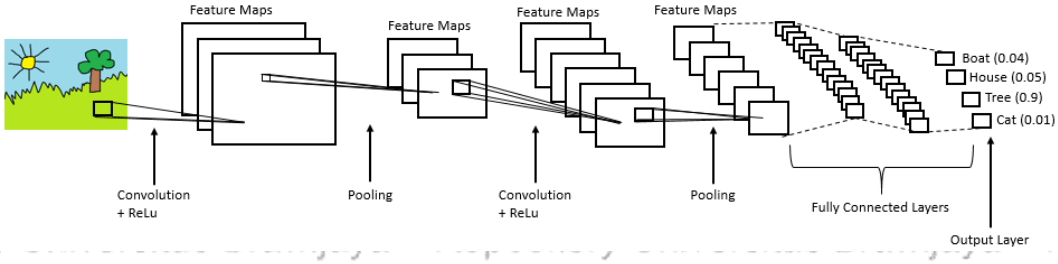


Gambar 2.5 Layer-layer pada Deep Learning

Deep Learning dapat memungkinkan membuat model komputasi yang tersusun dari beberapa *processing layer* untuk mempelajari pola pada sebuah data dalam berbagai tingkatan. Metode ini banyak digunakan dalam deteksi objek, pengenalan suara, dll. Algoritme yang menggunakan *Deep Learning* salah satunya adalah *Convolution Neural Network* (CNN) untuk melakukan klasifikasi sebuah citra.

2.2.5 *Convolution Neural Network* (CNN)

Convolution Neural Network (CNN) adalah salah satu algoritme penerapan daripada *Deep Learning* yang merupakan pengembangan dari *Multi Layer Perceptron* (MLP). Algoritme ini sangat cocok untuk mengolah data dalam bentuk grid, contohnya data dalam bentuk gambar. *Convolution Neural Network* digunkaan untuk mengklasifikasikan data yang sudah diberikan label dengan menggunakan metode *supervised learning*. Dalam *supervised learning* terdapat dua macam jenis data, yaitu data yang dilatih dan terdapat pula data nilai target tujuannya. Metode *supervised learning* ini bertujuaan untuk mengelompokan suatu data. CNN sering digunakan untuk mengenali benda-benda, melakukan deteksi atau segmentasi pada objek.

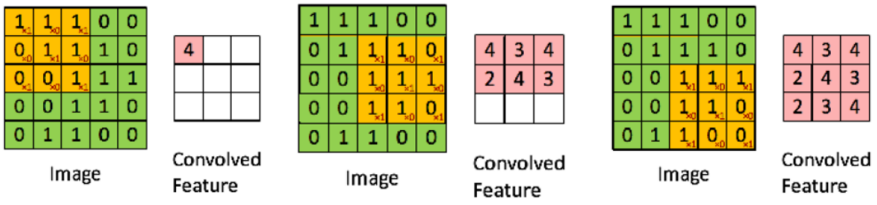


Gambar 2.6 Arsitektur Convolution Neural Network

Arsitektur yang dimiliki oleh *Convolution Neural Network* dapat dilihat pada Gambar 2.6 dan dapat dijelaskan sebagai berikut:

1. *Convolution Layer*

Convolution layer merupakan hasil operasi konvolusi pada citra atau *output* dari lapisan sebelumnya. *Layer* ini merupakan proses paling mendasar dari algoritme CNN. Konvolusi mengaplikasikan sebuah *kernel* pada citra seperti yang ditunjukan sebagai kotak kuning pada Gambar 2.8, kotak berwarna hijau adalah citra yang akan dilakukan proses konvolusi. Kernel digerakkan dari bagian kiri atas menuju bagian kanan bawah gambar, sehingga hasil dari proses konvolusi citra tersebut dapat dilihat pada Gambar 2.8.



Gambar 2.7 Proses *Convolution Layer*

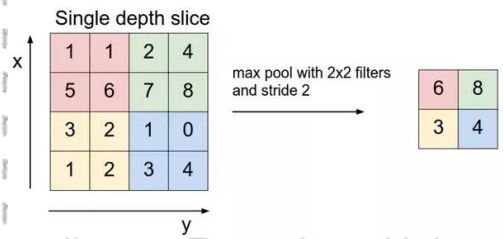
Tujuan dari proses konvolusi pada citra adalah untuk melakukan ekstraksi fitur. Hasil dari konvolusi berupa transformasi linear sesuai dengan informasi spasial pada data citra input.

2. *Pooling Layer*

Pooling Layer adalah lapisan dalam CNN yang menggunakan fungsi *feature map* untuk diolah dengan berbagai macam operasi statistik berdasarkan nilai piksel di sekitarnya. Pooling layer umumnya ditaruh setelah proses *convolution layer*. Pooling layer yang berada di antara lapisan konvolusi secara berurutan dalam arsitektur model *Convolution Neural Network* dapat secara signifikan dapat mengurangi jumlah data pada output *feature map*, sehingga jumlah kalkulasi yang diperlukan dalam jaringan dapat berkurang.

Berbagai jenis pooling layer digunakan dalam CNN adalah dengan mengambil nilai maksimal (*max-pooling*) atau nilai rata-rata (*average pooling*) dari bagian-bagian piksel pada citra. Metode pooling yang umum digunakan

adalah metode *max-pooling*. Proses dari *max-pooling* ditunjukan pada Gambar 2.8.



Gambar 2.8 Pooling Layer Metode *Max Polling*

3. Fully Connected Layer

Fully connected layer adalah suatu lapisan yang semua neuron aktivasi terhubung satu dengan lainnya. *Fully connected layer* digunakan dalam penerapan *Multi Layer Perceptron* (MLP) yang bertujuan untuk melakukan transformasi supaya dapat diklasifikasikan secara linear.

Terdapat perbedaan antara *fully connected layer* dan *convolution layer*. Neuron pada *fully connected layer* semuanya saling terhubung sedangkan pada *convolution layer* yang terhubung hanya beberapa daerah tertentu pada bagian input.

2.2.6 Python

Python adalah salah satu jenis bahasa pemrograman tingkat tinggi. Bahasa ini termasuk produk yang *free*, bersifat *opensource* dan dapat dijalankan di berbagai platform. Python termasuk bahasa yang jelas dan memiliki dukungan yang baik terhadap *input/output*, angka, gambar dan *plotting*. Bahasa ini memiliki sintaksis yang ringkas dan mudah digunakan. Python memiliki beberapa kelebihan dari bahasa pemrograman lain terlebih lagi dalam hal banyaknya modul yang dimilikinya.

Di dalam pemrograman *computer vision*, sering dilakukan reprentasi vektor dan matriks dan operasi terhadapnya (Solem, 2012). Python memiliki modul NumPy yang akan merepresentasikan vektor dan matriks dalam bentuk *array*. Untuk memvisualisasikan *output* dapat digunakan modul *Matplotlib* dan untuk operasi matematika tingkat lanjut, dapat menggunakan modul *SciPy*.

dlib adalah salah satu *library cross-platform* yang terdapat pada python yang memuat algoritme *Facial Landmark Detector*. *Facial Landmark Detector* adalah algoritme yang telah dibuat untuk dapat mendeteksi bagian-bagian wajah seperti mulut, hidung, mata dll, dalam waktu mili-detik dengan akurasi tinggi (Kazemi, et al., 2014). *Facial Landmark* yang diterapkan dalam *library* ini menggunakan 68 titik pada dataset *iBUG 300-W*.

BAB 3 METODOLOGI PENELITIAN

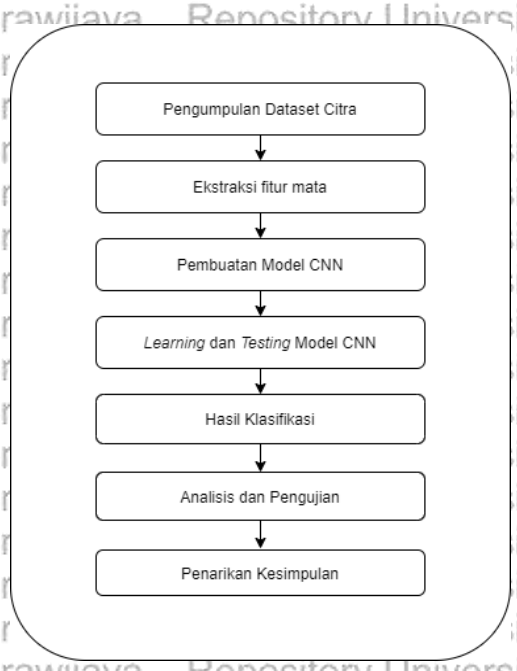
Bab ini menjelaskan tentang langkah-langkah metodologi yang digunakan untuk melakukan penelitian tentang deteksi posisi dan arah mata menggunakan *convolution neural network*.

3.1 Tipe Penelitian

Tipe penelitian yang dilakukan pada penelitian ini adalah non-implementatif diskriptif yang menerapkan metode-metode tententu terhadap permasalahan. Pada penelitian dijelaskan pula karakteristik dari objek yang diteliti dan akan disampaikan hasil dari analisis penelitian.

3.2 Strategi penelitian

Strategi penelitian menjelaskan alur dari jalannya sebuh sistem *input* berupa citra dengan objek merupakan wajah seseorang. Citra tersebut dilakukan *pre-processing,* dan ekstraksi fitur wajah khususnya pada mata dengan menggunakan *facial landmark*. Secara garis besar, strategi penelitian ini ditunjukkan pada Gambar 3.1.



Gambar 3.1 Strategi Penelitian

Berdasarkan pada Gambar 3.1, proses penelitian dimulai dari pengumpulan dataset citra yang digunakan dalam proses *training* dan *testing*. Selanjutnya dibentuklah metode CNN yang diharapkan dapat mengklasifikasikan citra mata. Hasil dari perancangan CNN kemudian diterapkan pada data *training* supaya komputer dapat mengenali klasifikasi dari citra. Setelah itu hasil *training* dilakukan uji coba untuk mengetahui hasil akurasi.

3.2.1 Lokasi Penelitian

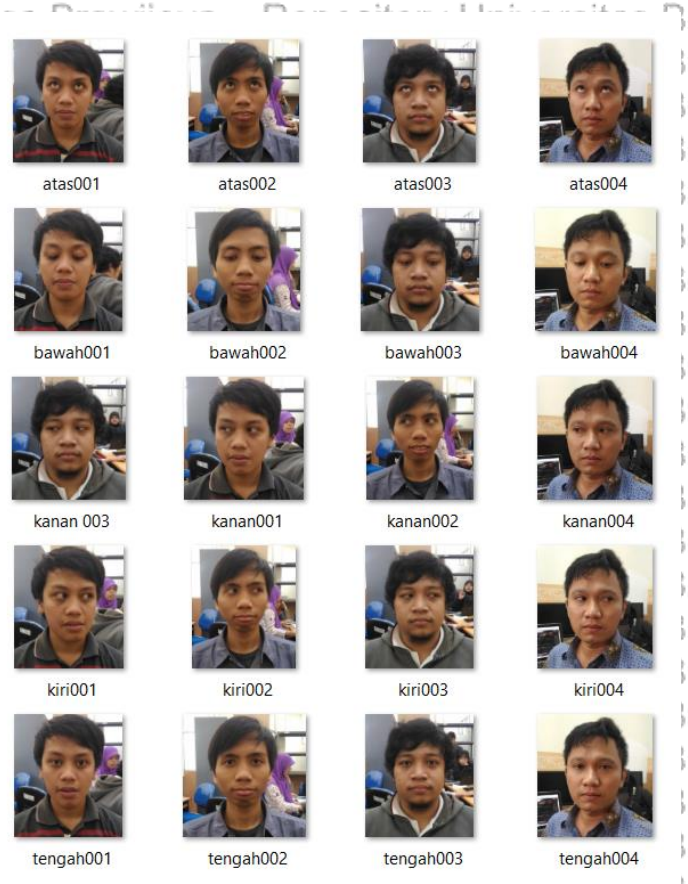
Lokasi peneletian berada di wilayah Universitas Brawijaya, tepatnya Ruang 4.2 Gedung F, lantai 4, Fakultas Ilmu Komputer.

3.2.2 Teknik Pengumpulan Data

Data yang digunakan adalah data citra wajah seseorang yang diambil secara manual dengan *smartphone* dan menyimpan citra wajah dengan arah mata yang berbeda-beda. Objek yang digunakan dalam penelitian ini yaitu:

1. Citra wajah orang yang digunakan tampak seluruh bagian wajah, dan dengan posisi tegak lurus dengan arah mata yang berbeda-beda.
2. Kelas yang digunakan berdasarkan posisi gerakan mata pada wajah, yaitu tengah, kiri, atas, kanan dan bawah.

Data berupa citra hasil dari proses pengumpulan dikelompokkan berdasarkan kelasnya masing-masing. Contoh hasil pengumpulan data yang sudah dinamai berdasarkan kelasnya ditampilkan pada Gambar 3.2.



Gambar 3.2 Citra Hasil Pengumpulan Data

3.2.3 Peralatan Pendukung

Peralatan pendukung adalah alat penunjang proses penelitian yang diperlukan dalam pengembangan sistem supaya berjalan sesuai harapan. Kebutuhan peralatan pendukung dari penelitian ini sebagai berikut.

1. Spesifikasi *hardware*.

   Laptop dengan spesifikasi Prosesor Intel(R) Core(TM) i3-8135U @2.1Ghz, dengan RAM 8GB DDR3 dan ruang penyimpanan 512GB SSD.
   Smartphone Sony Xperia XZ1 dengan kamera beresolusi 19MP.

2. Spesifikasi *software*.

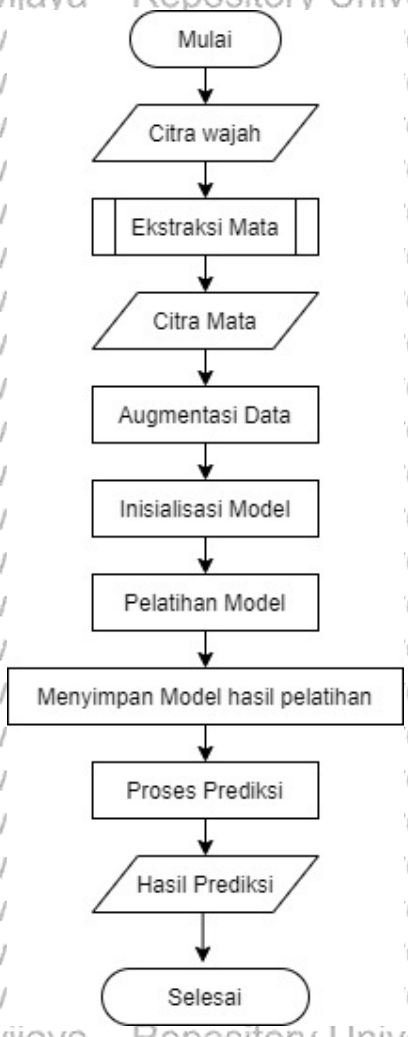
   Sistem operasi menggunakan Windows 10 64 bit build 1903, Microsoft Office 2019, IDE Spyder dengan Python 3.7.

BAB 4 PERANCANGAN

Bab ini akan menjelaskan bagaimana proses perancangan digunakan dalam penelitian ini dan sebagai acuan implementasinya. Perancangan ini terdiri dari perancangan *algoritme*, perhitungan manualisasi dan dilanjutkan dengan scenario pengujian.

4.1 Perancangan Proses CNN

Perancangan proses yang dalam penelitian ini secara garis besarnya dibagi menjadi 2 bagian, bagian yang pertama adalah proses pelatihan terhadap data latih, berikutnya proses klasifikasi terhadap data uji. Perancangan keseluruhan sistem disusun pada flowchart di Gambar 4.1.



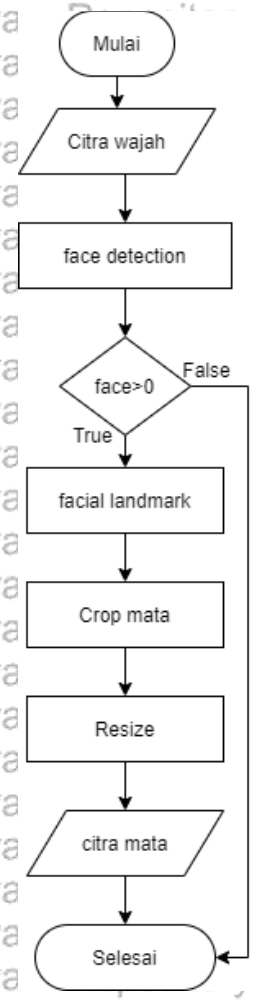
Gambar 4.1 Diagram Alir Secara Umum

Seperti yang ditunjukkan pada Gambar 4.1, sistem menerima masukan berupa citra wajah. Citra wajah tersebut kemudian dilakukan proses ekstraksi mata. Kemudian gambar mata yangg berhasil diekstrak kemudian akan dinormalisasi yang kemudian dilakukan proses pembelajaran dengan suatu

model yang telah diinisialisasikan atau sudah ditentukan arsitekturnya. Setelah proses pembelajaran selesai maka bobot, bias, dan arsitekturnya akan disimpan agar saat ingin memprediksi citra baru tidak perlu untuk melakukan proses pembelajaran kembali. Proses pembelajaran sudah selesai maka dilakukan proses prediksi dengan data uji yang telah dinormalisasikan

4.1.1 Ekstraksi Mata

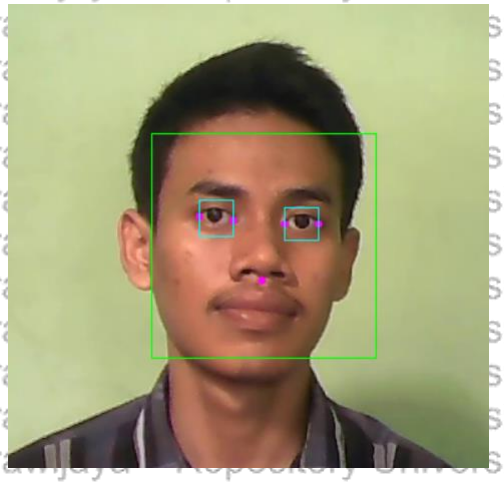
Ekstraksi fitur mata merupakan pre-processing yang dilakukan dalam peneletian ini untuk mengambil citra mata dari citra wajah. Gambar 4.2 menunjukkan proses dalam mengekstraksi citra mata.



Gambar 4.2 Proses Ekstraksi Citra Mata

Berdasarkan Gambar 4.2, proses ekstraksi citra mata diawali dengan proses face detection. Jika dalam citra ditemukan wajah maka akan dilanjutkan dengan deteksi fitur-fitur pada wajah dengan facial landmark. Umumnya *face landmark* menggunakan 68 *point*, dalam penelitian ini hanya digunakan 5 *point* karena hanya berfokus pada bagian mata saja. Setelah *face landmark* berhasil mendeteksi lokasi mata dalam citra wajah, maka dilanjutkan proses *crop* daerah tersebut. Hasil *crop* mata kemudian dilakukan *resize* dengan resolusi 32x32 piksel. Hasil mata kemudian disimpan supaya dapat dilakukan proses pelatihan

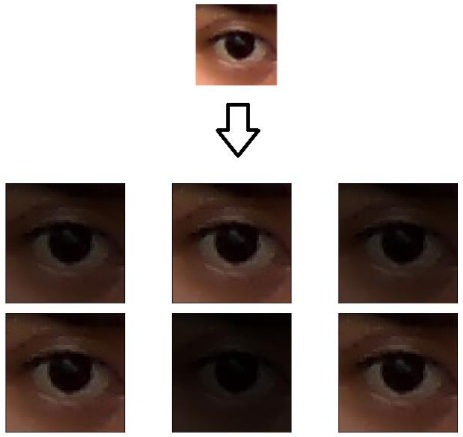
dan pengujian dalam CNN. Gambar 4.3 menunjukkan *facial landmark* dalam mendeteksi fitur mata dalam citra wajah.



Gambar 4.3 Hasil Deteksi Mata dengan *Facial Landmark*

4.1.2 Augmentasi Data

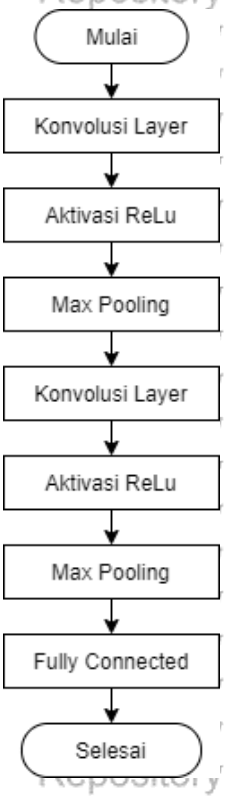
Augmentasi data adalah pemrosesan data sebelum dilakukan proses pelatihan dengan mengubah data mentah menjadi data berkualitas. Berbagai macam cara melakukan augmentasi data tergantung kebutuhan sistem, diantaranya yang sering digunakan dalam memproses sebuah citra adalah *cropping* citra, *resize* citra, *flip* citra, *rotate* citra, *zoom* citra dll. Dalam penelitian ini penggunaan augmentasi citra untuk melakukan variasi terhadap data berupa *brightness* pada citra dan menormalisasi citra, atau mengubah nilai per-piksel citra menjadi nilai di antara 0 dan 1. Tahap normalisasi data bertujuan untuk mengurangi dominasi antara variabel dengan nilai yang besar terhadap variabel dengan nilai yang kecil. Gambar 4.4 merupakan salah satu contoh dari proses augmentasi, di mana sebuah citra dapat menghasilkan data yang lebih banyak dan dapat digunakan untuk proses *training* dan *testing* dalam CNN.



Gambar 4.4 Contoh Augmentasi Citra Mata

4.1.3 Inisialisasi Model

Inisialisasi model dalam CNN akan menentukan lapisan dan kedalaman yang akan digunakan. Metode CNN berfokus pada pembelajaran fitur pada suatu gambar maka yang perlu untuk ditentukan adalah *layer convolutional* yaitu lapisan konvolusi dan *pooling*, sedangkan *fully connected* adalah metode klasifikasinya maka dalam penelitian ini cukup dengan 2 lapisan. Gambar 4.3 merupakan ilustrasi model yang akan digunakan.



Gambar 4.5 Perancangan Model

Lapisan konvolusi dan *pooling* sangat berpengaruh dalam menentukan hasil klasifikasi seperti yang ditunjukkan Gambar 4.5. Lapisan konvolusi dan *pooling* menghasilkan *output* berupa citra dengan dimensi yang lebih kecil atau sama dengan masukannya.

Dalam penelitian ini filter konvolusi yang digunakan adalah filter berukuran 3x3 dengan stride 1 untuk proses konvolusi. *Sedangkan* dalam proses *pooling* menggunakan filter 2x2 dengan stride 2 untuk memperkecil ukuran *image*. Ukuran filter yang semakin kecil akan menyebabkan sistem yang detail. Informasi dari citra akan semakin banyak dan menyebabkan efek samping berupa efek komputasi yang semakin besar pula.

Pada Tabel 4.1 merupakan hasil perhitungan citra dan kedalaman lapisan konvolusi dan *pooling* yang digunakan dalam penelitian ini.

Tabel 4.1 Perancangan Kedalaman Model

Layer (type)	Output Shape
conv2d_1 (Conv2D)	(None, 30, 30, 16)
activation_1 (Activation)	(None, 30, 30, 16)
max_pooling2d_1 (MaxPooling2D)	(None, 15, 15, 16)
conv2d_2 (Conv2D)	(None, 13, 13, 32)
activation_2 (Activation)	(None, 13, 13, 32)
max_pooling2d_2 (MaxPooling2D)	(None, 6, 6, 32)
flatten_1 (Flatten)	(None, 1152)
dense_1 (Dense)	(None, 20)
dense_2 (Dense)	(None, 5)

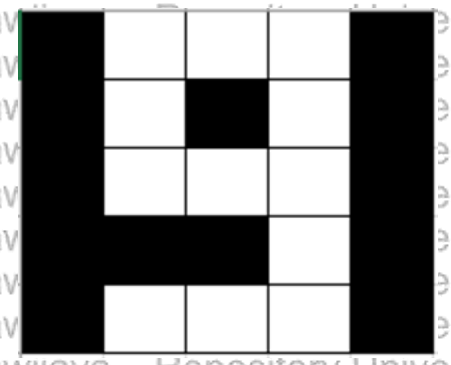
Berdasarkan Tabel 4.1 kedalam model yang dapat digunakan 32x32 adalah 2 kedalaman *layer* konvolusi (konvolusi dan *pooling*). Dalam melakukan proses konvolusi, besar matriks masukkan minimal harus lebih atau sama dengan besar matriks filternya. Penggunaan *filter* 3x3 untuk konvolusi tidak dapat lebih dari dua kedalaman *layer* konvolusi karena keluaran pada lapisan 2 sudah berupa matriks dengan ukuran kecil, yaitu 6x6. Pertama citra berukuran 32x32 dilakukan proses konvolusi dengan 16 *filter*, menghasilkan 16 *matrix* beresolusi 30x30. Kemudian dilakukan aktivasi ReLu untuk menghilangkan angka negatif pada *matrix*. Hasil aktivasi dilakuan pooling dengan metode *max pooling* menghasilkan *matrix* yang jauh lebih kecil yaitu 15x15. Hasil dari *max pooling* dilanjukan ke proses konvolusi tahap ke 2 dengan 32 filter yang menghasilkan *matrix* berukuran 13x13. Setelah dilakukan proses aktivasi *ReLu* dan *max pooling* kembali, menghasilkan matrix berukuran 6x6 sebanyak 32 layer. Kemudian *matrix* tersebut dilakukan proses *flatten* yang mengubahnya menjadi 1 dimensi dengan 1152 nilai. Nilai-nilai *flatten* tersebut menunjukkan bobot awal dan pada bagian *dense* menunjukkan jumlah neuron yang digunakan.

4.1.4 Pelatihan Model

Pelatihan model merupakan proses pembelajaran model hasil inisialisasi sebelumnya dengan data yang sudah dinormalisasikan. Dalam pelatihan model metode optimasi cukup berpengaruh dalam hasil akurasi dan berpengaruh pula pada kecepatan pembelajaran sistem. Terdapat beberapa metode optimasi stokastik beberapa contohnya adalah Stochastic Gradient Descent (SGD) Nesterov, Adadelta, Adagrad, RMSprop, dan Adam. Menurut Kingma & Ba, (2015) dalam penelitiannya menyebutkan bahwa optimasi Adam merupakan optimasi terbaik. Optimasi Adam menggabungkan keunggulan dari optimasi populer sebelumnya yaitu RMSprop dan Adagard dapat kemampuan menangani data non-stasioner dan data *sparse* dengan memori lebih sedikit.

4.2 Perhitungan Manualisasi

Perhitungan manualisasi metode CNN dilakukan dengan contoh sebuah citra berukuran 5x5 seperti pada Gambar 4.6. Citra input kemudian dilakukan konversi ke bentuk angka biner dan kemudian ditambahkan *padding* seperti pada Gambar 4.7.



Gambar 4.6 Contoh Citra Perhitungan Manual

0	0	0	0	0	0	0
0	0	1	1	1	0	0
0	0	1	0	1	0	0
0	0	1	1	1	0	0
0	0	0	0	1	0	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0

Gambar 4.7 Contoh Citra dalam Bentuk Angka dengan *Padding*

Inisialisasi random filter untuk proses konvolusi.

Filter 1			Filter 2			Filter 3			Filter 4		
1	0	-1	-1	0	1	1	1	1	-1	-1	-1
1	0	-1	-1	0	1	0	0	0	0	0	0
1	0	-1	-1	0	1	-1	-1	-1	1	1	1

Lakukan proses konvolusi citra dengan masing-masing filter diatas

$$H_{\_netk} = \sum_{k=1}^{n} x_k * w_{ki}$$

$$(x_k * w_{kl})_{ij} = \sum_m \sum_n x_{i+m,j+n} . w_{m,n}$$

$$H\_net_k = \begin{bmatrix} (x_k * w_{kl})_{11} & \cdots & (x_k * w_{kl})_{1j} \\ \vdots & \ddots & \vdots \\ (x_k * w_{kl})_{i1} & \cdots & (x_k * w_{kl})_{ij} \end{bmatrix}$$

$H\_net1_{11} = 0.1 + 0.0 + 0.-1 + 0.1 + 0.0 + 1.-1 + 0.1 + 0.0 + 1.-1 = -2$

$H\_net1_{12} = 0.1 + 0.0 + 0.-1 + 0.1 + 1.0 + 1.-1 + 0.1 + 1.0 + 0.-1 = -1$

$H\_net1_{13} = 0.1 + 0.0 + 0.-1 + 1.1 + 1.0 + 1.-1 + 1.1 + 0.0 + 1.-1 = 0$

$H\_net1_{14} = 0.1 + 0.0 + 0.-1 + 1.1 + 1.0 + 0.-1 + 0.1 + 1.0 + 0.-1 = 1$

$H\_net1_{15} = 0.1 + 0.0 + 0.-1 + 1.1 + 0.0 + 0.-1 + 1.1 + 0.0 + 0.-1 = 2$

$H\_net1_{21} = 0.1 + 0.0 + 1.-1 + 0.1 + 0.0 + 1.-1 + 0.1 + 0.0 + 1.-1 = -3$

$H\_net1_{22} = 0.1 + 1.0 + 1.-1 + 0.1 + 1.0 + 0.-1 + 0.1 + 1.0 + 1.-1 = -2$

$H\_net1_{23} = 1.1 + 1.0 + 1.-1 + 1.1 + 0.0 + 1.-1 + 1.1 + 1.0 + 1.-1 = 0$

$H\_net1_{24} = 1.1 + 1.0 + 0.-1 + 0.1 + 1.0 + 0.-1 + 1.1 + 1.0 + 0.-1 = 2$

$H\_net1_{25} = 1.1 + 0.0 + 0.-1 + 1.1 + 0.0 + 0.-1 + 1.1 + 0.0 + 0.-1 = 3$

$H\_net1_{31} = 0.1 + 0.0 + 1.-1 + 0.1 + 0.0 + 1.-1 + 0.1 + 0.0 + 0.-1 = -2$

$H\_net1_{32} = 0.1 + 1.0 + 0.-1 + 0.1 + 1.0 + 1.-1 + 0.1 + 0.0 + 0.-1 = -1$

$H\_net1_{33} = 1.1 + 0.0 + 1.-1 + 1.1 + 1.0 + 1.-1 + 0.1 + 0.0 + 1.-1 = -1$

$H\_net1_{34} = 0.1 + 1.0 + 0.-1 + 1.1 + 1.0 + 0.-1 + 0.1 + 1.0 + 0.-1 = 1$

$H\_net1_{35} = 1.1 + 0.0 + 0.-1 + 1.1 + 0.0 + 0.-1 + 1.1 + 0.0 + 0.-1 = 3$

$H\_net1_{41} = 0.1 + 0.0 + 1.-1 + 0.1 + 0.0 + 0.-1 + 0.1 + 0.0 + 1.-1 = -2$

$H\_net1_{42} = 0.1 + 1.0 + 1.-1 + 0.1 + 0.0 + 0.-1 + 0.1 + 1.0 + 1.-1 = -2$

$H\_net1_{43} = 1.1 + 1.0 + 1.-1 + 0.1 + 0.0 + 1.-1 + 1.1 + 1.0 + 1.-1 = -1$

$H\_net1_{44} = 1.1 + 1.0 + 0.-1 + 0.1 + 1.0 + 0.-1 + 1.1 + 1.0 + 0.-1 = 2$

$H\_net1_{45} = 1.1 + 0.0 + 0.-1 + 1.1 + 0.0 + 0.-1 + 1.1 + 0.0 + 0.-1 = 3$

$H\_net1_{51} = 0.1 + 0.0 + 0.-1 + 0.1 + 0.0 + 1.-1 + 0.1 + 0.0 + 0.-1 = -1$

$H\_net1_{52} = 0.1 + 0.0 + 0.-1 + 0.1 + 1.0 + 1.-1 + 0.1 + 0.0 + 0.-1 = -1$

$H\_net1_{53} = 0.1 + 0.0 + 1.-1 + 1.1 + 1.0 + 1.-1 + 0.1 + 0.0 + 0.-1 = -1$

$H\_net1_{54} = 0.1 + 1.0 + 0.-1 + 1.1 + 1.0 + 0.-1 + 0.1 + 0.0 + 0.-1 = 1$

$H\_net1_{55} = 1.1 + 0.0 + 0.-1 + 1.1 + 0.0 + 0.-1 + 0.1 + 0.0 + 0.-1 = 2$

$$H_{-netk1} = \begin{bmatrix} -2 & -1 & 0 & 1 & 2 \\ -3 & -2 & 0 & 2 & 3 \\ -2 & -1 & -1 & 1 & 3 \\ -2 & -2 & -1 & 2 & 3 \\ -1 & -1 & -1 & 1 & 2 \end{bmatrix}$$

$H_{-net2_{11}}$ = 0.-1 + 0.0 + 0.1 + 0.-1 + 0.0 + 1.1 + 0.-1 + 0.0 + 1.1 = 2

$H_{-net2_{12}}$ =0.-1 + 0.0 + 0.1 + 0.-1 + 1.0 + 1.1 + 0.-1 + 1.0 + 0.1 = 1

$H_{-net2_{13}}$ =0.-1 + 0.0 + 0.1 + 1.-1 + 1.0 + 1.1 + 1.-1 + 0.0 + 1.1 = 0

$H_{-net2_{14}}$ =0.-1 + 0.0 + 0.1 + 1.-1 + 1.0 + 0.1 + 0.-1 + 1.0 + 0.1 = -1

$H_{-net2_{15}}$ =0.-1 + 0.0 + 0.1 + 1.-1 + 0.0 + 0.1 + 1.-1 + 0.0 + 0.1 = -2

$H_{-net2_{21}}$ =0.-1 + 0.0 + 1.1 + 0.-1 + 0.0 + 1.1 + 0.-1 + 0.0 + 1.1 = 3

$H_{-net2_{22}}$ =0.-1 + 1.0 + 1.1 + 0.-1 + 1.0 + 0.1 + 0.-1 + 1.0 + 1.1 = 2

$H_{-net2_{23}}$ =1.-1 + 1.0 + 1.1 + 1.-1 + 0.0 + 1.1 + 1.-1 + 1.0 + 1.1 = 0

$H_{-net2_{24}}$ =1.-1 + 1.0 + 0.1 + 0.-1 + 1.0 + 0.1 + 1.-1 + 1.0 + 0.1 = -2

$H_{-net2_{25}}$ =1.-1 + 0.0 + 0.1 + 1.-1 + 0.0 + 0.1 + 1.-1 + 0.0 + 0.1 = -3

$H_{-net2_{31}}$ =0.-1 + 0.0 + 1.1 + 0.-1 + 0.0 + 1.1 + 0.-1 + 0.0 + 0.1 = 2

$H_{-net2_{32}}$ =0.-1 + 1.0 + 0.1 + 0.-1 + 1.0 + 1.1 + 0.-1 + 0.0 + 0.1 = 1

$H_{-net2_{33}}$ =1.-1 + 0.0 + 1.1 + 1.-1 + 1.0 + 1.1 + 0.-1 + 0.0 + 1.1 = 1

$H_{-net2_{34}}$ =0.-1 + 1.0 + 0.1 + 1.-1 + 1.0 + 0.1 + 0.-1 + 1.0 + 0.1 = -1

$H_{-net2_{35}}$ =1.-1 + 0.0 + 0.1 + 1.-1 + 0.0 + 0.1 + 1.-1 + 0.0 + 0.1 = -3

$H_{-net2_{41}}$ =0.-1 + 0.0 + 1.1 + 0.-1 + 0.0 + 0.1 + 0.-1 + 0.0 + 1.1 = 2

$H_{-net2_{42}}$ =0.-1 + 1.0 + 1.1 + 0.-1 + 0.0 + 0.1 + 0.-1 + 1.0 + 1.1 = 2

$H_{-net2_{43}}$ =1.-1 + 1.0 + 1.1 + 0.-1 + 0.0 + 1.1 + 1.-1 + 1.0 + 1.1 = 1

$H_{-net2_{44}}$ =1.-1 + 1.0 + 0.1 + 0.-1 + 1.0 + 0.1 + 1.-1 + 1.0 + 0.1 = -2

$H_{-net2_{45}}$ =1.-1 + 0.0 + 0.1 + 1.-1 + 0.0 + 0.1 + 1.-1 + 0.0 + 0.1 = -3

$H_{-net2_{51}}$ =0.-1 + 0.0 + 0.1 + 0.-1 + 0.0 + 1.1 + 0.-1 + 0.0 + 0.1 = 1

$H_{-net2_{52}}$ =0.-1 + 0.0 + 0.1 + 0.-1 + 1.0 + 1.1 + 0.-1 + 0.0 + 0.1 = 1

$H_{-net2_{53}}$ =0.-1 + 0.0 + 1.1 + 1.-1 + 1.0 + 1.1 + 0.-1 + 0.0 + 0.1 = 1

$H_{-net2_{54}}$ =0.-1 + 1.0 + 0.1 + 1.-1 + 1.0 + 0.1 + 0.-1 + 0.0 + 0.1 = -1

$H_{-net2_{55}}$ =1.-1 + 0.0 + 0.1 + 1.-1 + 0.0 + 0.1 + 0.-1 + 0.0 + 0.1 = -2

$$H_{-netk2} = \begin{bmatrix} 2 & 1 & 0 & -1 & -2 \\ 3 & 2 & 0 & -2 & -3 \\ 2 & 1 & 1 & -1 & -3 \\ 2 & 2 & 1 & -2 & -3 \\ 1 & 1 & 1 & -1 & -2 \end{bmatrix}$$

$H_{-net3_{11}}$ = 0.1 + 0.1 + 0.1 + 0.0 + 0.0 + 1.0 + 0.-1 + 0.-1 + 1.-1 = -1

$H_{-net3_{12}}$ =0.1 + 0.1 + 0.1 + 0.0 + 1.0 + 1.0 + 0.-1 + 1.-1 + 0.-1 = -1

$H_{-net3_{13}}$ =0.1 + 0.1 + 0.1 + 1.0 + 1.0 + 1.0 + 1.-1 + 0.-1 + 1.-1 = -2

$H_{-net3_{14}}$ =0.1 + 0.1 + 0.1 + 1.0 + 1.0 + 0.0 + 0.-1 + 1.-1 + 0.-1 = -1

$H_{-net3_{15}}$ =0.1 + 0.1 + 0.1 + 1.0 + 0.0 + 0.0 + 1.-1 + 0.-1 + 0.-1 = -1

$H_{-net3_{21}}$ =0.1 + 0.1 + 1.1 + 0.0 + 0.0 + 1.0 + 0.-1 + 0.-1 + 1.-1 = 0

$H_{-net3_{22}}$ =0.1 + 1.1 + 1.1 + 0.0 + 1.0 + 0.0 + 0.-1 + 1.-1 + 1.-1 = 0

$H_{-net3_{23}}$ =1.1 + 1.1 + 1.1 + 1.0 + 0.0 + 1.0 + 1.-1 + 1.-1 + 1.-1 = 0

$H_{-net3_{24}}$ =1.1 + 1.1 + 0.1 + 0.0 + 1.0 + 0.0 + 1.-1 + 1.-1 + 0.-1 = 0

$H_{-net3_{25}}$ =1.1 + 0.1 + 0.1 + 1.0 + 0.0 + 0.0 + 1.-1 + 0.-1 + 0.-1 = 0

$H_{-net3_{31}}$ =0.1 + 0.1 + 1.1 + 0.0 + 0.0 + 1.0 + 0.-1 + 0.-1 + 0.-1 = 1

$H_{-net3_{32}}$ =0.1 + 1.1 + 0.1 + 0.0 + 1.0 + 1.0 + 0.-1 + 0.-1 + 0.-1 = 1

$H_{-net3_{33}}$ =1.1 + 0.1 + 1.1 + 1.0 + 1.0 + 1.0 + 0.-1 + 0.-1 + 1.-1 = 1

$H_{-net3_{34}}$ =0.1 + 1.1 + 0.1 + 1.0 + 1.0 + 0.0 + 0.-1 + 1.-1 + 0.-1 = 0

$H_{-net3_{35}}$ =1.1 + 0.1 + 0.1 + 1.0 + 0.0 + 0.0 + 1.-1 + 0.-1 + 0.-1 = 0

$H_{-net3_{41}}$ =0.1 + 0.1 + 1.1 + 0.0 + 0.0 + 0.0 + 0.-1 + 0.-1 + 1.-1 = 0

$H_{-net3_{42}}$ =0.1 + 1.1 + 1.1 + 0.0 + 0.0 + 0.0 + 0.-1 + 1.-1 + 1.-1 = 0

$H_{-net3_{43}}$ =1.1 + 1.1 + 1.1 + 0.0 + 0.0 + 1.0 + 1.-1 + 1.-1 + 1.-1 = 0

$H_{-net3_{44}}$ =1.1 + 1.1 + 0.1 + 0.0 + 1.0 + 0.0 + 1.-1 + 1.-1 + 0.-1 = 0

$H_{-net3_{45}}$ =1.1 + 0.1 + 0.1 + 1.0 + 0.0 + 0.0 + 1.-1 + 0.-1 + 0.-1 = 0

$H_{-net3_{51}}$ =0.1 + 0.1 + 0.1 + 0.0 + 0.0 + 1.0 + 0.-1 + 0.-1 + 0.-1 = 0

$H_{-net3_{52}}$ =0.1 + 0.1 + 0.1 + 0.0 + 1.0 + 1.0 + 0.-1 + 0.-1 + 0.-1 = 0

$H_{-net3_{53}}$ =0.1 + 0.1 + 1.1 + 1.0 + 1.0 + 1.0 + 0.-1 + 0.-1 + 0.-1 = 1

$H_{-net3_{54}}$ =0.1 + 1.1 + 0.1 + 1.0 + 1.0 + 0.0 + 0.-1 + 0.-1 + 0.-1 = 1

$H_{-net3_{55}}$ =1.1 + 0.1 + 0.1 + 1.0 + 0.0 + 0.0 + 0.-1 + 0.-1 + 0.-1 = 1

$$H_{-netk3} = \begin{bmatrix} -1 & -1 & -2 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

$H_{-net4_{11}}$ =0.-1 + 0.-1 + 0.-1 + 0.0 + 0.0 + 1.0 + 0.1 + 0.1 + 1.1 = 1

$H_{-net4_{12}}$ =0.-1 + 0.-1 + 0.-1 + 0.0 + 1.0 + 1.0 + 0.1 + 1.1 + 0.1 = 1

$H_{-net4_{13}}$ =0.-1 + 0.-1 + 0.-1 + 1.0 + 1.0 + 1.0 + 1.1 + 0.1 + 1.1 = 2

$H_{-net4_{14}}$ =0.-1 + 0.-1 + 0.-1 + 1.0 + 1.0 + 0.0 + 0.1 + 1.1 + 0.1 = 1

$H_{-net4_{15}}$ =0.-1 + 0.-1 + 0.-1 + 1.0 + 0.0 + 0.0 + 1.1 + 0.1 + 0.1 = 1

$H_{-net4_{21}}$ =0.-1 + 0.-1 + 1.-1 + 0.0 + 0.0 + 1.0 + 0.1 + 0.1 + 1.1 = 0

$H_{-net4_{22}}$ =0.-1 + 1.-1 + 1.-1 + 0.0 + 1.0 + 0.0 + 0.1 + 1.1 + 1.1 = 0

$H_{-net4_{23}}$ =1.-1 + 1.-1 + 1.-1 + 1.0 + 0.0 + 1.0 + 1.1 + 1.1 + 1.1 = 0

$H_{-net4_{24}}$ =1.-1 + 1.-1 + 0.-1 + 0.0 + 1.0 + 0.0 + 1.1 + 1.1 + 0.1 = 0

$H_{-net4_{25}}$ =1.-1 + 0.-1 + 0.-1 + 1.0 + 0.0 + 0.0 + 1.1 + 0.1 + 0.1 = 0

$H_{-net4_{31}}$ =0.-1 + 0.-1 + 1.-1 + 0.0 + 0.0 + 1.0 + 0.1 + 0.1 + 0.1 = -1

$H_{-net4_{32}}$ =0.-1 + 1.-1 + 0.-1 + 0.0 + 1.0 + 1.0 + 0.1 + 0.1 + 0.1 = -1

$H_{-net4_{33}}$ =1.-1 + 0.-1 + 1.-1 + 1.0 + 1.0 + 1.0 + 0.1 + 0.1 + 1.1 = -1

$H_{-net4_{34}}$ =0.-1 + 1.-1 + 0.-1 + 1.0 + 1.0 + 0.0 + 0.1 + 1.1 + 0.1 = 0

$H_{-net4_{35}}$ =1.-1 + 0.-1 + 0.-1 + 1.0 + 0.0 + 0.0 + 1.1 + 0.1 + 0.1 = 0

$H_{-net4_{41}}$ =0.-1 + 0.-1 + 1.-1 + 0.0 + 0.0 + 0.0 + 0.1 + 0.1 + 1.1 = 0

$H_{-net4_{42}}$ =0.-1 + 1.-1 + 1.-1 + 0.0 + 0.0 + 0.0 + 0.1 + 1.1 + 1.1 = 0

$H_{-net4_{43}}$ =1.-1 + 1.-1 + 1.-1 + 0.0 + 0.0 + 1.0 + 1.1 + 1.1 + 1.1 = 0

$H_{-net4_{44}}$ =1.-1 + 1.-1 + 0.-1 + 0.0 + 1.0 + 0.0 + 1.1 + 1.1 + 0.1 = 0

$H_{-net4_{45}}$ =1.-1 + 0.-1 + 0.-1 + 1.0 + 0.0 + 0.0 + 1.1 + 0.1 + 0.1 = 0

$H_{-net4_{51}}$ =0.-1 + 0.-1 + 0.-1 + 0.0 + 0.0 + 1.0 + 0.1 + 0.1 + 0.1 = 0

$H_{-net4_{52}}$ =0.-1 + 0.-1 + 0.-1 + 0.0 + 1.0 + 1.0 + 0.1 + 0.1 + 0.1 = 0

$H_{-net4_{53}}$ =0.-1 + 0.-1 + 1.-1 + 1.0 + 1.0 + 1.0 + 0.1 + 0.1 + 0.1 = -1

$H_{-net4_{54}}$ =0.-1 + 1.-1 + 0.-1 + 1.0 + 1.0 + 0.0 + 0.1 + 0.1 + 0.1 = -1

$H_{-net4_{55}}$ =1.-1 + 0.-1 + 0.-1 + 1.0 + 0.0 + 0.0 + 0.1 + 0.1 + 0.1 = -1

$$H_{-netk4} = \begin{bmatrix} 1 & 1 & 2 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & -1 & -1 \end{bmatrix}$$

Hitung dengan fungsi ReLu (H_j)

$$H_j = f\left(H_{net\,j}\right) = \begin{cases} f(x)=x & jika\ x \geq 0 \\ f(x)=0 & jika\ x < 0 \end{cases}$$

$$H_1 = f(H_{-net2}) = f\left(\begin{bmatrix} -2 & -1 & 0 & 1 & 2 \\ -3 & -2 & 0 & 2 & 3 \\ -2 & -1 & -1 & 1 & 3 \\ -2 & -2 & -1 & 2 & 3 \\ -1 & -1 & -1 & 1 & 2 \end{bmatrix}\right) = \begin{bmatrix} 0 & 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 2 & 3 \\ 0 & 0 & 0 & 1 & 3 \\ 0 & 0 & 0 & 2 & 3 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

$$H_2 = f(H_{-net2}) = f\left(\begin{bmatrix} 2 & 1 & 0 & -1 & -2 \\ 3 & 2 & 0 & -2 & -3 \\ 2 & 1 & 1 & -1 & -3 \\ 2 & 2 & 1 & -2 & -3 \\ 1 & 1 & 1 & -1 & -2 \end{bmatrix}\right) = \begin{bmatrix} 2 & 1 & 0 & 0 & 0 \\ 3 & 2 & 0 & 0 & 0 \\ 2 & 1 & 1 & 0 & 0 \\ 2 & 2 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \end{bmatrix}$$

$$H_3 = f(H_{-net3}) = f\left(\begin{bmatrix} -1 & -1 & -2 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix}\right) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

$$H_4 = f(H_{-net4}) = f\left(\begin{bmatrix} 1 & 1 & 2 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & -1 & -1 \end{bmatrix}\right) = \begin{bmatrix} 1 & 1 & 2 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Proses selanjutnya adalah menyederhanakan hasilnya dengan *max pooling*.

$$P_{j_{m,n}} = \max\left(\left(H_{j_{m,\ n}}\right),\left(H_{j_{m+1,\ n}}\right),\left(H_{j_{m,\ n+1}}\right),\left(H_{j_{m+1,\ n+1}}\right)\right)$$

$P_{1_{1,1}}$ = max(0,0,0,0) = 0
$P_{1_{1,2}}$ = max(0,0,0,0) = 0
$P_{1_{1,3}}$ = max(0,1,0,2) = 2
$P_{1_{1,4}}$ = max(1,2,2,3) = 3
$P_{1_{2,1}}$ = max(0,0,0,0) = 0
$P_{1_{2,2}}$ = max(0,0,0,0) = 0
$P_{1_{2,3}}$ = max(0,2,0,1) = 2
$P_{1_{2,4}}$ = max(2,3,1,3) = 3

$P_{1_{3,1}}$ = max(0,0,0,0) = 0
$P_{1_{3,2}}$ = max(0,0,0,0) = 0
$P_{1_{3,3}}$ = max(0,1,0,2) = 2
$P_{1_{3,4}}$ = max(1,3,2,3) = 3
$P_{1_{4,1}}$ = max(0,0,0,0) = 0
$P_{1_{4,2}}$ = max(0,0,0,0) = 0
$P_{1_{4,3}}$ = max(0,2,0,1) = 2
$P_{1_{4,4}}$ = max(2,3,1,1) = 3

$$P_1 = \begin{bmatrix} 0 & 0 & 2 & 3 \\ 0 & 0 & 2 & 3 \\ 0 & 0 & 2 & 3 \\ 0 & 0 & 2 & 3 \end{bmatrix}$$

$P_{2_{1,1}}$ = max(2,1,3,2) = 3
$P_{2_{1,2}}$ = max(1,0,2,0) = 2
$P_{2_{1,3}}$ = max(0,0,0,0) = 0
$P_{2_{1,4}}$ = max(0,0,0,0) = 0
$P_{2_{2,1}}$ = max(3,2,2,1) = 3
$P_{2_{2,2}}$ = max(2,0,1,1) = 2
$P_{2_{2,3}}$ = max(0,0,1,0) = 1
$P_{2_{2,4}}$ = max(0,0,0,0) = 0

$P_{2_{3,1}}$ = max(2,1,2,2) = 2
$P_{2_{3,2}}$ = max(1,1,2,1) = 2
$P_{2_{3,3}}$ = max(1,0,1,0) = 1
$P_{2_{3,4}}$ = max(0,0,0,0) = 0
$P_{2_{4,1}}$ = max(2,2,1,1) = 2
$P_{2_{4,2}}$ = max(2,1,1,1) = 2
$P_{2_{4,3}}$ = max(1,0,1,0) = 1
$P_{2_{4,4}}$ = max(0,0,0,0) = 0

$$P_2 = \begin{bmatrix} 3 & 2 & 0 & 0 \\ 3 & 2 & 1 & 0 \\ 2 & 2 & 1 & 0 \\ 2 & 2 & 1 & 0 \end{bmatrix}$$

$P_{3_{1,1}}$ = max(0,0,0,0) = 0
$P_{3_{1,2}}$ = max(0,0,0,0) = 0
$P_{3_{1,3}}$ = max(0,0,0,0) = 0
$P_{3_{1,4}}$ = max(0,0,0,0) = 0
$P_{3_{2,1}}$ = max(0,0,1,1) = 1
$P_{3_{2,2}}$ = max(0,0,1,1) = 1
$P_{3_{2,3}}$ = max(0,0,1,0) = 1
$P_{3_{2,4}}$ = max(0,0,0,0) = 0

$P_{3_{3,1}}$ = max(1,1,0,0) = 1
$P_{3_{3,2}}$ = max(1,1,0,0) = 1
$P_{3_{3,3}}$ = max(1,0,0,0) = 1
$P_{3_{3,4}}$ = max(0,0,0,0) = 0
$P_{3_{4,1}}$ = max(0,0,0,0) = 0
$P_{3_{4,2}}$ = max(0,0,0,1) = 1
$P_{3_{4,3}}$ = max(0,0,1,1) = 1
$P_{3_{4,4}}$ = max(0,0,1,1) = 1

$$P_3 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix}$$

$P4_{1,1}$ = max(1,1,0,0) = 1
$P4_{1,2}$ = max(1,2,0,0) = 2
$P4_{1,3}$ = max(2,1,0,0) = 2
$P4_{1,4}$ = max(1,1,0,0) = 1
$P4_{2,1}$ = max(0,0,0,0) = 0
$P4_{2,2}$ = max(0,0,0,0) = 0
$P4_{2,3}$ = max(0,0,0,0) = 0
$P4_{2,4}$ = max(0,0,0,0) = 0
$P4_{3,1}$ = max(0,0,0,0) = 0
$P4_{3,2}$ = max(0,0,0,0) = 0
$P4_{3,3}$ = max(0,0,0,0) = 0
$P4_{3,4}$ = max(0,0,0,0) = 0
$P4_{4,1}$ = max(0,0,0,0) = 0
$P4_{4,2}$ = max(0,0,0,0) = 0
$P4_{4,3}$ = max(0,0,0,0) = 0
$P4_{4,4}$ = max(0,0,0,0) = 0

$$P_4 = \begin{bmatrix} 1 & 2 & 2 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Menggabungkan matriks konvolusi menjadi satu kesatuan dengan cara *flattening*.

$x_1 = 0$	$x_2 = 0$	$x_3 = 2$	$x_4 = 3$	$x_5 = 0$	$x_6 = 0$	$x_7 = 2$	$x_8 = 3$
$x_9 = 0$	$x_{10} = 0$	$x_{11} = 2$	$x_{12} = 3$	$x_{13} = 0$	$x_{14} = 0$	$x_{15} = 2$	$x_{16} = 3$
$x_{17} = 3$	$x_{18} = 2$	$x_{19} = 0$	$x_{20} = 0$	$x_{21} = 3$	$x_{22} = 2$	$x_{23} = 1$	$x_{24} = 0$
$x_{25} = 2$	$x_{26} = 2$	$x_{27} = 1$	$x_{28} = 0$	$x_{29} = 2$	$x_{30} = 2$	$x_{31} = 1$	$x_{32} = 0$
$x_{33} = 0$	$x_{34} = 0$	$x_{35} = 0$	$x_{36} = 0$	$x_{37} = 1$	$x_{38} = 1$	$x_{39} = 1$	$x_{40} = 0$
$x_{41} = 1$	$x_{42} = 1$	$x_{43} = 1$	$x_{44} = 0$	$x_{45} = 0$	$x_{46} = 1$	$x_{47} = 1$	$x_{48} = 1$
$x_{49} = 1$	$x_{50} = 2$	$x_{51} = 2$	$x_{52} = 1$	$x_{53} = 0$	$x_{54} = 0$	$x_{55} = 0$	$x_{56} = 0$
$x_{57} = 0$	$x_{58} = 0$	$x_{59} = 0$	$x_{60} = 0$	$x_{61} = 0$	$x_{62} = 0$	$x_{63} = 0$	$x_{64} = 0$

4.3 Skenario Pengujian

Skenario pengujian membahas pengujian yang akan digunakan. Pengujian ditunjukkan untuk dapat menemukan model terbaik yang hasilnya dapat dievaluasi.

4.3.1 Skenario Pengujian Loss dan Akurasi CNN

Proses pengujian *loss* dan akurasi akan membandingkan berbagai hasil dari bermacam-macam model CNN. Hasil akurasi dan *loss* akan terlihat dalam proses pelatihan pada setiap *epoch*-nya.

Tabel 4.2 Skenario Pengujian *Loss* dan Akurasi

Model	EP1	EP2	EP3	EP4	EP5	EP6	...	EPn
1								
2								
3								
4								
n								

4.3.2 Skenario Pengujian CNN pada video

Proses pengujian akurasi pada video dilakukan dengan *random sampling* terhadap sebagian *frame* untuk dihitung secara manual akurasinya. Model CNN yang diuji adalah model dengan hasil paling optimal pada pengujian sebelumnya.

Tabel 4.3 Skenario pengujian *random sampling* video.

Nama Percobaan	Prediksi Tepat	Prediksi Salah	Akurasi
Percobaan 1			
Percobaan 2			
Percobaan n			

BAB 5 IMPLEMENTASI

Bab ini menunjukkan proses implementasi sistem, yang terdiri dari penjelasan tentang lingkungan ker sistem dan implementasi dari kode sumber beserta penjelasannya.

5.1 Lingkungan Implementasi

Subbab ini menjelaskan berbagai lingkungan yang digunakan untuk implementasi sistem. Terdiri dari 2 macam lingkungan, yaitu lingkungan perangkat keras dan perangkat lunak yang digunakan untuk proses implementasi sistem.

5.1.1 Lingkungan Perangkat Keras

Pada Tabel 5.1 menunjukkan spesifikasi lingkungan perangkat keras yang digunakan dalam implementasi sistem.

Tabel 5.1 Lingkungan Perangkat Keras

Nama Hardware	Keterangan
Prosesor	Intel(R) Core(TM) i3-8145U CPU @ 2.30GHZ
RAM	8 GB RAM LPDDR3L
Storage	512GB SSD
VGA	IntelHD 620
Smartphone	Xperia XZ1 (Kamera 19MP) Smartphone digunakan untuk mengambil foto wajah.

5.1.2 Subbab Lima Satu Dua

Dalam Tabel 5.2 menunjukkan perangkat lunak yang digunakan dalam implementasi sistem.

Tabel 5.2 Lingkungan Perangkat Lunak

Nama Software	Keterangan
Windows 10 64 bit	Sistem operasi pada komputer
Python v3.7.4	Bahasa pemrograman dalam implementasi sistem
Microsoft Word 2019	Aplikasi pengolah dokumen untuk membuat laporan penelitian
Microsoft Excel 2019	Aplikasi pengolah angka untuk membuat perhitungan manualisasi

5.2 Implementasi Algoritme *Library*

Pada subbab ini merupakan proses implementasi dari *algoritme* berupa kode sumber disertai dengan penjelasan dan pembahasannya. Modul yang digunakan pada sub-bab ini adalah *dlib*, *tensorflow* dan *keras*. Modul *dlib* digunakan untuk ekstraksi fitur *facial landmark* pada wajah. Modul Keras merupakan *library* yang dipakai untuk proses *deep learning*. Secara garis besar, prosesnya terbagi menjadi 3 yaitu proses ekstraksi fitur mata dalam wajah, dilanjutkan dengan augmentasi data, kemudian penentuan dari model dari arsitektur CNN yang akan digunakan dan terakhir proses pembelajaran.

5.2.1 Implementasi Ekstraksi Citra Mata

Ekstraksi citra mata merupakan langkah *pre-processing* yang dilakukan karena penelitian ini difokuskan pada bagian mata saja. Pada bagian ini akan ditunjukkan kode program untuk melakukan ekstraksi gambar mata dalam citra suatu wajah dengan menggunakan *facial landmark*.

Tabel 5.3 Implementasi Extraksi Citra Mata

no	Source code
1	`import cv2`
2	`import dlib`
3	`import os`
4	`listatas = os.listdir('dataset1/atas')`
5	`listbawah = os.listdir('dataset1/bawah')`
6	`listkanan = os.listdir('dataset1/kanan')`
7	`listkiri = os.listdir('dataset1/kiri')`
8	`listtengah = os.listdir('dataset1/tengah')`
9	`detector=dlib.get_frontal_face_detector()`
10	`predictor=dlib.shape_predictor ("shape_predictor_5_face_landmarks.dat")`
11	`def eyeextract(pathinput,pathoutput):`
12	`gbr=cv2.imread(pathinput)`
13	`gray = cv2.cvtColor(gbr, cv2.COLOR_BGR2GRAY)`
14	`faces = detector(gray)`
15	`if (len(faces)>0):`
16	`landmarks = predictor(gray,faces[0])`
17	`#matakanan`
18	`x1=landmarks.part(3).x;`
19	`x2=landmarks.part(2).x;`
20	`y1=landmarks.part(3).y-int((landmarks.part(2).x-`
21	`landmarks.part(3).x)/2)`
22	`y2=landmarks.part(2).y+int((landmarks.part(2).x-`
23	`landmarks.part(3).x)/2)`
24	`matakanan = gbr[y2:y1, x2:x1]`
25	`if (matakanan.size>0):`
26	`matakanan = cv2.resize(matakanan, (32,32),`
27	`interpolation=cv2.INTER_CUBIC)`
28	`cv2.imwrite(pathoutput+'1.jpg',matakanan)`
29	`#matakiri`
30	`x1=landmarks.part(1).x;`
31	`x2=landmarks.part(0).x;`
32	`y1=landmarks.part(1).y-int((landmarks.part(0).x-`
33	`landmarks.part(1).x)/2)`

```
34          y2=landmarks.part(0).y+int((landmarks.part(0).x-
35                                      landmarks.part(1).x)/2)
36        matakiri = gbr[y1:y2, x1:x2]
37        if (matakiri.size>0):
38            matakiri = cv2.resize(matakiri, (32,32),
39                                  interpolation= cv2.INTER_CUBIC)
40            cv2.imwrite(pathoutput+'2.jpg',matakiri)
41 for i in range(len(listatas)):
42     eyeextract('dataset1/atas/'+listatas[i],
43                'dataset1-mata/atas/'+listatas[i][0:-4])
44 for i in range(len(listbawah)):
45     eyeextract('dataset1/bawah/'+listbawah[i],
46                'dataset1-mata/bawah/'+listbawah[i][0:-4])
47 for i in range(len(listkanan)):
48     eyeextract('dataset1/kanan/'+listkanan[i],
49                'dataset1-mata/kanan/'+listkanan[i][0:-4])
50 for i in range(len(listkiri)):
51     eyeextract('dataset1/kiri/'+listkiri[i],
52                'dataset1-mata/kiri/'+listkiri[i][0:-4])
53 for i in range(len(listtengah)):
54     eyeextract('dataset1/tengah/'+listtengah[i],
55                'dataset1-mata/tengah/'+listtengah[i][0:-4])
```

Berdasarkan Tabel 5.3 baris 1-3 merupakan inisialisasi *library* yang dibutuhkan. Baris 4-8 membaca daftar nama file citra wajah. Baris 9 merupakan inisialisasi modul *face detection*. Baris 10 adalah inisialisasi *facial landmark 5 point*. Baris 11 inisialisasi prosedur ekstraksi mata. Baris 12 pembacaan file citra wajah. Baris 13 mengubah citra dari RGB ke grayscale. Baris 14 dilakukan proses deteksi wajah. Baris 15 jika wajah ditemukan, lanjutkan proses berikutnya. Baris 16 melakukan *5 point facial landmark* terhadap wajah yang sudah ditemukan. Baris 18-23 menyimpan posisi koordinat mata kanan. Baris 24 melakukan crop terhadap mata kanan. Baris 25 jika mata kanan berhasil di-*crop* maka dilanjukkan dengan Baris 26-27 yaitu proses *resize* citra mata. Baris 28 menyimpan hasil citra mata kanan. Baris 30-35 menyimpan posisi koordinat mata kiri. Baris 36 melakukan *crop* terhadap mata kiri. Baris 37 jika mata kiri berhasil di-*crop* maka dilanjukkan dengan Baris 38-39 yaitu proses resize citra mata. Baris 40 menyimpan hasil citra mata kiri. Baris 41-42 adalah perulangan ekstraksi mata pada setiap folder dimana citra wajah disimpan

5.2.2 Implementasi Augmentasi Data

Augmentasi data merupakan proses pengolahan data sebelum dilakukan proses pembelajaran. Proses ini untuk memperbanyak jumlah data dengan mengubah tingkat kecerahan pada data citra masukkan, melakukan normalisasi data, dengan menentukan ukuran *batch size*.

Tabel 5.4 Implementasi Augmentasi Data

no	Source code
1	img_width, img_height = 32, 32
2	batch_size = 32
3	train_datagen = ImageDataGenerator(
4	rescale=1. / 255,

```
5       brightness_range=(0.1,0.9),
6       horizontal_flip=False,vertical_flip=False)
7   test_datagen = ImageDataGenerator(
8       rescale=1. / 255,
9       brightness_range=(0.1,0.9),
10      horizontal_flip=False,vertical_flip=False)
11  train_generator = train_datagen.flow_from_directory(
12      'dataset1-train/training',
13      target_size=(img_height, img_width),
14      batch_size=batch_size,
15      class_mode='sparse')
16  validation_generator = test_datagen.flow_from_directory(
17      'dataset1-train/testing',
18      target_size=(img_height, img_width),
19      batch_size=batch_size,
20      class_mode='sparse')
```

Berdasarkan Tabel 5.4 pada baris 1-2 merupakan inisialisasi parameter yang menunjukkan dimensi citra, sedangkan batch_size murupakan jumlah literasi yang diperlukan dalam pelatihan. Baris 3-16 adalah proses augmentasi pada data latih dengan parameter manipulasi brightness dan normalisasi. Baris 7-10 adalah proses augmentasi pada data *testing* dengan parameter manipulasi *brightness* dan normalisasi. Baris 11-15 melakukan eksekusi generator data terhadap data latih sebanyak batch_size. Baris 16-20 melakukan eksekusi generator data terhadap data testing sebanyak *batch_size*.

5.2.3 Implementasi Penentuan Arsitektur CNN

Bagian ini menunjukkan kode program dalam pembentukan arsitektur dari CNN yang terdiri dari 3 layer, yaitu layer konvolusi, layer *pooling* dan layer *fully connected.*

Tabel 5.5 Implementasi Arsitektur CNN

no	Source code
1	`model = Sequential()`
2	`model.add(Convolution2D(16,3,3,` `input_shape=(img_width,img_height, 3)))`
3	`model.add(Activation("relu"))`
4	`model.add(MaxPooling2D(pool_size=(2, 2)))`
5	`model.add(Convolution2D(32, 3, 3))`
6	`model.add(Activation("relu"))`
7	`model.add(MaxPooling2D(pool_size=(2, 2)))`
8	`model.add(Flatten())`
9	`model.add(Dense(output_dim = 20, activation="relu"))`
10	`model.add(Dense(output_dim =5, activation="softmax"))`
11	`model.summary()`
12	`model.compile(loss='sparse_categorical_crossentropy',`
13	`optimizer=optimizers.adam(),`
14	`metrics=['accuracy'])`

Pada Tabel 5.5 baris 1, dilakukan proses inisialisasi model CNN. Baris 2-7 menambahkan arsitektur yang dipakai dalam CNN, yang menentukan kedalamannya. Terdiri dari konvolusi, aktivasi ReLu dan proses pooling. Baris 8 menunjukkan proses *flattening*, yaitu mengubah matrix hasil konvolusi menjadi 1 dimensi dan menjadikannya *fully connected*. Baris 9-10 menambahkan *fully connected layer* dengan kedalaman 2, masing-masing mewakili jumlah neuron dalam jaringan saraf. Baris 11 adalah perintah untuk menampilkan model arsitektur CNN yang sudah dibuat. Baris 12-14 melakukan *compile* arsitektur CNN yang sudah dibuat dengan memilih optimasi yang akan digunakan

5.2.4 Implemestasi Proses Pembelajaran CNN

Pada bagian ini akan ditunjukkan kode program mengenai proses dan variabel yang digunakan pada pelatihan.

Tabel 5.6 Implementasi Pelatihan CNN

no	Source code
1	`model.fit_generator(`
2	`    train_generator,`
3	`    steps_per_epoch=batch_size,`
4	`    validation_data=validation_generator,`
5	`    validation_steps=batch_size,`
6	`    epochs=10)`
7	`model.save('model.h5')`
8	`model.get_weights()`
9	`model.save_weights('weights.h5')`

Pada Tabel 5.6 baris 1, menjalankan fungsi *fit_generator* yang merupakan moduk untuk pelatihan, baris 2 menunjukkan input data berupa citra hasil dari augmentasi data yang sudah dinormalisasi, baris 3 *step_per_epoch* merupakan banyaknya langkah pelatihan tiap epochnya. Baris 4-5 merupakan proses validasi terhadap hasil pelatihan. Baris 6 *epoch* adalah jumlah literasi yg dilakukan dalam pelaksanaan pelatihan dan validasi. Baris 7-9 melakukan penyimpanan hasil akhir dari konfigurasi model beserta bobot akhir hasil pelatihan.

5.2.5 Implementasi Pengujian *Random Sampling* Video

Bagian ini menunjukkan kode program yang digunakan untuk pengujian hasil dari konfigurasi model dan bobot terhadap *frame* acak dalam sebuah *video testing.*

Tabel 5.7 Implementasi Pengujian *Random Sampling* Video

no	Source code
1	`model = load_model('model.h5')`
2	`model.load_weights('weights.h5')`
3	`def predict(x):`
4	`  x = np.expand_dims(x, axis=0)`
5	`  array = model.predict(x)`
6	`  result = array[0]`
7	`  hasil = np.argmax(result)`
8	`  if hasil == 0:`

```
    print("Atas")
  elif hasil == 1:
    print("Bawah")
  elif hasil == 2:
    print("Kanan")
  elif hasil == 3:
    print("Kiri")
  elif hasil == 4:
    print("Tengah")
  return hasil
cap = cv2.VideoCapture('video-testing.mp4')
length = int(cv2.VideoCapture.get(cap,
    int(cv2.CAP_PROP_FRAME_COUNT)))
daftarrandom = []
for I in range(0,51):
    n = random.randint(0,length)
    daftarrandom.append(n)
print(daftarrandom)
detector=dlib.get_frontal_face_detector()
predictor=dlib.shape_predictor(
"shape_predictor_5_face_landmarks.dat")
while cap.isOpened():
    _, frame = cap.read()
    if not _:
        time.sleep(2)
        break
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    start = time.time()
    faces = detector(gray)
    if (len(faces)>0):
        x1 = faces[0].left()
        y1 = faces[0].top()
        x2 = faces[0].right()
        y2 = faces[0].bottom()
        cv2.rectangle(frame, (x1, y1), (x2, y2),
                      (0, 255, 0), 1)
        landmarks = predictor(gray,faces[0])
        for n in range(0, 5):
            x = landmarks.part(n).x
            y = landmarks.part(n).y
            cv2.circle(frame, (x, y), 3, (255, 0, 255), -1)
        #matakanan
        x1=landmarks.part(3).x;
        x2=landmarks.part(2).x;
        y1=landmarks.part(3).y-int((landmarks.part(2).x-
                                    landmarks.part(3).x)/2);
        y2=landmarks.part(2).y+int((landmarks.part(2).x-
                                    landmarks.part(3).x)/2);
        cv2.rectangle(frame, (x1, y1), (x2, y2),
                      (255, 255, 0), 1)
        matakanan = frame[y2:y1, x2:x1]
        #matakiri
        x1=landmarks.part(1).x;
        x2=landmarks.part(0).x;
        y1=landmarks.part(1).y-int((landmarks.part(0).x-
                                    landmarks.part(1).x)/2);
```

```
63          y2=landmarks.part(0).y+int((landmarks.part(0).x-
64                                      landmarks.part(1).x)/2);
65          cv2.rectangle(frame, (x1, y1), (x2, y2),
66                        (255, 255, 0), 1)
67          matakiri = frame[y2:y1, x2:x1]
68          result=predict(cv2.resize(matakanan, dsize=(32, 32),
69                         interpolation=cv2.INTER_CUBIC))
70          font = cv2.FONT_HERSHEY_SIMPLEX
71          cv2.putText(frame,str(round(1/(time.time()-start)))+
72            ` fps (`+str(round((time.time()-start)*1000))+
73            ` ms)',(10,20),font,0.5,(255,255,255),1,
74  cv2.LINE_AA)
75          if (result==0):
76              cv2.putText(frame,'Atas',(10,70),
77                          font, 1,(0,255,255),2,cv2.LINE_AA)
78          elif (result==1):
79              cv2.putText(frame,'Bawah',(10,70),
80                          font, 1,(0,255,255),2,cv2.LINE_AA)
81          elif (result==2):
82              cv2.putText(frame,'Kanan',(10,70),
83                          font, 1,(0,255,255),2,cv2.LINE_AA)
84          elif (result==3):
85              cv2.putText(frame,'Kiri',(10,70),
86                          font, 1,(0,255,255),2,cv2.LINE_AA)
87          elif (result==4):
88              cv2.putText(frame,'Tengah',(10,70),
89                          font, 1,(0,255,255),2,cv2.LINE_AA)
90          print((time.time() - start))
91          if int(cap.get(cv2.CAP_PROP_POS_FRAMES)) in daftarrandom:
92              cv2.imwrite(`random-test/0/'+
93  str(cap.get(cv2.CAP_PROP_POS_FRAMES))+
94                          `.jpg',frame)
95      cv2.imshow("Frame", frame)
```

Pada Tabel 5.7 baris 1-2, inisialisasi model CNN hasil pelatihan. Baris 8-18 adalah fungsi untuk mengeluarkan hasil prediksi dengan input citra mata dan output adalah hasil klasifikasi. Baris 19-21 membuka file video dan menghitung total frame. Baris 22-26 membuat nilai random antara 0 sampai jumlah *frame* sebanyak 50 buah. No 27 merupakan inisialisasi modul *face detection*. Baris 28-29 adalah inisialisasi *facial landmark 5 point*. Baris 30-31 inisialisasi pembacaan *frame* video citra wajah. Baris 35 mengubah citra dari RGB ke *grayscale*. Baris 37 dilakukan proses deteksi wajah. Baris 38 jika wajah ditemukan, lanjutkan proses berikutnya. Baris 39-44 membuat kotak penanda wajah yang ditemukan pada citra. Baris 45 melakukan *5 point facial landmark* terhadap wajah yang sudah ditemukan. Baris 46-49 membuat tanda pada *landmark* yang ditemukan. Baris 50-57 menyimpan posisi koordinat mata kanan. Baris 58 melakukan *crop* terhadap mata kanan. Baris 59-66 menyimpan posisi koordinat mata kiri. Baris 67 melakukan *crop* terhadap mata kiri. Baris 68-69 melakukan prediksi pada gerakan mata kanan. Baris 70-89 meluliskan hasil prediksi pada gambar. Baris 91-94 jika *frame* sekarang termasuk dalam *frame* yang akan dilakukan sampling, maka disimpan gambarnya. Baris 95 menampilkan *output frame* pada layar.

BAB 6 PENGUJIAN DAN ANALISIS

Pengujian sistem dilakukan berdasarkan skenario pengujian yang dibuat sebelumnya. Masing-masing hasil dari pengujian dilakukan analisa dan disimpulkan hasilnya.

6.1 Pengujian *Loss* dan Akurasi pada Data

Pengujian ini membandingkan *loss* dan akurasi antara model CNN yang sudah dibuat. Pengujian *loss* dan akurasi dapat juga digunakan untuk mengetahui di mana sistem menghasilkan akurasi terbaik dengan waktu tercepat. Pengujian ini menggunakan input citra sejumlah 160 yang dikelompokkan ke dalam 5 kelas arah mata. Dengan pembagian 120 citra yang digunakan sebagai data latih dan 40 citra sisanya sebagai data uji. Model CNN yang diujikan dapat dilihat pada Tabel 6.1.

Tabel 6.1 Model CNN

Nama Model	*Batch size*	Filter 1	Filter 2	*Fully connected*
Model 1	8	32	64	20
Model 2	8	32	64	10
Model 3	8	16	32	20
Model 4	8	16	32	10
Model 5	16	32	64	20
Model 6	16	32	64	10
Model 7	16	16	32	20
Model 8	16	16	32	10
Model 9	32	32	64	20
Model 10	32	32	64	10
Model 11	32	16	32	20
Model 12	32	16	32	10

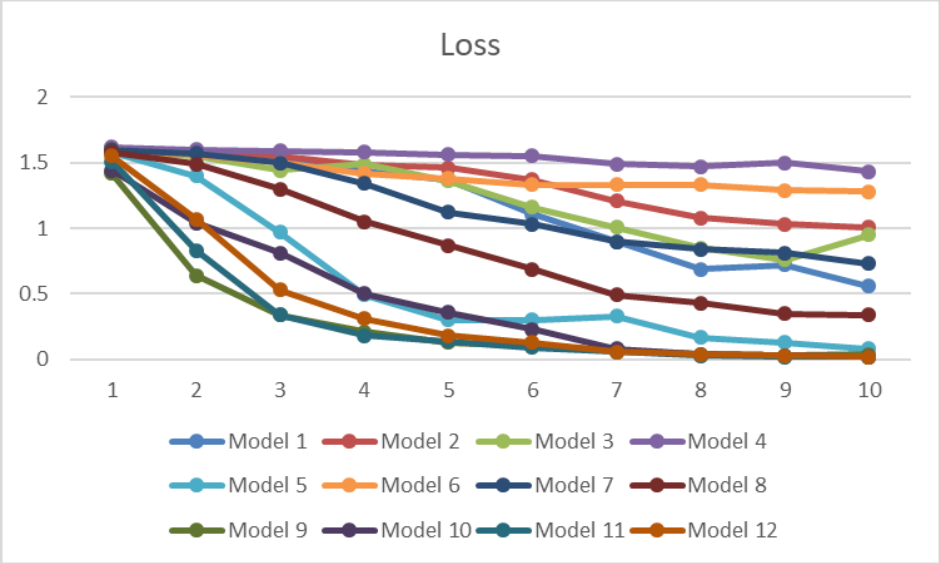
Berdasarkan Tabel 6.1 ukuran batch size yang akan diuji adalah 8,16,32. Jumlah Filter yang digunakan adalah 32 dengan 64 dan 16 dengan 32. Jumlah neuron *hidden layer* dalam *fully connected* yang digunakan 20 dan 10. *Epoch* yang digunakan dalam pengujian ini berjumlah 10. Jumlah kedalaman arsitektur CNN berjumlah 2. Tabel 6.2 menunjukkan hasil perekaman *loss* dari proses pembelajaran sistem.

Tabel 6.2 Hasil Perekaman Nilai Loss

Model	EP1	EP2	EP3	EP4	EP5	EP6	EP7	EP8	EP9	EP10
1	1.60	1.56	1.52	1.43	1.37	1.11	0.90	0.69	0.72	0.56
2	1.61	1.59	1.55	1.48	1.46	1.37	1.21	1.08	1.03	1.01
3	1.59	1.55	1.44	1.49	1.36	1.16	1.01	0.85	0.76	0.95
4	1.62	1.60	1.59	1.58	1.56	1.55	1.49	1.47	1.50	1.43
5	1.58	1.40	0.97	0.49	0.30	0.30	0.33	0.17	0.13	0.08
6	1.60	1.56	1.50	1.42	1.38	1.33	1.33	1.33	1.29	1.28
7	1.60	1.57	1.50	1.34	1.12	1.03	0.90	0.84	0.81	0.73
8	1.58	1.49	1.30	1.05	0.87	0.69	0.49	0.43	0.35	0.34

Tabel 6.3 Hasil Perekaman Nilai Loss (lanjutan)

Model	EP1	EP2	EP3	EP4	EP5	EP6	EP7	EP8	EP9	EP10
9	1.42	0.64	0.34	0.21	0.13	0.11	0.07	0.04	0.03	0.04
10	1.44	1.04	0.81	0.50	0.36	0.23	0.08	0.04	0.03	0.02
11	1.51	0.83	0.34	0.18	0.14	0.09	0.06	0.03	0.02	0.02
12	1.55	1.07	0.53	0.31	0.18	0.13	0.06	0.04	0.03	0.02



Gambar 6.1 Grafik Loss pada Tiap Model

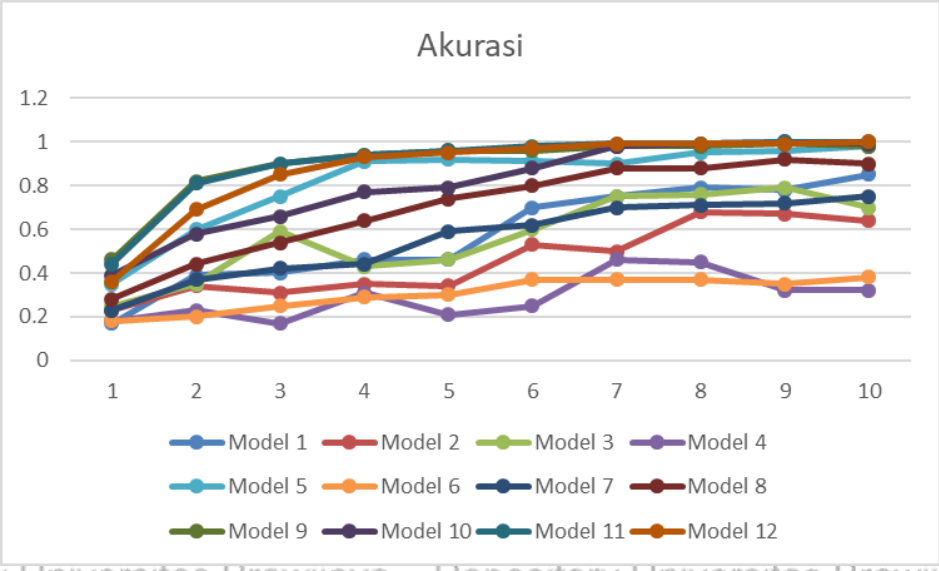
Berdasarkan Tabel 6.1, Tabel 6.2 dan Gambar 6.1 dapat diketahui bahwa ukuran *batch size* sangat memengaruhi hasil rata-rata nilai *loss* pada setiap model. Jika dirata-rata penggunaan *batch size* 8 menghasilkan nilai *loss* tertinggi, dengan *batch size* 16 hasil menjadi lebih baik, dan *batch size* 32 yang memberikan nilai loss terkecil. *Batch size* dalam penelitian ini menentukan proses augmentasi citra, jika semakin besar nilainya maka jumlah gambar yang digunakan pelatihan dan pengujian secara otomatis menjadi semakin besar, karena data bisa lebih beragam. Penggunaan *batch size* sangat mempengaruhi waktu komputasi yang diperlukan untuk proses learning seperti yang ditunjukkan pada Tabel 6.4. Selain *batch size*, perbedaan jumlah filter juga memengaruhi nilai loss, jumlah filter konvolusi yang lebih besar menghasilkan nilai *loss* yang lebih kecil. Ini dikarenakan lebih banyak data yang bisa terekam oleh sistem. Model 5, Model 9, Model 10, Model 11 dan Model 12 adalah model yang memiliki nilai *loss* dibawah 0.1, dan bisa dibuktikan pula dengan nilai akurasi pada Gambar 6.2.

Tabel 6.4 Hasil Perekaman Nilai Akurasi

Model	EP1	EP2	EP3	EP4	EP5	EP6	EP7	EP8	EP9	EP10
1	0.17	0.39	0.40	0.46	0.46	0.70	0.75	0.79	0.78	0.85
2	0.23	0.34	0.31	0.35	0.34	0.53	0.50	0.68	0.67	0.64
3	0.25	0.35	0.59	0.43	0.46	0.60	0.75	0.76	0.79	0.70
4	0.18	0.23	0.17	0.31	0.21	0.25	0.46	0.45	0.32	0.32
5	0.35	0.60	0.75	0.91	0.92	0.91	0.90	0.95	0.96	0.98

Tabel 6.5 Hasil Perekaman Nilai Akurasi (lanjutan)

Model	EP1	EP2	EP3	EP4	EP5	EP6	EP7	EP8	EP9	EP10
6	0.18	0.20	0.25	0.29	0.30	0.37	0.37	0.37	0.35	0.38
7	0.23	0.37	0.42	0.44	0.59	0.62	0.70	0.71	0.72	0.75
8	0.28	0.44	0.54	0.64	0.74	0.80	0.88	0.88	0.92	0.90
9	0.46	0.82	0.90	0.94	0.96	0.96	0.98	0.98	0.99	0.98
10	0.39	0.58	0.66	0.77	0.79	0.88	0.98	0.99	1.00	0.99
11	0.44	0.81	0.90	0.94	0.96	0.98	0.99	0.99	1.00	1.00
12	0.36	0.69	0.85	0.93	0.95	0.97	0.99	0.99	0.99	1.00



Gambar 6.2 Grafik Akurasi Setiap Model

Gambar 6.2 dan Tabel 6.3 menunjukkan semua model yang menggunakan batch size 32 memiliki akurasi di atas 0,95. Hanya model 5 yang termasuk diatas 0,95 akurasi yang menggunakan *batch size* 16. Berdasarkan evaluasi nilai *loss*, akurasi dan waktu komputasi yang diperlukan dengan 120 data latih dan 40 data uji yang tersusun dari 5 kelas. Arsitektur terbaik adalah Model 5, yang dengan parameter 16 batch size, 32 filter untuk konvolusi layer 1 dan 64 filter untuk konvolusi layer dengan *fully connected* layer berjumlah 20.

Tabel 6.6 Waktu komputasi proses *learning*

Model	Waktu komputasi
Model 1	3,5 detik
Model 2	3,78 detik
Model 3	2,84 detik
Model 4	2,99 detik
Model 5	8,62 detik
Model 6	8,59 detik
Model 7	8 detik
Model 8	7,66 detik

Tabel 6.7 Waktu komputasi proses *learning* (lanjutan)

Model	Waktu komputasi
Model 9	26,2 detik
Model 10	25,91 detik
Model 11	24,51 detik
Model 12	24,36 detik

6.2 Pengujian Akurasi pada Video

Pengujian ini bertujuan untuk mengetahui hasil kemampuan sistem dalam melakukan prediksi dalam dunia nyata. *File* video yang digunakan berdurasi 101 detik yang terdiri dari 3049 frame. Di dalam video tersebut terdapat 10 orang yang menggerakkan matanya dalam berbagai kondisi. Pengujuian dilakukan dengan menggunakan *random sampling* sebanyak 50 frame dari keseluruhan dan diulang sebanyak tiga kali. Model CNN yang dipilih digunakan pada pengujian ini adalah Model 5. Hasil dari pengujian Model 5 terhadap *random sampling* pada video ditunjukkan oleh Tabel 6.5.

Tabel 6.8 Hasil akurasi *random sampling* video.

Nama Percobaan	Prediksi Tepat	Prediksi Salah	Akurasi
Percobaan 1	48	2	0,96
Percobaan 2	48	2	0,96
Percobaan 3	47	3	0,94

Dari hasil pengujian pada Tabel 6.5, pada percobaan pertama dan kedua algoritme berhasil mengklasifikasikan dengan benar 48 dari 50 citra, dan percobaan ketiga berhasil mengklasifikasikan 47 dari 50 citra, sehingga nilai rata-rata akurasi dari ketiga percobaan di atas adalah 0,953. Contoh gambar hasil klasifikasi yang benar ditunjukkan oleh Gambar 6.3 sedangkan hasil klasifikasi yang salah ditunjukkan oleh Gambar 6.4.

Gambar 6.3 Contoh Hasil Klasifikasi Arah Mata yang Benar

Gambar 6.4 Contoh Hasil Klasifikasi Arah Mata yang Salah

BAB 7 PENUTUP

7.1 Kesimpulan

Berdasarkan hasil dari pengujian dan analisis yang sudah dilakukan, kesimpulan hasil yang didapatkan adalah dalam membangun arsitektur model *Convolution Neural Network* untuk klasifikasi arah mata membutuhkan 2 layer konvolusi dengan 32 *filter* untuk *layer* 1 dan 64 *filter* untuk layer 2. Menggunakan *batch size* sejumlah 16 dalam augmentasi citra dengan 20 *fully connected* menghasilkan nilai loss 0,08, dengan akurasi 0,98 dan waktu pelatihan 8.62 detik. Model tersebut kemudian dilakukan uji terhadap video yang diambil 50 *frame* secara acak sebanyak tiga kali, menghasilkan rata-rata akurasi sebesar 0,95.

7.2 Saran

Penelitian ini jauh dari sempurna dan masih banyak yang perlu ditingkatkan. Diharapkan semoga penelitian yang akan datang bisa dilakukan memperbaiki supaya menjadi lebih baik. Adapun saran yang dapat berguna untuk penelitian selanjutnya adalah diharapkan untuk penelitian berikutnya jumlah *dataset* diperbanyak dengan segala kondisi pencahayaan, arah, jarak dan sudut pengambilan karena dalam penelitian ini, citra wajah untuk pelatihan dan pengujian jumlahnya sangat terbatas.

DAFTAR REFERENSI

Abhirawa, Halprin, Jondri dan Anditya Arifianto. 2017. *Pengenalan Wajah Menggunakan Convolutional Neural Network*. Bandung : Universitas Telkom.

Ahmad Hania, Abu., 2017. Mengenal Artificial Intelligence, Machine Learning, Neural Network, dan Deep Learning. *Jurnal Teknologi Indonesia*.

Atturmudzi, Aji., 2008. *Eye Detection Menggunakan Template Matching, Algoritma Genetika, dan Hough Transform*. Bandung: Institut Teknologi Telkom.

A. Benoit, A. Caplier, and L. Bonnaud, 2005. Gaze direction estimation tool based on head motion analysis or iris position estimation. In 2005 13th European Signal Processing Conference (pp. 1-4).

B. Kroon, S. Maas, S. Boughorbel, and A. Hanjalic., 2009. Eye localization in low and standard definition content with application to face matching. *Comput. Vis. Image Underst.*, vol. 113, no. 8, pp. 921–933,.

Danukusumo, Kefin Pudi., 2017. *Implementasi Deep Learning Menggunakan Convolutional Neural Network Untuk Klasifikasi Citra Candi Berbasis GPU*. Yogyakarta : Universitas Atma Jaya.

Duchowski, A., 2002. A breadth-first survey of eye-tracking applications. *Behavior Research Methods, Instruments, & Computers*, 34(4), pp.455-470.

Hammal, Z., Massot, C., Bedoya, G. and Caplier, A., 2005. Eyes Segmentation Applied to Gaze Direction and Vigilance Estimation. In: Pattern Recognition and Image Analysis. [online] Springer Berlin Heidelberg.pp.236–246. Available at: <http://dx.doi.org/10.1007/11552499_27>.

Kazemi, Vahid. Sullivan, Josephine., 2014. *One Millisecond Face Alignment with an Ensemble of Regression Trees*. Stockholm : KTH Royal Institute of Technology.

Kim, K.-N., Ramakrishna, R.S., 1999. Vision-based Eyegaze Tracking for Human Computer Interface. In: *IEEE Int. Conf. On Systems, Man, and Cybernetics*, vol. 2, pp. 324–329

Matsumoto, Y., Zelinsky, A., 2000. An algorithm for real-time stereo vision implementation of head pose and gaze direction measurement. In: *Proceedings of IEEE fourth Int. Conf. on Faze and Gesture Recognition*, pp. 499–505

N. Edenborough, R. Hammoud, A. Harbach, A. Ingold, B. Kisacanin, P. Malawey, T. Newman, G. Scharenbroch, S. Skiver, M. Smith, A. Wilhelm, G. Witt, E. Yoder, and H. Zhang., 2005. Driver state monitor from DELPHI. In 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) (pp. 1206-1207 vol. 2).

Ohtera, R., Horiuchi, T., Kotera, H., 2006. *Eye-gaze Detection from Monocular Camera Image Based on Physiological Eyeball Models*. In: IWAIT2006. Proc. International Workshop on Advanced Image Technology, pp. 639–664

Ohtera, R., Horiuchi, T., Tominaga, S., 2007. *Eye-gaze Detection from Monocular Camera Image Using Parametric Template matching*. In: ACCV 2007. Part 1, LNCS, pp. 708-717

Pavlin-Premrl, D., Waterston, J., McGuigan, S., Infeld, B., Sultana, R., O'Sullivan, R. and Gerraty, R.P., 2015. Importance of spontaneous nystagmus detection in the differential diagnosis of acute vertigo. Journal of Clinical Neuroscience, [online] 22(3), pp.504–507. Available at: <http://dx.doi.org/10.1016/j.jocn.2014.09.011>.

Putra, D., 2010. *Pengolahan Citra Digital*. Yogyakarta: ANDI.

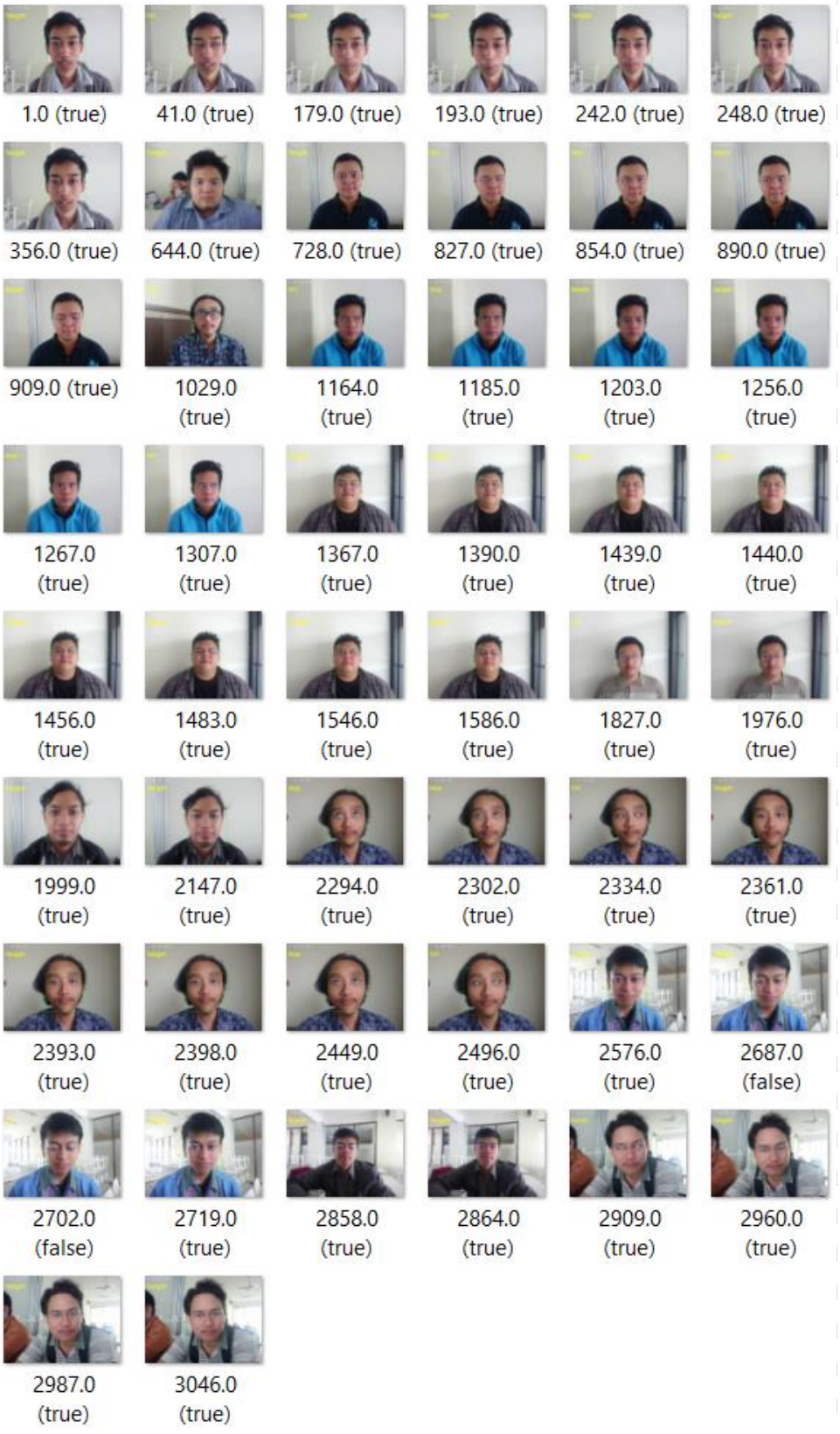
Solem, J. Erik, 2012. *Programming Computer Vision with Python*. O'Reilly Media.

Suartika, I Wayan, Arya Yudhi Wijaya & Rully Soelaiman, 2016. *Klasifikasi Citra Menggunakan Convolutional Neural Network (CNN) pada Caltech 101*. Surabaya : Jurnal Teknik ITS.

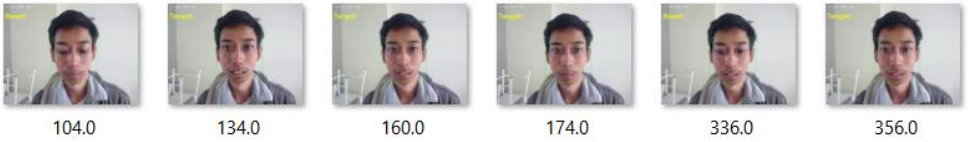
Syarif, Muhammad & Wijanarto., 2015. Deteksi Kedipan Mata dengan Haar Cascade Classifier dan Contour untuk Password Login Sistem. *Techno.COM*, Vol. 14, No. 4, pp.242-249.

Wang, J.G., and Sung, E., 2001. Gaze Determination via Images of Irises. *Image and Vision Computing*, 19, p.891-911.

LAMPIRAN A HASIL RANDOM SAMPLING PERCOBAAN 1



LAMPIRAN B HASIL RANDOM SAMPLING PERCOBAAN 2


104.0 134.0 160.0 174.0 336.0 356.0


429.0 493.0 506.0 515.0 659.0 919.0


946.0 989.0 1040.0 1055.0 1066.0 1167.0


1186.0 1311.0 1389.0 1414.0 1509.0 1510.0


1514.0 1666.0 1704.0 1712.0 1753.0 1778.0 (false)


1834.0 1844.0 1895.0 (false) 1905.0 2062.0 2085.0


2143.0 2182.0 2240.0 2244.0 2393.0 2594.0


2638.0 2729.0 2741.0 2776.0 2837.0 2862.0


2882.0 2903.0

LAMPIRAN C HASIL RANDOM SAMPLING PERCOBAAN 3