TUGAS AKHIR – KS184822

IDENTIFIKASI WAJAH MENGGUNAKAN METODE *MULTILAYER PERCEPTRON* (MLP) DAN *CONVOLUTIONAL NEURAL NETWORK* (CNN)

ULFA SITI NURAINI
NRP 062115 4000 0021

Dosen Pembimbing
Dr. Dra. Kartika Fithriasari, M.Si

PROGRAM STUDI SARJANA
DEPARTEMEN STATISTIKA
FAKULTAS MATEMATIKA, KOMPUTASI, DAN SAINS DATA
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA 2019

TUGAS AKHIR – KS184822

IDENTIFIKASI WAJAH MENGGUNAKAN METODE *MULTILAYER PERCEPTRON* (MLP) DAN *CONVOLUTIONAL NEURAL NETWORK* (CNN)

ULFA SITI NURAINI
NRP 062115 4000 0021

Dosen Pembimbing
Dr. Dra. Kartika Fithriasari, M.Si

PROGRAM STUDI SARJANA
DEPARTEMEN STATISTIKA
FAKULTAS MATEMATIKA, KOMPUTASI, DAN SAINS DATA
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA 2019

FINAL PROJECT – KS184822

FACE IDENTIFICATION USING MULTILAYER PERCEPTRON (MLP) AND CONVOLUTIONAL NEURAL NETWORK (CNN)

ULFA SITI NURAINI
SN 062115 4000 0021

Supervisor
Dr. Dra. Kartika Fithriasari, M.Si

UNDERGRADUATE PROGRAMME
DEPARTMENT OF STATISTICS
FACULTY OF MATHEMATICS, COMPUTING, AND DATA SCIENCE
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA 2019

LEMBAR PENGESAHAN

IDENTIFIKASI WAJAH MENGGUNAKAN METODE *MULTILAYER PERCEPTRON* (MLP) DAN *CONVOLUTIONAL NEURAL NETWORK* (CNN)

TUGAS AKHIR

Diajukan untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Statistika
pada
Program Studi Sarjana Departemen Statistika
Fakultas Matematika, Komputasi, dan Sains Data
Institut Teknologi Sepuluh Nopember

Oleh :
Ulfa Siti Nuraini
NRP. 062115 4000 0021

Disetujui oleh Pembimbing:
Dr. Dra. Kartika Fithriasari, M.Si ( )
NIP. 19691212 199303 2 002

Mengetahui,
Kepala Departemen Statistika

Dr. Suhartono
NIP. 19710929 199512 1 001

SURABAYA, JULI 2019

IDENTIFIKASI WAJAH MENGGUNAKAN METODE *MULTILAYER PERCEPTRON* (MLP) DAN *CONVOLUTIONAL NEURAL NETWORK* (CNN)

Identitas Mahasiswa	**:**	**Ulfa Siti Nuraini**
NRP	**:**	**062115 4000 0021**
Departemen	**:**	**Statistika**
Dosen Pembimbing	**:**	**Dr. Dra. Kartika Fithriasari, M.Si**

Abstrak

Presensi merupakan sebuah kegiatan pengambilan data guna mengetahui jumlah kehadiran dalam suatu kondisi. Presensi secara manual rentan dalam terjadinya kecurangan misal pemalsuan tanda tangan. Bahkan presensi yang menggunakan teknologi yaitu GPS juga belum bisa menjadi tolak ukur kedisiplinan waktu. Sehingga perlu digunakan sistem biometrik yaitu wajah. Wajah termasuk dalam biometrik fisiologis dengan tingkat tantangan yang lebih besar daripada biometrik fisiologis yang lain. Dalam pengolahan citra, sering menggunakan algoritma machine learning. Metode yang umum digunakan yaitu Neural Network dimana yang memiliki tambahan layer disebut Multilayer Perceptron (MLP). Selain itu teknik lain yang digunakan dalam pengolahan citra yaitu Convolutional Neural Network (CNN). Dalam tugas akhir ini, dilakukan perbandingan metode MLP dan CNN dalam pengolahan citra dengan studi kasus identifikasi wajah. Penelitian ini menggunakan 4 ×10 × 8 citra sebagai input dan variabel target adalah identitas dari 4 orang. Kesimpulan yang didapatkan yaitu metode Convolutional Neural Network yang dilihat dari accuracy, precision, sensitivity, dan Fscore memiliki hasil yang lebih baik daripada metode Multilayer Perceptron.

Kata kunci: Citra, Convolutional Neural Network, Kinerja Klasifikasi, Multilayer Perceptron, Wajah

(Halaman ini sengaja dikosongkan)

FACE IDENTIFICATION USING MULTILAYER PERCEPTRON (MLP) AND CONVOLUTIONAL NEURAL NETWORK (CNN)

Name	**:**	**Ulfa Siti Nuraini**
Student Number	**:**	**062115 4000 0021**
Department	**:**	**Statistics**
Supervisor	**:**	**Dr. Dra. Kartika Fithriasari, M.Si**

Abstract

Presence is an activity of taking data to find out the number of attendance. Manually presence is susceptible to fraud, like signature forgery. Even presence that uses GPS technology cannot be a benchmark for discipline of time. So, it is necessary used biometric systems, namely face. Face is included in physiological biometrics with a greater level of challenge than other physiological biometrics. Algorithm that is often used for image processing is machine learning. The commonly used method is the Neural Network where the additional layer is called the Multilayer Perceptron (MLP). Besides, other technique used in image processing is Convolutional Neural Network (CNN). In this final project, a comparison of MLP and CNN methods was carried out in image processing with a face identification case study. This study uses 4 × 10 × 8 images as an input and identity of 4 people as a target. The conclusion is the Convolutional Neural Network method which is seen from accuracy, precision, sensitivity, and Fscore, has better results than the Multilayer Perceptron method.

Keywords: Convolutional Neural Network, Face, Image, Multilayer Perceptron

(Halaman ini sengaja dikosongkan)

KATA PENGANTAR

Puji syukur penulis panjatkan atas rahmat dan hidayah yang diberikan Allah SWT sehingga penulis dapat menyelesaikan laporan Tugas Akhir yang berjudul "Identifikasi Wajah Menggunakan Metode *Multilayer Perceptron* (MLP) dan *Convolutional Neural Network* (CNN)" dengan lancar.

Penulis menyadari bahwa Tugas Akhir ini dapat terselesaikan tidak terlepas dari bantuan dan dukungan berbagai pihak. Oleh karena itu, penulis menyampaikan terima kasih kepada:

1. Kedua orang tua, Bapak Ali dan Ibu Tatuk Eko Sihwinantu serta keluarga Mbak Alwin Iklamatul Fuadah, Mas Hartono, Adik Dwi Arini Salsabila dan Raisha Aqila Azzahra atas segala do'a, nasehat, kasih sayang, dan dukungan yang diberikan kepada penulis demi kesuksesan dan kebahagiaan penulis.
2. Dr. Suhartono selaku Ketua Departemen Statistika dan Dr. Santi Wulan Purnami, S.Si, M.Si. selaku Ketua Program Studi Sarjana yang telah memberikan fasilitas, sarana, dan prasarana.
3. Dr. Dra. Kartika Fithriasari, M.Si selaku dosen wali yang telah memberikan saran, dukungan, arahan, nasihat selama proses belajar di Departemen Statistika ITS, dan juga selaku dosen pembimbing Tugas Akhir yang telah meluangkan waktu dan dengan sangat sabar memberikan bimbingan, saran, dukungan, serta motivasi selama penyusunan Tugas Akhir.
4. Pratnya Paramitha Oktaviana,S.Si.,M.Si. dan Prof. Drs. Nur Iriawan, MIKomp.,Ph.D. selaku dosen penguji yang selalu sabar dalam mengomentari serta memberikan masukan dan saran dalam penyelesaian Tugas Akhir.
5. Seluruh dosen Statistika ITS yang telah memberikan ilmu dan pengetahuan yang tak ternilai harganya, serta segenap karyawan Departemen Statistika ITS, khususnya bapak

Effendy dan Bapak Anton yang selalu siap siaga membantu dalam administrasi.

6. Sahabat-sahabat *Vitamin Squad*: Imas Ayu, Shindi Shella, Henidar Islami, dan Fitria Nurul.
7. Devita Prima Vernanda, Moch. Ihsan Ananto, dan Moh. Haidar Alvin yang telah bersedia untuk menjadi data dalam Tugas Akhir ini.
8. *My partner* : Taufik Azmi, Arlandio Nur Fawzi, dan Nurun Nahdliyah, serta Rike Andriyani yang selalu menjadi penyemangat dan pendengar yang baik.
9. *Pikachu* : Kak Rizky Fauzy, Kak Annisa Nurhadirat, Achmad Naufal, dan A. R. Syihab.
10. Teman-teman satu dosen pembimbing : Farizah, Cabun, Syihab, Shindi, Ihsan, Naren, Rahayu, dan Fonda yang selalu memberi *support* selama proses pengerjaan Tugas Akhir.
11. *Statistics Computer Course* (SCC) 16/17 dan 17/18, khususnya *Training Development*: Mas Irfan, Mas Udin, Feli, Fiika, Dio, Asva, Rachel, Kinan, Nisfi, Nadhifa yang memberikan pembelajaran berorganisasi.
12. Teman-teman Statistika ITS ∑26 angkatan 2015, yang selalu memberikan dukungan kepada penulis selama ini.
13. Semua teman, relasi dan berbagai pihak yang tidak bisa penulis sebutkan identitasnya satu persatu yang telah membantu dalam penulisan laporan ini.

Besar harapan penulis untuk mendapatkan kritik dan saran yang membangun sehingga Tugas Akhir ini dapat memberikan manfaat bagi semua pihak yang terkait.

Surabaya, Juli 2019

Penulis

DAFTAR ISI

Halaman

DAFTAR GAMBAR

(Halaman ini sengaja dikosongkan)

DAFTAR TABEL

(Halaman ini sengaja dikosongkan)

DAFTAR LAMPIRAN

(Halaman ini sengaja dikosongkan)

BAB I
PENDAHULUAN

1.1 Latar Belakang

Sistem autentikasi identitas manusia yang mendapatkan kinerja tinggi sangat diperlukan dalam era ini. Salah satu contoh dari autentikasi identitas manusia adalah presensi. Presensi dalam Kamus Besar Bahasa Indonesia berarti kehadiran, sehingga presensi dapat diartikan sebagai sebuah kegiatan pengambilan data guna mengetahui jumlah kehadiran dalam suatu kondisi. Presensi secara manual rentan misal tanda tangan dapat menimbulkan beberapa masalah. Seringkali memanfaatkan celah dan saling bekerjasama untuk melakukan kecurangan, misal seorang pegawai yang hadir menandatangani pegawai lain yang tidak hadir dalam suatu pekerjaan. Saat ini sudah ada aplikasi presensi yang sudah memanfaatkan sistem teknologi, sebagai contoh SiPERLU (Sistem Informasi Presensi Lumajang) yang merupakan sistem presensi untuk Aparatur Sipil Negara (ASN) di kabupaten Lumajang. Sistem aplikasi ini menggunakan GPS dari *handphone* sehingga dapat diketahui lokasi pegawai tersebut saat jam masuk ataupun pulang. Plt. Sekretaris Badan Kepegawaian Daerah (BKD) Kabupaten Lumajang, Cipto Sujarwo menerangkan bahwa penerapan Si Perlu tersebut sebelumnya telah diterapkan bagi PNS dan diklaim berhasil mengurangi ketidaksiplinan pegawai karena diterapkan sanksi bagi pelanggarnya (Rahmat, 2019). Namun menurut Ir. Indah Amperawati sebagai Wakil Bupati Lumajang saat itu, aplikasi SiPERLU selama ini belum bisa menjadi tolak ukur kedisiplinan waktu karena masih banyak ditemukan kecurangan, seperti minta tolong teman kantor atau bahkan menggunakan jasa ojek *online* (Arianto, 2019). Selain itu, presensi secara konvensional dengan menggunakan kartu identitas juga tidak cukup handal, karena terdapat kemungkinan kartu identitas tersebut hilang dan digunakan oleh pengguna yang tidak berwenang. Untuk itu, ada sebuah sistem yang dapat menangani masalah tersebut yaitu sistem biometrik yang mana keunikan fitur

dan karaketistik biometrik ini tidak dapat dipindahtangankan (*non-transferable characteristics*) (Dunstone & Yager, 2009).

Dalam biometrik, terdapat biometrik fisiologis yaitu sidik jari yang kerap kali telah digunakan dalam berbagai instansi untuk mencatat kehadiran pegawainya. Penggunaan sidik jari ini sangat bermanfaat dalam pencatatan kehadiran karena tidak dapat dimanipulasi oleh orang lain. Selain sidik jari, terdapat biometrik dengan tingkat tantangan yang lebih besar yaitu wajah. Wajah memiliki penambahan komponen struktural, seperti rambut, atau dengan/tanpa kacamata, ataupun penggunaan kosmetik yang memiliki variabilitas tinggi dalam bentuk, ukuran, dan warna. Selain itu, terdapat faktor lain yang meliputi pencahayaan, oklusi, dan ekspresi. Pencahayaan adalah persebaran cahaya yang dapat membuat bayangan baru pada beberapa bagian dari wajah. Oklusi adalah hasil dari suatu objek yang dapat ditambahkan pada sekitar wajah seperti topi, kacamata, kerudung, dan sebagainya. Serta ekspresi adalah pandangan raut muka yang memperlihatkan perasaan, seperti sedih, senang, cemberut, dan lainnya (Dunstone & Yager, 2009).

Seiring berkembangnya teknologi, data yang digunakan tidak selalu berbentuk angka dengan struktur data yang telah dikonstruksi di awal. Namun, data juga dapat berupa gambar, suara, dan lain sebagainya. Penelitian pada data berupa gambar pernah dilakukan oleh Iriawan dkk (2018) dimana menggunakan beberapa metode klaster yang khususnya dapat meng-*handle noise* pada segmentasi citra tumor otak. Dalam pengolahan data yang seperti itu sering menggunakan algoritma *machine learning*. *Machine learning* adalah kecerdasan buatan yang bertujuan untuk mengoptimalkan kinerja dari suatu sistem dengan mempelajari data sampel atau data histori (Alpaydin, 2009). Dengan bahasa yang sederhana, teknik *machine learning* sebagai algoritma komputer untuk mempelajari data, mengenali pola, dan membuat model berdasarkan data historik. Jaringan Syaraf Tiruan (*Artificial Neural Network*, ANN) merupakan salah satu bagian dari *machine learning*. ANN adalah salah satu bentuk kecerdasan buatan yang

mempunyai kemampuan untuk belajar dari data dan tidak membutuhkan waktu lama dalam pembuatan model (Setiawan & Rudiyanto, 2004). Jenis model ANN yang terdiri dari banyak lapisan disebut sebagai *Multilayer Perceptron* (MLP) yang menghubungkan secara penuh antar neuronnya dan memiliki kemampuan klasifikasi yang powerful. Salah satu penelitian pernah dilakukan oleh Gurpreet Singh dan Manoj Sachan (2014) dalam studi kasus mengidentifikasi tulisan tangan karakter Gurmukhi (aksara India) dengan menggunakan MLP. Dari penelitian tersebut didapatkan akurasi maksimum hingga 98,96%.

Teknik lain yang memiliki hasil signifikan dalam pengenalan citra adalah *Convolutional Neural Network* (CNN) (Fukushima, 1980). Hal tersebut dikarenakan CNN berusaha meniru sistem pengenalan citra pada visual *cortex* manusia sehingga memiliki kemampuan mengolah informasi citra. Kemampuan CNN di klaim sebagai model terbaik untuk memecahkan permasalah *object detection* dan *object recognition* (Karpathy, 2018). Suatu penelitian dilakukan oleh Tivive & Bouzerdoum (2006) tentang identifikasi wajah berdasarkan jenis kelamin dengan menggunakan metode *Convolutional Neural Network* (CNN) dan didapatkan akurasi sebesar 85,7%.

Perbandingan metode *Multilayer Perceptron* dan *Convolutional Neural Network* pernah dilakukan oleh Edgar Medina dkk (2017) untuk mendeteksi klasifikasi *algae* di dalam pipa. Pada penelitian tersebut didapatkan metode terbaik yaitu dengan *Convolutional Neural Network* (CNN) yang memberikan akurasi tinggi sebesar 99,39%, namun dengan menggunakan *Multilayer Perceptron* (MLP) identifikasi *algae* mendapatkan akurasi sebesar 95,7%. Perbandingan kedua metode ini yang dilakukan pada tugas akhir ini namun pada studi kasus identifikasi wajah dengan metode *Multilayer Perceptron* (MLP) dan *Convolutional Neural Network* (CNN) agar dapat digunakan dalam pencatatan kehadiran dengan sistem biometrik selain menggunakan sidik jari.

1.2 Rumusan Masalah

Identifikasi manusia dengan sistem biometrik sudah pernah dilakukan yaitu dengan *fingerprint scanner*. Seiring kemajuan teknologi, sistem biometrik yang lain perlu juga digunakan untuk pencatatan kehadiran, salah satunya dengan melakukan identifikasi wajah. Berdasarkan penelitian sebelumnya, identifikasi citra sering menggunakan metode *Multilayer Perceptron* (MLP) dan *Convolutional Neural Network* (CNN). Oleh karena itu, maka didapatkan permasalahan utama adalah apa metode terbaik antara *Multilayer Perceptron* (MLP) dan *Convolutional Neural Network* (CNN) dalam identifikasi wajah? Metode terbaik tersebut kemudian akan digunakan dalam program atau *graphic user interface* (GUI) untuk melakukan identifikasi wajah.

1.3 Tujuan Penelitian

Berdasarkan rumusan masalah di atas, tujuan utama yang ingin dicapai dalam penelitian ini adalah untuk mendapatkan metode terbaik antara *Multilayer Perceptron* (MLP) dan *Convolutional Neural Network* (CNN) dalam identifikasi wajah. Setelah didapat metode terbaik, maka akan dibuat juga program atau *graphic user interface* (GUI) yang dapat mengidentifikasi wajah.

1.4 Manfaat Penelitian

Pada penelitian ini diharapkan dapat dimanfaatkan oleh pihak instansi atau lembaga seperti pemerintah atau universitas dengan tujuan pengenalan identitas seseorang melalui identifikasi wajah yang dapat dijadikan sebagai pencatatan kehadiran atau absensi. Selain itu, dapat memberikan kontribusi bagi dunia penelitian khususnya dalam pengembangan identifikasi wajah.

1.5 Batasan Masalah

Batasan masalah pada penelitian ini adalah sebagai berikut.

1. Setiap foto diambil menggunakan *webcam* Acer Aspire E 11 berdimensi 640 × 480 pixel dengan eksistensi .JPEG serta kedalaman 24 bit.
2. Jumlah orang yang digunakan pada penelitian ini adalah 4 orang, dengan setiap foto hanya terdiri dari 1 orang.
3. Setiap orang diambil 10 foto dari sisi yang berbeda dan pengambilan foto sejajar dengan *webcam*.

(Halaman ini sengaja dikosongkan)

BAB II
TINJAUAN PUSTAKA

Bab ini membahas mengenai *Multilayer Perceptron* (MLP), *Convolutional Neural Network* (CNN), fungsi aktivasi, optimasi parameter, k-*fold cross validation*, dan kinerja klasifikasi.

2.1 Citra Digital

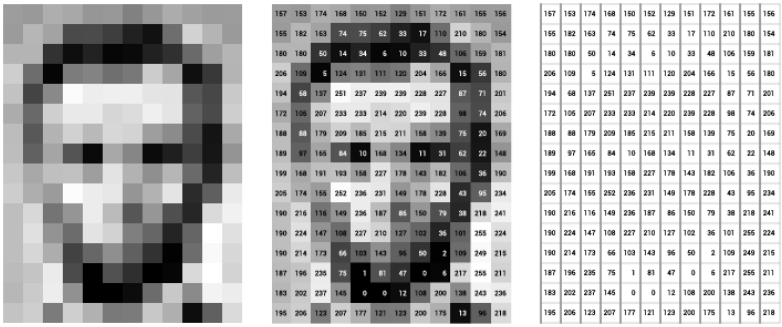
Secara harfiah, citra adalah gambar pada dua dimensi. Ditinjau dari sudut pandang matematis, citra merupakan fungsi dari intensitas cahaya yang termaktup dalam dua dimensi (Munir, 2004). Citra digital adalah data digital pada komputer yang merepresentasikan sebuah citra. Sebuah citra yang diwakili oleh $f(x,y)$ berbentuk *matrix* yang terdiri dari M kolom dan N baris, dimana perpotongan antara kolom dan baris inilah yang disebut *picture element*, *image elements*, *pels*, atau *pixel*. Citra digital juga dapat digambarkan sebagai fungsi $f(x,y)$ dengan x dan y merupakan koordinat pada sebuah bidang datar yang merepresentasikan kumpulan *pixel* dalam dua dimensi (Woods & Gonzales, 2008).

Dalam penelitian tentang citra digital, biasanya terdapat tipe-tipe dasar citra digital yaitu citra RGB dan citra *grayscale*. Citra warna RGB biasanya disebut Citra *True Color* atau citra asli yang ditangkap oleh kamera. Pada citra dengan tipe RGB, setiap *pixel* memiliki 3 komponen warna, yaitu merah (*Red* = R), hijau (*Green* = G), dan biru (*Blue* = B). Setiap komponen warna memiliki jangkauan nilai antara 0 hingga 255. Warna pada *pixel* ditentukan dari kombinasi ketiga komponen warna tersebut. Citra bertipe *grayscale* merupakan citra yang hanya memiliki satu nilai pada setiap *pixel*nya, nilai tersebut disebut sebagai derajat keabuan. Derajat keabuan memiliki nilai antara 0 (hitam) sampai 255 (putih) (Woods & Gonzales, 2008). Proses perubahan citra berwarna ke dalam citra *grayscale* dapat dilakukan dengan pembobotan pada setiap komponen warna R, G, dan B. Pembobotan ini didasari pada tingkat kepekaan manusia dalam memandang komposisi warna

(Faculty of Chemistry, 2019). Persamaan pembobotan ini dapat dirumuskan pada persamaan (2.1).

$$gray = 0{,}299\text{R}+0{,}587\text{G}+0{,}114\text{B}. \qquad (2.1)$$

Berikut adalah contoh citra bertipe *grayscale* pada Gambar 2.1.



Gambar 2.1 Citra *Grayscale*

Gambar 2.1 dapat diimplementasikan sebagai *matrix* dengan 16 kolom dan 12 baris. Dimisalkan suatu citra $\mathbf{f_n(x,y)}$ bertipe *grayscale c* kolom dan *r* baris, maka *matrix* dari citra tersebut dapat dilihat sebagai berikut (Woods & Gonzales, 2008).

$$\mathbf{f}_n(\mathbf{x},\mathbf{y}) = \begin{bmatrix} f_n(0,0) & f_n(0,1) & \cdots & f_n(0,c-1) \\ f_n(1,0) & f_n(1,1) & \cdots & f_n(1,c-1) \\ \vdots & \vdots & \ddots & \vdots \\ f_n(r-1,0) & f_n(r-1,1) & \cdots & f_n(r-1,c-1) \end{bmatrix},$$

dengan

x = 0,1,2, … , r-1,
y = 0,1,2, … , c-1,
$f_n(x,y)$ = 0,1,2, …, 255,

dimana : c = jumlah *pixel* kolom (*column*) pada *array* citra,
r = jumlah *pixel* baris (*row*) pada *array* citra,
$f_n(x,y)$ = nilai derajat keabuan (*graylevel*).

Hasil dari satu citra $f_n(x,y)$ tersebut diubah dari bentuk *matrix* ke bentuk vektor sehingga akan mendapatkan hasil vektor sebagai berikut untuk satu citra.

$$\mathbf{f}_n(\mathbf{x},\mathbf{y}) = \begin{bmatrix} f_n(0,0) & f_n(0,1) & \cdots & f_n(0,c-1) & f_n(1,0) & \cdots & f_n(r-1,c-1) \end{bmatrix}$$

Untuk keseluruhan citra yang akan diolah, maka dilakukan perubahan *matrix* setiap citra ke dalam bentuk vektor lalu menggabungkannya dalam 1 *matrix* dengan ukuran kolom sebanyak $n_x = r \times c$ dan baris sebanayak N (banyaknya gambar). Hasil dari *matrix* tersebut adalah sebagai berikut.

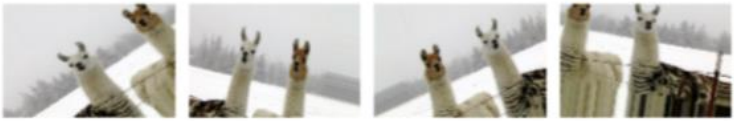
$$\mathbf{f}(\mathbf{x},\mathbf{y}) = \begin{bmatrix} f_1(0,0) & f_1(0,1) & \cdots & f_1(0,c-1) & f_1(1,0) & \cdots & f_1(r-1,c-1) \\ f_2(0,0) & f_2(0,1) & \cdots & f_2(0,c-1) & f_2(1,0) & \cdots & f_2(r-1,c-1) \\ \vdots & \vdots & \cdots & \vdots & \vdots & \cdots & \vdots \\ f_N(0,0) & f_N(0,1) & \cdots & f_N(0,c-1) & f_N(1,0) & \cdots & f_N(r-1,c-1) \end{bmatrix}$$

2.2 Pengolahan Citra

Pengolahan citra (*image processing*) adalah suatu cara untuk memproses atau memanipulasi citra menjadi citra lain yang kualitasnya lebih baik, agar citra yang mengalami gangguan mudah diinterpretasi, baik oleh manusia maupun mesin (Gonzalez, Woods, & Eddins, 2009). Pengolahan citra dapat juga digunakan untuk menambahkan data *training* sehingga dapat mengurangi *overfitting*, yang juga disebut dengan augmentasi citra (Wang & Perez, 2017). Contoh operasi augmentasi citra adalah sebagai berikut.

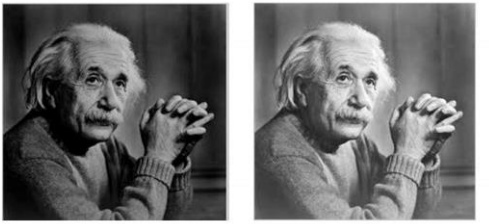
1. Transformasi secara tradisional

   Transformasi secara tradisional memanipulasi citra dengan cara kombinas. Duplikasi citra untuk setiap citra *input* dapat dengan cara rotasi (*rotated*), pembalikan (*flip*), dan lain-lain.



Gambar 2.2 Contoh Transformasi secara Tradisional

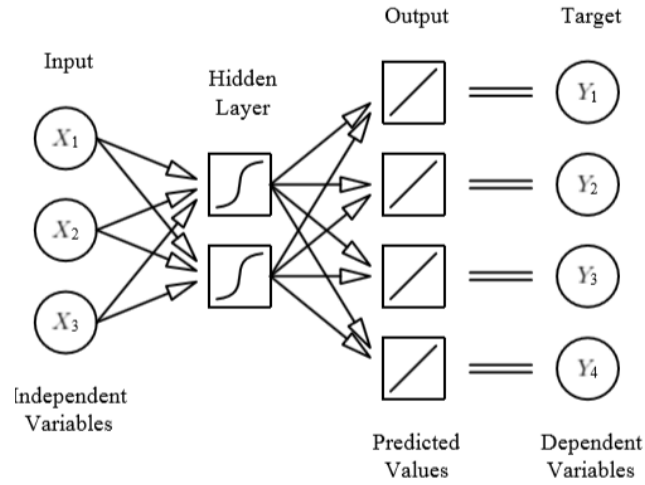
2. *Image enhancement*
   Operasi ini bertujuan untuk memperbaiki kualitas citra dengan cara memanipulasi parameter-parameter citra, sehingga ciri-ciri khusus yang terdapat dalam citra lebih ditonjolkan. Beberapa contoh *image enhancement* adalah perbaikan gelap/terang (*brightness*), perbaikan *contrast*, penajaman (*sharpening*), pemberian warna semu (*pseudocoloring*), dan lain-lain.



Gambar 2.3 Contoh Augmentasi *Image Enhancement*

2.3 *Multilayer Perceptron* (MLP)

Multilayer Perceptron (MLP) adalah sebuah metode pada *Neural Network* dimana model didalamnya menyertakan estimasi bobot dan *hidden layer*, dan didalam *hidden layer* tersebut menggunakan fungsi aktivasi nonlinier (Sarle, 1994). Fungsi aktivasi merupakan fungsi yang menggambarkan hubungan antara *input* untuk mengeluarkan nilai *output* yang mungkin berbentuk *linear* ataupun *non-linear* (Nwankpa dkk, 2018). Dimana jika tidak ada fungsi aktivasi didalam suatu *neuron* maka fungsi pada *neuron* tersebut mengikuti fungsi *linear* atau nilai yang masuk pada *neuron* tersebut sama dengan nilai keluar dari *neuron.* Pada dasarnya, MLP terdiri dari 3 layer, yaitu *input layer, hidden layer, dan output layer* yang ditunjukkan pada Gambar 2.4. Dalam MLP, dapat memvariasikan jumlah *hidden layer* dan jumlah neuron didalamnya *hidden layer* (Sarle, 1994). *Hidden layer* menerima input yang telah terboboti dari *input layer* dan mengirimkan hasilnya kepada *layer* yang berikutnya.



Gambar 2.4 Arsitektur *Multilayer Perceptron*

Gambar 2.4 menggunakan 1 *hidden layer* berjumlah 2 *neurons.* Setiap *input* yang terhubung pada setiap *neuron* pada *hidden layer* maupun *output layer* masing-masing memiliki bias dan pembobot. Persamaan (2.2) merupakan persamaan dalam perhitungan pada *neuron hidden layer* sebelum masuk pada *neuron hidden layer*. Setelah itu perhitungan pada *neuron hidden layer* dengan fungsi aktivasi didalamnya pada persamaan (2.3) (Sarle, 1994). Fungsi aktivasi yang digunakan pada setiap *neuron* di *hidden layer* pada penelitian ini adalah fungsi aktivasi ReLU (*Rectified Linear Unit*). ReLU merupakan fungsi aktivasi yang memaksa nilai yang kurang dari 0 ke nilai 0. (Nwankpa dkk, 2018).

$$p_j = a_j + \sum_{j=1}^{n_h} w_{i,j} X_i, \tag{2.2}$$

$$q_j = f_h(p_j) = \max(0, p_j) = \begin{cases} p_j, & \text{jika } p_j \geq 0, \\ 0, & \text{jika } p_j < 0, \end{cases} \tag{2.3}$$

dengan:

X_i = variabel input,
a_j = nilai bias untuk *hidden layer*,
w_{ij} = nilai pembobot pada *hidden layer*,
p_j = *output* setiap *neuron hidden layer* tanpa fungsi aktivasi,
q_j = *output* setiap *neuron hidden layer* setelah dimasukkan dalam fungsi aktivasi,
i = 1,2,…,n_x, n_x merupakan banyak *pixel* pada input citra,
j = 1,2,…,n_h. n_h menunjukkan banyak *neuron* pada *hidden layer*.

Selanjutnya dilakukan perhitungan untuk menghubungkan *hidden layer* dan *output layer* pada persamaan (2.4). Setelah itu, untuk mendapatkan nilai *predicted value* maka hasil dari persamaan (2.4) dimasukkan ke dalam fungsi aktivasi pada setiap *neuron output layer* pada penelitian ini yaitu fungsi aktivasi *softmax*. Fungsi aktivasi *softmax* merupakan fungsi *non-linear* dimana input untuk fungsi ini berupa bilangan *real* dan *output* dari fungsi ini memiliki *range* antara 0 hingga 1, dengan jumlah dari semua peluang yang dihasilkan adalah bernilai 1. Fungsi aktivasi *softmax* digunakan pada *multiclass* pada klasifikasi yang digunakan pada *output layer* (Nwankpa dkk, 2018). Perhitungan fungsi aktivasi *softmax* terdapat pada persamaan (2.5).

$$r_k = b_k + \sum_{k=1}^{K} v_{j,k} q_j, \tag{2.4}$$

$$\hat{y}_k = f_o(r_k) = \frac{\exp(r_k)}{\sum_{k=1}^{K} \exp(r_k)}, \tag{2.5}$$

dengan :
b_k = nilai bias untuk *output layer*,
v_{jk} = nilai pembobot pada *output layer*,
q_j = nilai *output* yang keluar dari setiap *neuron hidden layer*,
r_k = *output* setiap *neuron output layer* tanpa fungsi aktivasi,
$\hat{y}_k$ = *output* setiap *neuron output layer* setelah dimasukkan dalam fungsi aktivasi,
j = 1,2,…,n_h. n_h menunjukkan banyak *neuron* pada *hidden layer*.

k = 1,2,…,K. K menunjukkan banyak kelas

Terakhir, perhitungan untuk mendapatkan hasil klasifikasi. Target atau kelas yang didapatkan merupakan keputusan nilai *predicted value* yang tidak selalu menghasilkan nilai bulat (1,2,…,K) namun dapat berupa nilai peluang dengan penentuan kelas melalui *threshold* dari peneliti (Sarle, 1994).

2.3.1 *Loss*

Loss atau *loss function* merupakan fungsi yang digunakan sebagai kriteria yang harus diminimumkan yang dapat dijadikan solusi untuk membandingkan model (Russell & Marksll, 1999). *Loss* sangat erat kaitannya dengan fungsi aktivasi pada *output layer*. Fungsi akivasi *softmax* akan menghasilkan nilai probabilitas, dimana perhitungan *error* antara dua distribusi probabilitas menggunakan *loss function* yang bernama *cross-entropy* (GoodFellow, Bengio, & Courville, 2016). Secara teknis, *cross-entropy* memperkirakan perbedaan antara probabilitas hasil dari *output layer* dengan hasil target berupa nilai 0 dan 1. Perhitungan *cross-entropy* tercantum pada persamaan (2.6) .

$$Loss = -\sum_{k=1}^{K} y_k \ln(\hat{y}_k), \tag{2.6}$$

dimana:

$Loss$ = *loss function* yang digunakan,

y_k = nilai target berupa nilai 0 dan 1,

$\hat{y}_k$ = hasil *predicted value* didapatkan dari persamaan (2.5),

k = 1,2,…,K. K menunjukkan banyak kelas.

2.3.2 Optimasi Parameter *Adam*

Optimasi parameter digunakan untuk meminimumkan nilai *loss* sehingga *loss* merupakan kunci dalam optimasi parameter bias dan bobot. Optimasi parameter yang digunakan dalam penelitian ini yaitu optimasi parameter *Adam*. *Adam* (*adaptive moment estimation*) merupakan algoritma optimasi *training* adaptif yang dirancang khusus untuk *training* pada metode *deep learning*. *Adam* menggunakan *gradient*, lalu estimasi momen pertama dan kedua,

serta mengoreksi dengan *bias correction* (Kingma & Ba, 2014). Dalam mengoptimasi parameter dengan menggunakan *Adam*, hal pertama yang harus dilakukan adalah menghitung *gradient* dari *loss function* terhadap parameter yaitu bias dan pembobot. Penyelesaian *gradient* pada *neural network* ini menggunakan *chain rule* untuk mendapatkan *partial derivative* (Bishop, 2006). *Gradient loss function* terhadap bias pada *input layer* (a_j), bobot antara *input layer* dan *hidden layer* ($w_{i,j}$), bias pada *hidden layer* (b_k), serta bobot antara *input layer* dan *output layer* ($v_{j,k}$) terdapat pada Tabel 2.1.

Tabel 2.1 *Gradient Loss Function* terhadap Parameter

Parameter (θ)	***Gradient* (g_t)**
a_j	$\frac{\partial Loss}{\partial a_j} = \frac{\partial Loss}{\partial r_k}\frac{\partial r_k}{\partial q_j}\frac{\partial q_j}{\partial p_j}\frac{\partial p_j}{\partial a_j} = \begin{cases}(y_k - \hat{y}_k)v_{j,k}, & \text{jika } p_j \geq 0\\ 0, & \text{jika } p_j < 0\end{cases}$
$w_{i,j}$	$\frac{\partial Loss}{\partial w_{i,j}} = \frac{\partial Loss}{\partial r_k}\frac{\partial r_k}{\partial q_j}\frac{\partial q_j}{\partial p_j}\frac{\partial p_j}{\partial w_{i,j}} = \begin{cases}(y_k - \hat{y}_k)v_{j,k}X_i, & \text{jika } p_j \geq 0\\ 0, & \text{jika } p_j < 0\end{cases}$
b_k	$\frac{\partial Loss}{\partial b_k} = \frac{\partial Loss}{\partial r_k}\frac{\partial r_k}{\partial b_k} = (y_k - \hat{y}_k)$
$v_{j,k}$	$\frac{\partial Loss}{\partial v_{j,k}} = \frac{\partial Loss}{\partial r_k}\frac{\partial r_k}{\partial v_{j,k}} = (y_k - \hat{y}_k)q_j$

Selanjutnya meng-*update gradient exponential moving averages* (m_t) dan *gradient* kuadrat (v_t) dengan *hyper-parameters* yaitu β_1 dan β_2 (secara *default* yaitu $\beta_1 = 0{,}9$ dan $\beta_2 = 0{,}999$). *Moving averages* ini mengestimasi momen pertama (rata-rata) dan momen kedua (*uncentered variance*) dari *gradient* dengan inisialisasi vektor momen adalah $m_0 = 0$ dan $v_0 = 0$. Persamaan untuk menghitung nilai m_t dan v_t terdapat pada persamaan (2.7) dan (2.8) (Kingma & Ba, 2014).

$$m_t = \beta_1 \cdot m_{t-1} + (1-\beta_1) \cdot g_t \tag{2.7}$$

$$v_t = \beta_2 \cdot v_{t-1} + (1-\beta_2) \cdot g_t \odot g_t \tag{2.8}$$

dimana g_t adalah hasil dari *gradient* fungsi *loss* pada setiap parameter yang tercantum pada Tabel 2.1. Sedangkan $g_t \odot g_t$ adalah perkalian masing-masing elemen vektor. Dimana *t* mencantumkan banyak iterasi (Kingma & Ba, 2014).

Setelah itu, hasil dari m_t dan v_t dikoreksi untuk memperbaiki nilai bias yang didapatkan. Jika hasil dari momen tidak diperbaiki maka akan mengakibatkan bobot *gradient* yang kecil. Persamaan untuk menghitung nilai $\hat{m}_t$ dan $\hat{v}_t$ terdapat pada persamaan (2.9) dan (2.10) (Kingma & Ba, 2014).

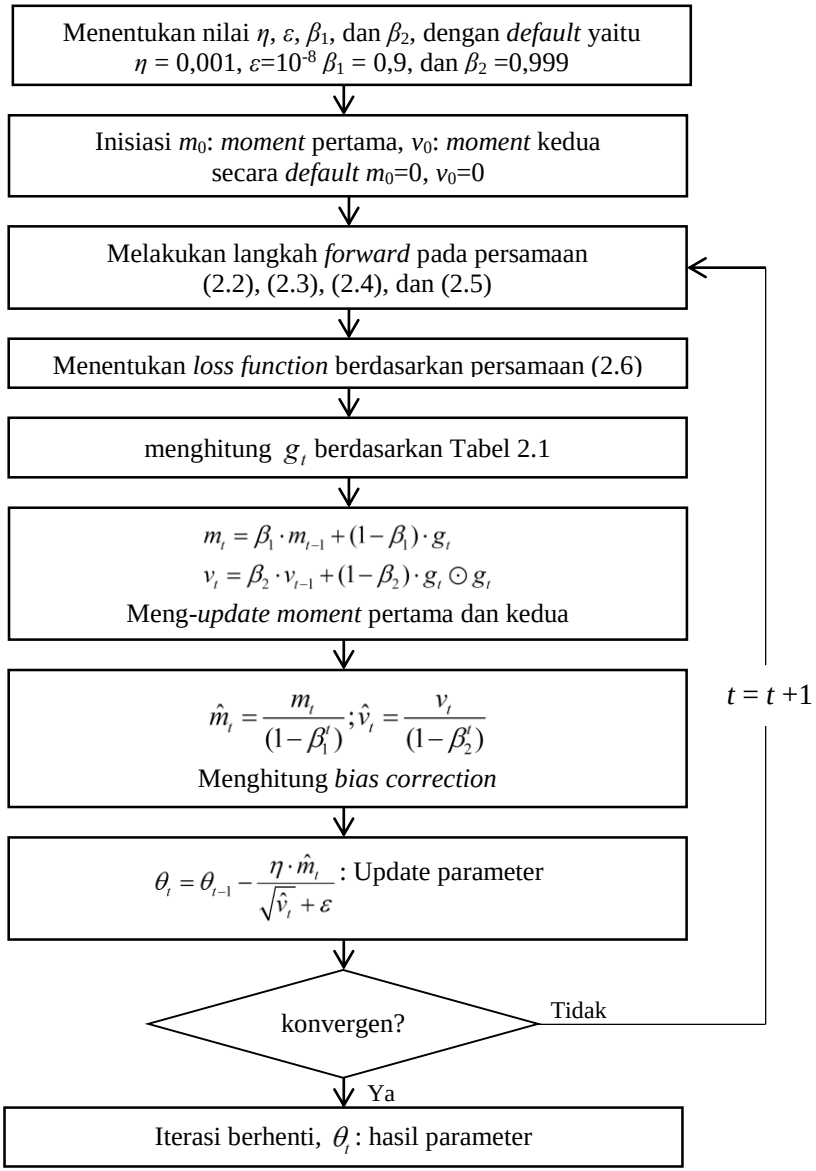
$$\hat{m}_t = \frac{m_t}{(1-\beta_1^t)}, \tag{2.9}$$

$$\hat{v}_t = \frac{v_t}{(1-\beta_2^t)}. \tag{2.10}$$

Terakhir, meng-*update* parameter baik untuk bias maupun pembobot. *Update* parameter ini menggunakan nilai η =0,001, serta ε = 10^{-8}. Persamaan untuk *update* parameter terdapat pada persamaan (2.11) (Kingma & Ba, 2014).

$$\theta_t = \theta_{t-1} - \frac{\eta \cdot \hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon}, \tag{2.11}$$

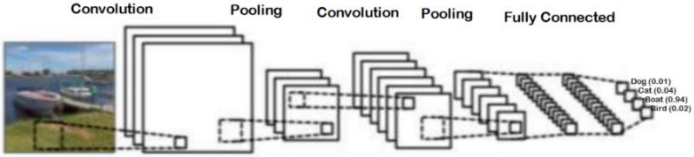
dimana θ dapat didefinisikan sebagai parameter yaitu bias pada *input layer* (a_j), bobot antara *input layer* dan *hidden layer* ($w_{i,j}$), bias pada *hidden layer* (b_k), serta bobot antara *input layer* dan *output layer* ($v_{j,k}$). Algoritma *Adam* ini disajikan dalam diagram alir pada Gambar 2.5.



Gambar 2.5 Diagram Algoritma *Adam*

2.4 *Convolutional Neural Network* (CNN)

Convolutional Neural Network (CNN) adalah pengembangan dari *Multilayer Perceptron* (MLP) yang didesain untuk mengolah data dua dimensi dalam bentuk citra. CNN mengikuti asumsi dasar *Neural Network* (NN), namun dalam CNN tidak semua *neuron* memiliki konektifitas penuh, kecuali pada lapisan yang sepenuhnya terhubung. Selain itu, arsitektur CNN memiliki komposisi layer berbeda yang terdiri dari *convolutional layer* untuk menggenerasi peta fitur, pooling layer untuk mereduksi dimensi, dan *fully-connected layer* untuk mengklasifikasikan ekstraksi fitur (Ferreira & Giraldi, 2017). Berikut merupakan ilustrasi arsitektur pada metode CNN yang ditunjukkan pada Gambar 2.6.



Gambar 2.6 Arsitektur *Convolutional Neural Network* (CNN)

Dengan kata lain, CNN mentransformasikan gambar original *layer* per *layer* dari nilai *pixel* gambar kedalam nilai skoring kelas untuk klasifikasi. *Layer* dasar yang digunakan dalam CNN adalah sebagai berikut.

1. *Convolutional Layer*

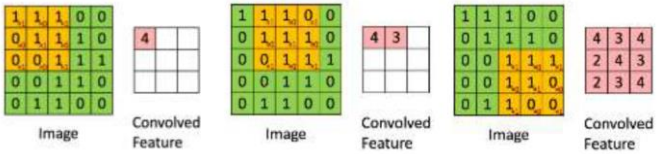
Gambar yang masuk dalam *convolutional layer* akan dilakukan konvolusi. Konvolusi pada pengolahan gambar merupakan salah satu metode untuk mendapatkan ekstraksi fitur dari sebuah gambar. Proses konvolusi adalah mengalikan sebuah gambar dengan sebuah *convolution kernel* atau *filter* yang dinyatakan dalam bentuk *matrix* dengan ukuran yang biasanya lebih kecil dari ukuran gambar. Operasi konvolusi dilakukan dengan menggeser *convolution kernel* pada setiap *pixel*. Hasil dari operasi konvolusi tersebut disimpan di dalam *matrix* yang baru (Ferreira & Giraldi, 2017). Persamaan 2.12 merupakan persamaan

operasi *convolution*, serta Gambar 2.6 menunjukkan ilustrasi dari *convolutional layer* (Ganegedara, 2018).

$$FM_{a,b} = bias + \sum_{c}^{C}\sum_{d}^{D} Z_{c,d} \times X_{a+c-1,b+d-1} \qquad (2.12)$$

keterangan :

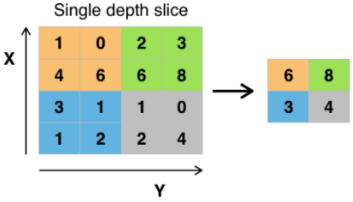
$FM_{a,b}$ = *feature map* ke-*a*,*b*,
*b*ias = bias pada *feature map*,
$Z_{c,d}$ = bobot pada *convolution kernel* ke-*c*,*d*,
X = input,
a = 1,2, …, *A*. *A* merupakan panjang *pixel* pada *feature map*,
b = 1,2, …, *B*. *B* merupakan lebar *pixel* pada *feature map*,
c = 1,2, …, *C*. *C* yaitu ukuran panjang *convolution kernel*,
d = 1,2, …, *D*. *D* yaitu ukuran lebar *convolution kernel*.



Gambar 2.7 Ilustrasi *Convolutional Layers*

2. *Pooling layer*

Layer ini digunakan untuk memastikan bahwa gambar hanya berfokus pada pola-pola tertentu yang menjadi ciri dari gambar tersebut. *Layer* ini merangkum data dengan menggeser jendela di seluruh peta fitur, menerapkan operasi di dalam jendela yaitu nilai maksimum pada setiap jendela sehingga dapat mereduksi dimensi (Ferreira & Giraldi, 2017). Ilustrasi proses *pooling* ditunjukkan pada Gambar 2.8.



Gambar 2.8 Ilustrasi *Pooling Layers*

3. *Fully-connected layers*

Layer ini digunakan untuk melakukan transformasi pada dimensi data agar dapat diklasifikasikan secara linear. *Neuron* pada lapisan ini dijadikan dalam 1 vektor, seperti *input layer* yang memiliki koneksi penuh ke semua aktivasi di lapisan sebelumnya. Pada *layer* ini memiliki algoritma yang sama dengan *Multilayer Perceptron* (Ferreira & Giraldi, 2017).

Pada metode CNN juga menggunakan fungsi aktivasi, dimana fungsi aktivasi yang digunakan digunakan setelah melewati *convolutional layer* sebelum *pooling layer*. Fungsi aktivasi yang digunakan yaitu ReLU (Ferreira & Giraldi, 2017). Persamaan untuk fungsi aktivasi ReLU tercantum pada persamaan (2.13). Fungsi aktivasi untuk *output* sama dengan metode MLP yaitu *softmax* yang terdapat pada persamaan (2.5).

$$f_h(FM_{a,b}) = \max(0, FM_{a,b}) = \begin{cases} FM_{a,b}, & \text{jika } FM_{a,b} \geq 0, \\ 0, & \text{jika } FM_{a,b} < 0, \end{cases} \quad (2.13)$$

dimana:

$FM_{a,b}$ = *feature map* ke-*a,b* didapatkan dari persamaan (2.12),
a = 1,2, … *A*. *A* merupakan panjang *pixel* pada *feature map,*
b = 1,2, … *B*. *B* merupakan lebar *pixel* pada *feature map.*

Penelitian ini, sama dengan metode *Multilayer Perceptron,* metode CNN juga menggunakan optimasi parameter *Adam* dimana proses pengoptimasian parameter berkutip pada subbab 2.3.2 tentang optimasi parameter *Adam.*

2.5 Evaluasi Ketepatan Klasifikasi

Kinerja dari suatu metode klasifikasi sangat penting digunakan untuk mendapatkan model yang terbaik. Model klasifikasi akan mengeluarkan hasil dalam bentuk diskrit maupun kontinu. Dimana diskrit akan memprediksi label kelas dari *testing*, sedangkan kontinu akan merepresentasikan estimasi dari probabilitas kelas prediksi. Hasil dalam bentuk diskrit, direpresentasikan dalam bentuk *confusion matrix* (Tharwat, 2018).

Tabel 2.1 merupakan *confusion matrix* untuk kasus *binary classification.*

Tabel 2.2 Confusion Matrix untuk Binary Classification

Kelas Aktual	Kelas Prediksi	
	Positif	Negatif
Positif	TP	FN
Negatif	FP	TN

Kinerja klasifikasi dapat dievaluasi dari nilai TP, FP, FN, dan TN. TP (*true positive*) adalah banyaknya pengamatan yang berkategori positif yang tepat diklasifikasikan ke kategori positif, FP (*false positive*) adalah banyaknya pengamatan yang memiliki kategori positif yang diklasifikasikan ke kategori negatif, TN (*true negative*) adalah banyaknya pengamatan yang berkategori negatif yang tepat diklasifikasikan ke kategori negatif, dan FN (*false negative*) adalah banyaknya pengamatan yang berkategori negatif yang diklasifikasikan ke kategori negatif (Tharwat, 2018). Untuk kasus *multiclass classification* pada penelitian ini dapat dilihat melalui Tabel 2.2.

Tabel 2.3 *Confusion Matrix*

Kelas Aktual	Kelas Prediksi					Total
	C_1	C_2	C_3	...	C_K	
C_1	n_{11}	n_{12}	n_{13}	...	n_{1k}	n_1
C_2	n_{21}	n_{22}	n_{23}	...	n_{2k}	n_2
⋮	⋮	⋮	⋮	⋮	⋮	⋮
C_K	n_{K1}	n_{K2}	n_{K3}	...	n_{KK}	n_K
Total	n_1	n_2	n_3	...	n_K	N

Penilaian kinerja klasifikasi untuk *multiclass classification* didefinisikan perkelas secara individual C_k dimana k = 1,2, …, K dengan K merupakan banyak kelas yang digunakan. Kualitas kinerja klasifikasi untuk keselutuhan dapat dihitung melalui rata-rata kinerja klasifikasi yang didapatkan dari tiap kelasnya (Sokolova & Lapalme, 2009). Pada kasus *multiclass* kinerja klasifikasi dapat diukur dengan menggunakan *accuracy, precision, sensitivity,* dan *Fscore*. *Accuracy* adalah banyak pengamatan yang

terklasifikasi secara tepat. *Fscore* didapatkan dari nilai kombinasi antara *precision* dan *sensitivity*. *Precision* adalah banyaknya pengamatan yang tepat terprediksi positif dari keseluruhan dengan hasil prediksi positif, *sensitivity* adalah banyaknya pengamatan yang tepat diklasifikasikan sesuai kategorinya (Sokolova & Lapalme, 2009). Perhitungan *accuracy*, *precision*, *sensitivity*, dan *Fscore* dapat dilakukan dengan menggunakan persamaan (2.9), (2.10), (2.11), dan (2.12).

$$accuracy = \frac{\sum_{k=1}^{K} \frac{TP_k + TN_k}{TP_k + FN_k + FP_k + TN_k}}{K}, \tag{2.14}$$

$$precision = \frac{\sum_{k=1}^{K} \frac{TP_k}{TP_k + FP_k}}{K}, \tag{2.15}$$

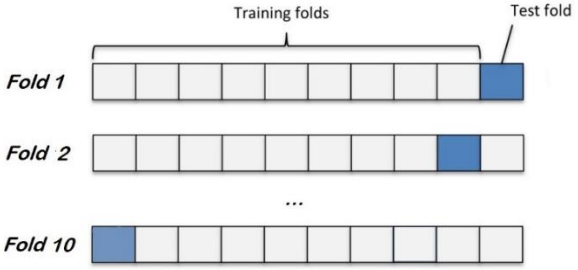
$$sensitivity = \frac{\sum_{k=1}^{K} \frac{TP_k}{TP_k + FN_k}}{K}, \tag{2.16}$$

$$Fscore = \frac{2(precision \times sensitivity)}{precision + sensitivity}. \tag{2.17}$$

2.6 K-*folds Cross Validation* (KCV)

Metode K-*folds Cross Validation* (KCV) merupakan suatu metode yang dapat diandalkan (*reliable*) untuk memprediksi kesalahan dalam suatu klasifikasi. Metode ini banyak digunakan oleh peneliti untuk mengurangi bias yang terjadi karena pengambilan sampel data yang akan digunakan. KCV secara berulang-ulang membagi data menjadi data *training* dan data *testing*, dimana setiap data berkesempatan menjadi data *testing* (Gokgoz & Subasi, 2015). Nilai k yang sering digunakan adalah 10, karena merupakan nilai yang paling memadai untuk mendapatkan perkiraan kesalahan terbaik (Berthold & Hand, 2010). Data dibagi menjadi 10 bagian, dimana 9 bagian digunakan

sebagai data *training* dan 1 bagian lainya menjadi data *testing*, kemudian dilakukan pengulangan hingga 10 kali, sehingga setiap data berkesempatan menjadi data *training* maupun data *testing*. Ilustrasi pembagian data menggunakan metode validasi ini terdapat pada Gambar 2.9.



Gambar 2.9 Ilustrasi Prosedur *10-fold cross validation*

BAB III
METODOLOGI PENELITIAN

3.1 Sumber Data

Pada penelitian ini data yang digunakan adalah data primer, diambil dari hasil foto *webcam* berdimensi 640 × 480 *pixel*. Pengambilan data ini diambil pada hari Selasa, tanggal 9 April 2019 pada pukul 12.30 – 15.00 di Ruang T103B Departemen Statistika ITS Surabaya. Data yang digunakan adalah citra foto 4 orang. Setiap orang diambil 10 foto dari sisi yang berbeda. Data foto yang digunakan terlampir pada Lampiran 1. Data yang didapatkan, selanjutnya dilakukan *resize* menjadi 160 × 120 serta mengganti tipe citra dari RGB menjadi *grayscale.* Semakin banyak input citra yang digunakan maka akan memberikan *learning* lebih baik dalam pembuatan model. Oleh karena itu, dilakukan augmentasi citra atau perbanyakan data. Contoh augmentasi citra yang digunakan tercantum pada Gambar 3.1.

Gambar 3.1 Augmentasi Citra

3.2 Variabel Penelitian

Variabel penelitian yang digunakan tercantum pada Tabel 3.1.

Tabel 3.1 Variabel Penelitian

Variabel	Keterangan	Skala
$X_{i,n}$	Nilai *grayscale* setiap *pixel*	Interval
Y_k	Identitas orang yang digunakan: Y_1 : Orang pertama Y_2 : Orang kedua Y_3 : Orang ketiga Y_4 : Orang keempat	Nominal

Dengan:
$0 < i \leq n_x$; n_x berjumlah 19.200 merupakan total *pixel* ($160 \times 120 = 19.200$) dari setiap gambar,
$0 < n \leq N$; $N = 320 = 4 \times 10 \times (1+7)$,
dengan 4 merupakan banyaknya orang yang difoto masing-masing 10 kali serta augmentasi data sebanyak 7.

3.3 Struktur Data

Struktur data secara umum yang digunakan dalam penelitian ini yang terdapat pada Tabel 3.2. Observasi yang digunakan berupa nilai *grayscale* pada setiap *pixel*.

Tabel 3.2 Struktur Data

Orang ke-	No	X_1	X_2	…	X_{19200}	Y_k
1	1	$X_{1,1}$	$X_{2,1}$	…	$X_{19200,1}$	Y_1
1	2	$X_{1,2}$	$X_{2,2}$	…	$X_{19200,2}$	Y_1
…	…	…	…	…	…	…
1	80	$X_{1,80}$	$X_{2,80}$	…	$X_{19200,80}$	Y_1
2	81	$X_{1,81}$	$X_{2,81}$	…	$X_{19200,81}$	Y_2
…	…	…	…	…	…	…
2	160	$X_{1,160}$	$X_{2,160}$	…	$X_{19200,160}$	Y_2
3	161	$X_{1,161}$	$X_{2,161}$	…	$X_{19200,161}$	Y_3
…	…	…	…	…	…	…
3	240	$X_{1,240}$	$X_{2,240}$	…	$X_{19200,240}$	Y_3
4	241	$X_{1,241}$	$X_{2,241}$	…	$X_{19200,241}$	Y_4
…	…	…	…	…	…	…
4	320	$X_{1,320}$	$X_{2,320}$	…	$X_{19200,320}$	Y_4

3.4 Langkah Analisis

Langkah-langkah penelitian yang telah dilakukan berdasarkan dengan tujuan penelitian ini adalah sebagai berikut.

1. Mengumpulkan data dengan menggunakan hasil foto dari *webcam*. Proses dari gambar untuk diubah dalam bentuk data adalah sebagai berikut.
    a. Gambar terdiri dari panjang dan lebar yang diwakilkan oleh kotak-kotak *pixel* yang tersusun mengikuti ukuran gambar.
    b. Setiap *pixel* mewakilkan 1 warna.
    c. Warna terbentuk dari kombinasi tiga komponen warna utama yaitu R (*Red*), G (*Green*), dan B (*Blue*). Setiap komponen warna utama memiliki tingkat gradasi yang memiliki nilai antara 0-255, dengan semakin kecil nilai maka warna yang dihasilkan akan semakin gelap.
    d. Dari kombinasi komponen warna utama tersebut, diperoleh sebanyak 255^3 kombinasi warna yang dibentuk pada komputer.
    e. Komputer menangkap 3 nilai pada komponen warna utama pada masing-masing *pixel* pembentuk gambar, jika ketiga komponen dipisahkan maka akan didapatkan 3 *matrix* yang masing-masing berisi nilai R, G, dan B dengan ukuran kolom *matrix* adalah panjang gambar dan baris *matrix* adalah lebar gambar.
    f. Sehingga data yang didapatkan adalah 3 *matrix* dengan ukuran panjang dan lebar gambar (dalam penelitian ini berukuran 640×480) yang berisi nilai antara 0-255.
2. *Preprocessing* data

    Tahap *preprocessing* data meliputi:
    a. Mengubah citra RGB ke *grayscale.* Pengubahan dari RGB ke *grayscale* tercantum pada persamaan (2.1).
    b. *Resize* citra dari 640×480 menjadi 160×120. Langkah-langkah *resize* citra dalam penelitian ini adalah sebagai berikut.

1. Menentukan perubahan ukuran citra dari 640×480 menjadi 160×120 yaitu 0,25 kali ukuran semula.
2. Nilai pada *pixel* dengan ukuran 160×120 sama dengan nilai pada *pixel* kelipatan satu per 0,25 dari ukuran semula.
3. Memperbanyak data dengan augmentasi citra yang tercantum pada Gambar 3.1 untuk setiap gambarnya.

3. Membagi data *training* dan data *testing* dengan menggunakan K-*folds cross validation*.
4. Melakukan klasifikasi dengan menggunakan metode *Multilayer Perceptron* (MLP) dengan langkah-langkah sebagai berikut.
   1. Menentukan jumlah *hidden layer* dan *neuron* yang digunakan.
   2. Melakukan inisialisasi parameter bias dan bobot.
   3. Melakukan perhitungan dengan persamaan (2.2), (2.3), (2.4), dan (2.5).
   4. Menghitung nilai *loss function* sesuai persamaan (2.6).
   5. Jika nilai *loss* yang didapatkan kurang dari nilai minimum yang ditentukan yaitu 0,05 maka lanjut ke langkah 9. Jika tidak, maka lanjut ke langkah 6.
   6. Jika iterasi yang dilakukan lebih dari 300 maka lanjut ke langkah 9. Jika tidak maka lanjut ke langkah 7.
   7. Melakukan *update* parameter (bias dan bobot) sesuai dengan subbab 2.3.2.
   8. Parameter yang telah di-*update* merupakan parameter baru yang digunakan untuk perhitungan kembali ke langkah 3,4, dan 5 untuk iterasi selanjutnya.
   9. Iterasi *training* selesai. Model, bias, bobot, serta nilai *loss* yang didapatkan disimpan lalu dicobakan pada data *testing*.
   10. Menghitung nilai kinerja klasifikasi baik *accuracy*, *precision, sensitivity,* dan *Fscore* pada hasil klasifikasi data *testing.*

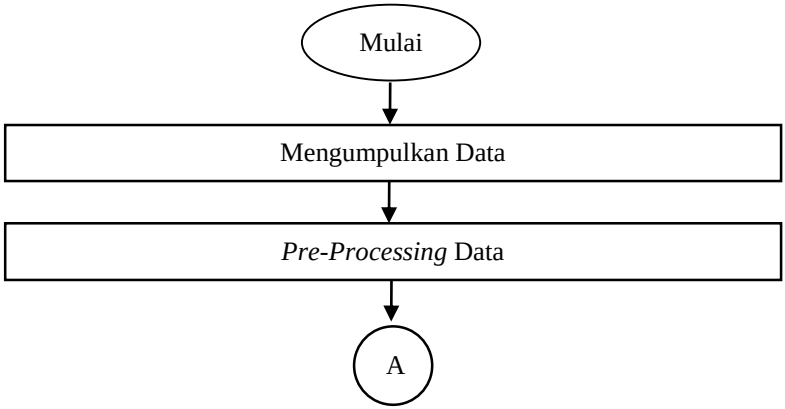
11. Melakukan langkah 2 hingga 10 pada *fold* yang lain sehingga didapatkan 10 nilai masing-masing kinerja klasifikasi.
12. Menghitung nilai rata-rata dari semua hasil kinerja klasifikasi.

5. Melakukan klasifikasi dengan menggunakan metode *Convolutional Neural Network* (CNN) dengan langkah-langkah sebagai berikut.
    1. Menentukan ukuran *convolution kernel* dan ukuran *pooling* yang digunakan.
    2. Melakukan inisialisasi parameter bias dan bobot.
    3. Melakukan perhitungan operasi konvolusi sesuai dengan persamaan (2.12).
    4. Menghitung fungsi aktivasi ReLU sesuai dengan persamaan (2.13).
    5. Melakukan operasi *max-pooling* sesuai dengan Gambar 2.8.
    6. Menjadikan keseluruhan *pixel* yang didapatkan dari langkah 5 ke dalam 1 vektor.
    7. Melakukan perhitungan dengan persamaan (2.2), (2.3), (2.4), (2.5) dengan input merupakan vektor dari langkah 6 dengan jumlah *neuron* pada *hidden layer* adalah 10.
    8. Jika nilai *loss* yang didapatkan kurang dari nilai minimum yang ditentukan yaitu 0,05 maka lanjut ke langkah 12. Jika tidak, maka lanjut ke langkah 9.
    9. Jika iterasi yang dilakukan lebih dari 300 maka lanjut ke langkah 12. Jika tidak maka lanjut ke langkah 10.
    10. Melakukan *update* parameter (bias dan bobot) sesuai dengan subbab 2.3.2.
    11. Parameter yang telah di-*update* merupakan parameter baru yang digunakan untuk perhitungan kembali ke langkah 3 sampai dengan 7 untuk iterasi selanjutnya.
    12. Iterasi *training* selesai. Model, bias, bobot, serta nilai *loss* yang didapatkan disimpan lalu dicobakan pada data *testing*.

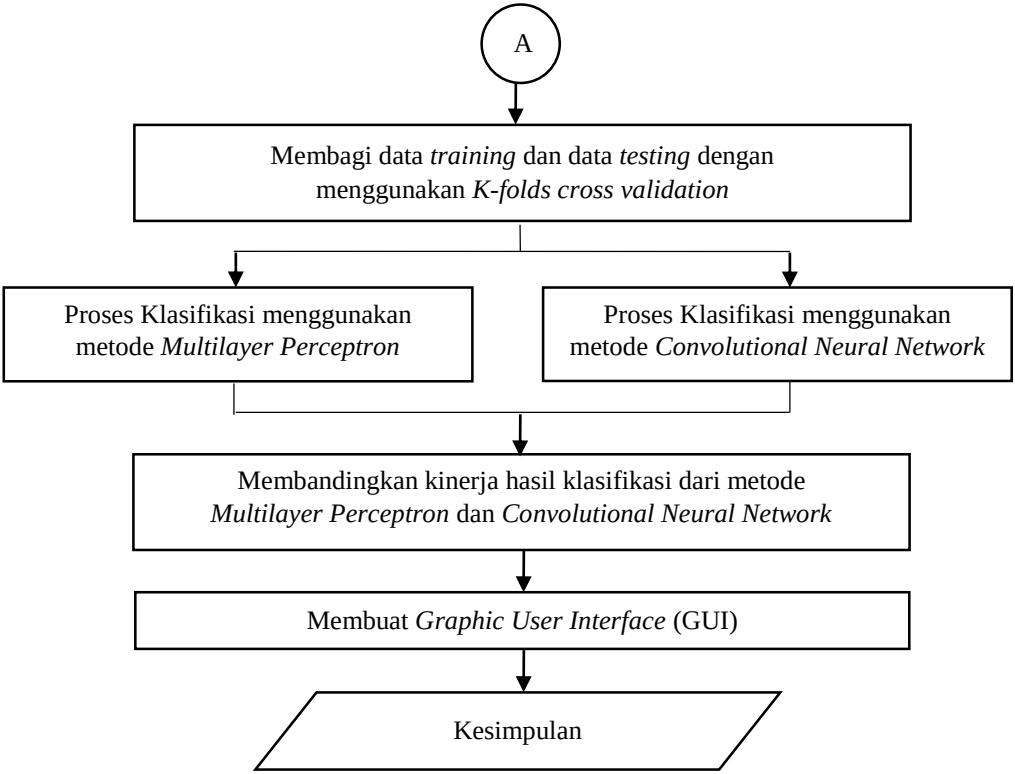
13. Menghitung nilai kinerja klasifikasi baik *accuracy*, *precision*, *sensitivity*, dan *Fscore* pada hasil klasifikasi data *testing.*
14. Melakukan langkah 2 hingga 13 pada *fold* yang lain sehingga didapatkan 10 nilai masing-masing kinerja klasifikasi.
15. Menghitung nilai rata-rata dari semua hasil kinerja klasifikasi

6. Membandingkan kinerja dari metode MLP dan CNN yang telah didapatkan baik *accuracy*, *precision*, *sensitivity*, dan *Fscore*.
7. Membuat program atau *graphic user interface* (GUI) dengan menggunakan metode terbaik yang telah didapatkan.
8. Mengambil kesimpulan dan saran.

Langkah-langkah analisis secara umum digambarkan pada diagram alir pada Gambar 3.2.



Gambar 3.2 Diagram Alir



Gambar 3.2 Diagram Alir (Lanjutan)

(Halaman ini sengaja dikosongkan)

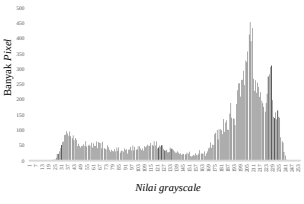
BAB IV
ANALISIS DAN PEMBAHASAN

Pada penelitian ini dilakukan klasifikasi identitas orang berdasarkan nilai *grayscale* setiap *pixel*. Penelitian ini dibandingkan dua metode yaitu *Multilayer Perceptron* (MLP) dan *Convolutional Neural Network* (CNN). Kebaikan hasil klasifikasi tersebut didapatkan dari hasil nilai *accuracy, precision, sensitivity* dan *Fscore.*

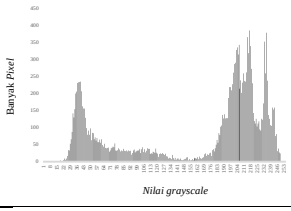
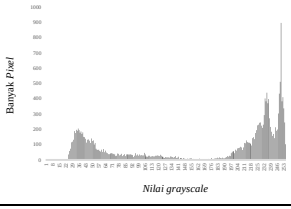
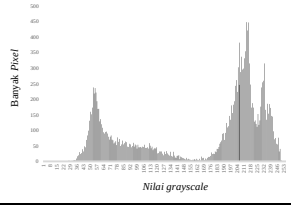
4.1 Deskripsi Data Hasil *Preprocessing*

Tahapan *preprocessing* data adalah dengan menggunakan mengubah citra RGB menjadi *grayscale*. Setelah itu *resize* citra menjadi 160 × 120. *Matrix* berukuran 160 × 120 dijadikan vektor yang berukuran 19200. Vektor tersebut berisi nilai *grayscale* (0-255) dari 1 citra. Selanjutnya dihitung banyaknya *pixel* setiap nilai *grayscale* pada citra tersebut. Dari hasil perhitungan tersebut, dibuat *histogram* citra. Sehingga *histogram* citra terbentuk dari banyaknya *pixel* yang memiliki nilai *grayscale*, dengan sumbu horizontal dari *histogram* adalah nilai dari 0 – 255 sedangkan sumbu vertikal adalah banyak *pixel* pada nilai *grayscale* tersebut. Begitu seterusnya untuk keseluruhan citra. Bentuk *histogram* dari salah satu citra masing-masing kelas yang digunakan terdapat pada Tabel 4.1.

Tabel 4.1 *Histogram* Salah Satu Citra Setiap Kelas

Y_k	Citra	Histogram
Y_1		

Tabel 4.1 *Histogram* Salah Satu Citra Setiap Kelas (Lanjutan)

Y_k	Citra	Histogram
Y_2		
Y_3		
Y_4		

Dari Tabel 4.1 diatas dapat dilihat bahwa *histogram* yang tergambarkan antara satu foto dengan yang lain tidak jauh berbeda. *Histogram* itu terlihat memiliki 2 pola yang terbagi atas warna hitam hingga abu-abu dan warna abu-abu hingga putih. Hal ini diakibatkan karena digunakannya penggunaan *background.* Namun dalam penelitian ini, penggunaan *background* juga digunakan dalam *training*.

4.2 Klasifikasi Wajah

Setelah dilakukan *preprocessing* data, selanjutnya akan dilakukan pembuatan model untuk mengklasifikasikan citra tersebut menurut identitas orang yang telah ditentukan. Metode

klasifikasi yang digunakan adalah metode *Multilayer Perceptron* (MLP) dan *Convolutional Neural Network* (CNN).

4.2.1 *Multilayer Perceptron (MLP)*

Dalam metode *Multilayer Perceptron* (MLP) akan dilakukan perbandingan hasil kinerja klasifikasi berdasarkan jumlah *hidden layer* yang digunakan dan jumlah *neurons* pada setiap *hidden layer*. Dari hasil *pre-processing* data, didapatkan 19.200 variabel input. Percobaan yang dilakukan adalah dilakukan pada 1 *hidden layer* dan 2 *hidden layer* dengan masing-masing *neurons* memiliki jumlah berkisar antara 10-200 unit. Variabel *output* yaitu sebanyak 4, sama dengan jumlah orang yang diambil. Fungsi aktivasi pada *output* yang digunakan *softmax* dengan optimasi parameter yaitu *Adam*. Dalam tahap *training*, proses iterasi akan berhenti jika nilai *loss* yang dihasilkan bernilai kurang dari minimum *loss* yang ditetapkan yaitu 0,05 atau lebih dari jumlah iterasi (*epoch*) maksimum yang ditetapkan yaitu 300. Setelah iterasi berhenti maka model yang dan parameter yang didapatkan akan menjadi model pada data *testing*. Pembagian data *training* dan *testing* dilakukan dengan menggunakan metode K-*folds cross validation* Metode ini digunakan untuk mengurangi bias dalam proses pembagian menjadi data *training* dan *testing* dengan cara membagi data kedalam nilai yang disebut *fold*. Pada penelitian ini digunakan 10-*folds cross validation* yaitu perbandingan antara jumlah *training* dan *testing* yaitu 288:32.

Hasil rata-rata kinerja klasifikasi yang didapat dalam sepuluh *fold* dengan satu dan dua *hidden layer* didapatkan pada Tabel 4.2.

Tabel 4.2 Kinerja Klasifikasi MLP

neurons	1 *hidden layer*				2 *hidden layer*			
	Accuracy	*Precision*	*Sensitivity*	*Fscore*	*Accuracy*	*Precision*	*Sensitivity*	*Fscore*
10	0,8813	0,8877	0,8813	0,8844	0,9125	0,9168	0,9125	0,9147
20	0,9094	0,9149	0,9094	0,9121	0,9313	0,9375	0,9313	0,9343
30	0,9188	0,9235	0,9188	0,9211	0,9313	0,9368	0,9313	0,9340
40	0,3906	0,1743	0,3906	0,2372	0,9156	0,9204	0,9156	0,9180
50	0,9281	0,9323	0,9281	0,9302	0,9250	0,9319	0,9250	0,9284

Tabel 4.2 Kinerja Klasifikasi MLP (Lanjutan)

neurons	1 *hidden layer*				2 *hidden layer*			
	Accuracy	*Precision*	*Sensitivity*	*Fscore*	*Accuracy*	*Precision*	*Sensitivity*	*Fscore*
60	0,9313	0,9371	0,9313	0,9342	0,9281	0,9338	0,9281	0,9309
70	0,9313	0,9384	0,9313	0,9348	0,9250	0,9309	0,9250	0,9279
80	0,9313	0,9371	0,9313	0,9342	0,9250	0,9309	0,9250	0,9279
90	**0,9406**	**0,9463**	**0,9406**	**0,9434**	0,9281	0,9341	0,9281	0,9311
100	0,9281	0,9352	0,9281	0,9317	0,9313	0,9375	0,9313	0,9343
110	0,9344	0,9406	0,9344	0,9375	0,9250	0,9309	0,9250	0,9279
120	0,9219	0,9272	0,9219	0,9245	0,9344	0,9402	0,9344	0,9373
130	0,9313	0,9380	0,9313	0,9346	0,9250	0,9317	0,9250	0,9283
140	0,9344	0,9401	0,9344	0,9372	0,9281	0,9338	0,9281	0,9309
150	0,9250	0,9307	0,9250	0,9278	0,9250	0,9306	0,9250	0,9278
160	0,9344	0,9412	0,9344	0,9377	0,9313	0,9372	0,9313	0,9342
170	0,9281	0,9336	0,9281	0,9308	0,9281	0,9349	0,9281	0,9315
180	0,4969	0,2490	0,4969	0,3318	0,9313	0,9372	0,9313	0,9342
190	0,9219	0,9297	0,9219	0,9257	0,9281	0,9349	0,9281	0,9315
200	0,9281	0,9328	0,9281	0,9305	**0,9344**	**0,9415**	**0,9344**	**0,9379**

Kinerja klasifikasi pada Tabel 4.2 didapat dari rata-rata nilai *testing* untuk *accuracy*, *precision*, *sensitivity*, dan *Fscore* pada setiap kelas dibagi dengan jumlah kelasnya untuk kesepuluh *fold*. Dari Tabel 4.2 dapat terlihat bahwa tidak selalu lebih banyak *neurons* pada *hidden layer* mendapatkan kinerja klasifikasi yang semakin besar pula. Kinerja klasifikasi terbaik pada 1 *hidden layer* merupakan jumlah *neurons* sebanyak 90. Kinerja klasifikasi pada jumlah *neurons* tersebut yaitu *accuracy* dengan 0,9406, *precision* yaitu 0,9463, *sensitivity* senilai 0,9406, dan *Fscore* sebesar 0,9434. Berbeda dengan 2 *hidden layer*, hasil kinerja klasifikasi terbaik terdapat dengan *neurons* sebanyak 200 pada masing-masing *hidden layer*. Kinerja klasifikasi terbaik untuk 2 *hidden layer* turun menjadi *accuracy* sebesar 0,9344, *precision* yaitu 0,9415, *sensitivity* sebesar 0,9344, dan *Fscore* bernilai 0,9379. Dari kasus dan penelitian ini mengindikasikan bahwa semakin banyak *hidden layer* tidak selalu dapat menambahkan kinerja klasifikasi. Meskipun kinerja klasifikasi pada 1 *hidden layer* lebih besar daripada 2 *hidden layer*, namun hasil yang didapatkan tidak

berbeda signifikan. Selain itu nilai akurasi yang didapatkan lebih dari 90% mengindikasikan bahwa hasil yang didapatkan sangat bagus (*excellent*) dalam klasifikasi wajah.

Selanjutnya dilakukan pemeriksaan pada setiap *fold* untuk mendapatkan kinerja klasifikasi pada setiap *fold*. Tabel 4.3 merupakan hasil kinerja klasifikasi setiap *fold* pada 1 *hidden layer* untuk 90 *neurons*.

Tabel 4.3 Kinerja Klasifikasi MLP setiap *fold*

fold	***Accuracy***	***Precision***	***Sensitivity***	***Fscore***
1	0,9688	0,9722	0,9688	0,9705
2	0,9375	0,9444	0,9375	0,9410
3	0,9063	0,9087	0,9063	0,9075
4	0,9688	0,9722	0,9688	0,9705
5	0,9688	0,9722	0,9688	0,9705
6	**1,0000**	**1,0000**	**1,0000**	**1,0000**
7	0,9375	0,9444	0,9375	0,9410
8	0,9063	0,9222	0,9063	0,9142
9	0,8750	0,8819	0,8750	0,8785
10	0,9375	0,9444	0,9375	0,9410
Rata-rata	0,9406	0,9463	0,9406	0,9434

Tabel 4.3 menunjukkan nilai kinerja klasifikasi terbaik untuk *fold* ke-6 adalah *fold* dengan nilai kinerja klasifikasi *testing* terbesar yaitu 100%. Hal ini berarti semua data *testing* terklasifikasi secara tepat untuk pembagian data *training* dan *testing* pada *fold* ke-6 terdapat pada Lampiran. Dari hasil tersebut, maka didapatkan model persamaan yang digunakan pada data baru atau ingin diketahui targetnya. Persamaan ini terdiri dari persamaan antar layer dan persamaan fungsi aktivasi. Berikut merupakan persamaan antara *layer input* untuk masing-masing *neurons* pada *hidden layer* berdasarkan persamaan (2.2).

$$p_1 = -0,02635x_1 + 0,01238x_2 - 0,02522\ x_3 + ... - 0,03239\ x_{19200},$$
$$p_2 = -0,00495 - 0,00241\ x_1 - 0,00645\ x_2 - 0,0052\ x_3 + ... - 0,00194\ x_{19200},$$
$$\vdots$$
$$p_{90} = -0,02887\ x_1 - 0,04364\ x_2 - 0,00854\ x_3 + ... - 0,02806\ x_{19200}.$$

Selanjutnya persamaan untuk setiap *neurons* pada *hidden layer* dengan fungsi aktivasi ReLU berdasarkan persamaan (2.3) yakni sebagai berikut.

$$q_1 = f_h(p_1) = \max(0, p_1) = \begin{cases} p_1, & \text{jika } p_1 \geq 0, \\ 0, & \text{jika } p_1 < 0, \end{cases}$$

$$q_2 = f_h(p_2) = \max(0, p_2) = \begin{cases} p_2, & \text{jika } p_2 \geq 0, \\ 0, & \text{jika } p_2 < 0, \end{cases}$$

$$\vdots$$

$$q_{90} = f_h(p_{90}) = \max(0, p_{90}) = \begin{cases} p_{90}, & \text{jika } p_{90} \geq 0, \\ 0, & \text{jika } p_{90} < 0, \end{cases}$$

Persamaan pada *output layer* sebelum dimasukkan fungsi aktivasi berdasarkan persamaan (2.4) adalah sebagai berikut

$$r_1 = -0{,}02427 - 0{,}02635\ q_1 - 0{,}03152\ q_2 + \ldots - 0{,}0173\ q_{90},$$
$$r_2 = -0{,}00371 + 0{,}00712\ q_1 - 0{,}00879 q_2 + \ldots + 0{,}04415\ q_{90},$$
$$r_3 = -0{,}01399 + 0{,}00177\ q_1 + 0{,}00679\ q_2 + \ldots - 0{,}00791\ q_{90},$$
$$r_4 = 0{,}03028 + 0{,}03057\ q_1 - 0{,}01641 q_2 + \ldots - 0{,}03878\ q_{90}.$$

Sehingga persamaan untuk setiap *neurons* pada *output layer* dengan fungsi aktivasi *softmax* berdasarkan persamaan (2.5) adalah sebagai berikut.

$$\hat{y}_1 = f_o(r_1) = \frac{\exp(r_1)}{\exp(r_1) + \exp(r_2) + \exp(r_3) + \exp(r_4)},$$

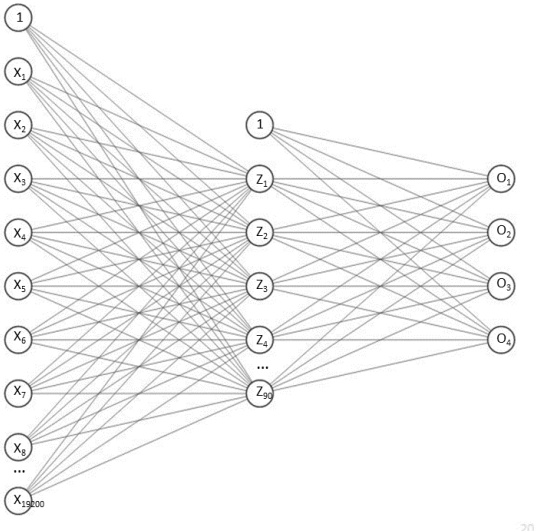
$$\hat{y}_2 = f_o(r_2) = \frac{\exp(r_2)}{\exp(r_1) + \exp(r_2) + \exp(r_3) + \exp(r_4)},$$

$$\hat{y}_3 = f_o(r_3) = \frac{\exp(r_3)}{\exp(r_1) + \exp(r_2) + \exp(r_3) + \exp(r_4)},$$

$$\hat{y}_4 = f_o(r_4) = \frac{\exp(r_4)}{\exp(r_1) + \exp(r_2) + \exp(r_3) + \exp(r_4)}.$$

Hasil dari persamaan tersebut bernilai antara 0 hingga 1. Sehingga untuk menentukan kelas prediksi yang dihasilkan (kelas *k*), maka dilihat dari nilai $\hat{y}_k$ maksimum.

Parameter atau pembobot yang dihasilkan pada persamaan tersebut berjumlah 1.728.454 (didapatkan dari jumlah parameter antara *input layer* dan *hidden layer* sebanyak 1.728.090 dan jumlah parameter antara *hidden layer* dan *output layer* berjumlah 364). Dari hasil terbaik tersebut maka digambarkan ilustrasi jaringan *Multilayer Perceptron* (MLP) pada Gambar 4.1.



Gambar 4.1 Ilustrasi Model MLP 1 *hidden layer*

4.2.2 *Convolutional Neural Network*

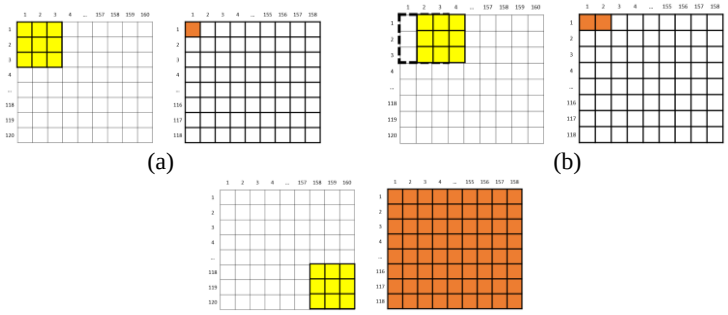
Setelah dilakukan klasifikasi dengan metode *Multilayer Perceptron* (MLP), maka selanjutnya dilakukan klasifikasi dengan metode *Convolutional Neural Network* (CNN). Metode CNN merupakan pengembangan dari MLP yang khusus digunakan dalam proses data citra. Pada metode CNN ini dilakukan perbandingan hasil kinerja klasifikasi, dimana perbandingan tersebut adalah ukuran *kernel* pada *convolutional layer* dan *pooling*

layer. Ukuran ini dicobakan antara 2×2 sampai 11×11 untuk setiap *convolution kernel* dan *pooling kernel* sehingga didapatkan 11 model yang disajikan pada Tabel 4.4.

Tabel 4.4 Percobaan Metode CNN

Model	***Convolution Kernel***	***Pooling Kernel***
Model 1	3 × 3	2 × 2
Model 2	5 × 5	2 × 2
Model 3	5 × 5	4 × 4
Model 4	6 × 6	5 × 5
Model 5	7 × 7	2 × 2
Model 6	9 × 9	2 × 2
Model 7	9 × 9	4 × 4
Model 8	9 × 9	8 × 8
Model 9	11 × 11	2 × 2
Model 10	11 × 11	5 × 5
Model 11	11 × 11	10 × 10

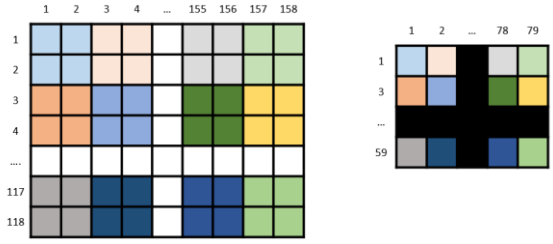
Tabel 4.4 ini digunakan atas dasar hasil reduksi dimensi yang tidak menghilangkan *pixel* pada gambar. Ilustrasi salah satu model yaitu model pertama dengan *convolution kernel* berukuran 3 × 3 tercantum pada Gambar 4.2.



Gambar 4.2 Ilustrasi Proses *Convolution* : (a) Pertama, (b) Kedua, dan (c) Terakhir

Setelah proses konvolusi, maka diakukan fungsi aktivasi ReLU kemudian dimasukkan ke dalam *pooling layer*. Berikut merupakan ilustrasi dari *pooling layer*. Setiap warna pada Gambar

4.3(b) merupakan nilai maksimum dari warna yang sama pada Gambar 4.3a(a).



(a) (b)

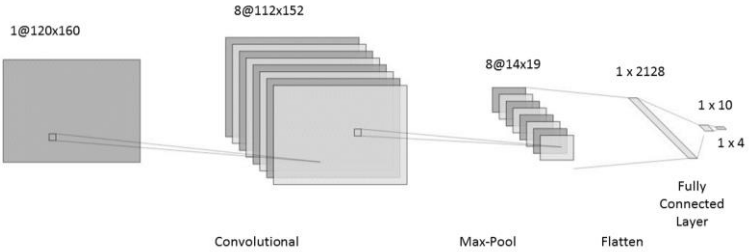
Gambar 4.3 Proses *pooling layer*: (a) *feature map* dan (b) hasil *max-pooling*

Pada CNN ini digunakan jumlah *neurons* pada *fully connected layer* yaitu 10 *neurons* dan *layer output* yaitu 4 dengan fungsi aktivasi *softmax*. Fungsi aktivasi ReLU digunakan setelah *convolutional layer*. Sama dengan klasifikasi wajah dengan MLP, proses iterasi *training* akan berhenti jika nilai *loss* yang dihasilkan bernilai kurang dari minimum *loss* yang ditetapkan yaitu 0,05 atau lebih dari jumlah iterasi (*epoch*) maksimum yang ditetapkan yaitu 300. Setelah iterasi berhenti maka model yang dan parameter yang didapatkan akan menjadi model pada data *testing*. Hasil kinerja klasifikasi yang didapatkan pada metode CNN yang disajikan dalam Tabel 4.5.

Tabel 4.5 Kinerja Klasifikasi CNN

Model	***Accuracy***	***Precision***	***Sensitivity***	***Fscore***
Model 1	0,9250	0,9336	0,9250	0,9292
Model 2	0,9344	0,9388	0,9344	0,9366
Model 3	0,9313	0,9367	0,9313	0,9340
Model 4	0,8844	0,8619	0,8844	0,8717
Model 5	0,9219	0,9289	0,9219	0,9254
Model 6	0,9063	0,9141	0,9063	0,9102
Model 7	0,9281	0,9333	0,9281	0,9307
Model 8	**0,9406**	**0,9473**	**0,9406**	**0,9439**
Model 9	0,8969	0,9059	0,8969	0,9014
Model 10	0,9375	0,9436	0,9375	0,9405
Model 11	0,9375	0,9426	0,9375	0,9400

Tabel 4.5 menunjukkan rata-rata kinerja klasifikasi yang didapatkan dari setiap *fold.* Kinerja klasifikasi yang terbaik merupakan model 8 dengan *convolutional kernel* berukuran 9×9 dan *pooling kernel* berukuran 8×8. Hasil ini mendapatkan *accuracy* sebesar 0,9406, *precision* sebesar 0,9473, *sensitivity* sebesar 0,9406 dan *Fscore* senilai 0,9439. Perbanyakan jumlah ukuran *kernel* pada setiap *layer* tidak mengindikasikan bahwa adanya penambahan kinerja klasifikasi. Gambar 4.4 merupakan ilustrasi model CNN dengan kinerja klasifikasi terbaik.



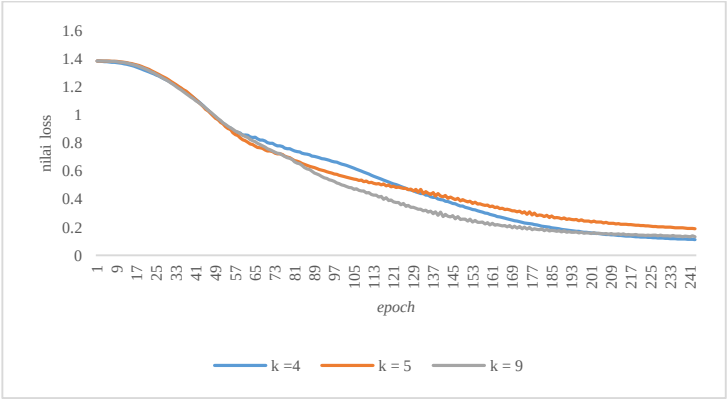
Gambar 4.4 Ilustrasi Model *Convolutional Neural Network*

Selanjutnya dilakukan pemeriksaan pada setiap *fold* untuk mendapatkan kinerja klasifikasi pada setiap *fold.* Tabel 4.6 merupakan hasil kinerja klasifikasi setiap *fold* dengan metode CNN.

Tabel 4.6 Kinerja Klasifikasi CNN setiap *fold*

fold	***Accuracy***	***Precision***	***Sensitivity***	***Fscore***
1	0,9375	0,9375	0,9375	0,9375
2	0,9375	0,9500	0,9375	0,9437
3	0,9375	0,9500	0,9375	0,9437
4	**0,9688**	**0,9722**	**0,9688**	**0,9705**
5	**0,9688**	**0,9722**	**0,9688**	**0,9705**
6	0,9063	0,9318	0,9063	0,9189
7	0,9375	0,9444	0,9375	0,9410
8	0,9063	0,9053	0,9063	0,9058
9	**0,9688**	**0,9722**	**0,9688**	**0,9705**
10	0,9375	0,9375	0,9375	0,9375
Rata-rata	0,9406	0,9473	0,9406	0,9439

Pada Tabel 4.6 didapatkan bahwa *fold* terbaik berjumlah 3 *fold* yaitu *fold* ke-4, 5, dan 9. Untuk mendapatkan model terbaik maka dilanjutkan dengan melihat nilai *loss* pada *fold* tersebut untuk menentukan nilai *fold* yang memiliki nilai *loss* minimum. Grafik *loss* setiap *epoch* tercantum pada Gambar 4.5 dan nilai *loss* untuk *fold* tersebut tercantum pada Tabel 4.7.



Gambar 4.5 Grafik *loss* setiap *epoch*

Tabel 4.7 Nilai *Loss* untuk *Fold* ke-4,5, dan 9

fold	***Loss***
4	**0,1123**
5	0,1898
9	0,1331

Gambar 4.5 merupakan grafik *loss* untuk *fold* ke-4, 5, dan 9. Pada grafik tersebut dapat dilihat bahwa nilai *loss* untuk *epoch* ke-243 mendapatkan nilai *loss* pada *fold* ke-4 lebih kecil daripada *fold* ke-5 dan ke-9. Dari Tabel 4.7 juga dapat dilihat bahwa *fold* ke-4 memiliki nilai *loss* yang paling kecil dibanding *fold* ke-5 dan ke-9 yaitu senilai 0,1123. Sehingga *fold* terbaik dengan metode *Convolutional Neural Network* dengan *convolutional filter* berupa 9×9 dan *pooling filter* berukuran 8×8 adalah *fold* ke-4.

4.2.3 Perbandingan Kinerja Metode MLP dan CNN

Perbandingan kinerja metode *Multilayer Perceptron* (MLP) dan *Convolutional Neural Network* (CNN) dapat dilihat dari hasil kinerja klasifikasi dengan perulangan beberapa kali. Dalam penelitian ini dilakukan perulangan sebanyak 10 kali. Model yang dibandingkan yaitu metode MLP dengan 1 *hidden layer* 90 *neurons* pada *fold* ke-6 dan metode CNN model 8 pada *fold* ke-4. Dari perulangan tersebut maka didapatkan nilai rata-rata *accuracy*, *precision*, *sensitivity*, dan *Fscore* untuk data *testing* sehingga didapatkan metode dengan kinerja terbaik. Hasil rata-rata kinerja klasifikasi kedua metode dapat dilihat pada Tabel 4.8.

Tabel 4.8 Perbandingan Kinerja Klasifikasi Antar Metode

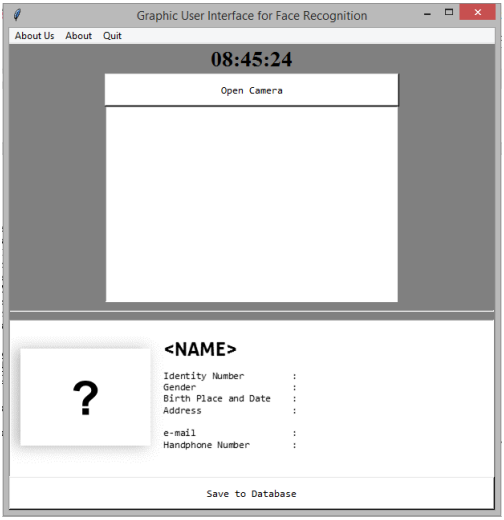
Perulangan	**MLP**				**CNN**			
	Accuracy	***Precision***	***Sensitivity***	***Fscore***	***Accuracy***	***Precision***	***Sensitivity***	***Fscore***
1	0,8750	0,9006	0,8750	0,8876	0,9688	0,9722	0,9688	0,9705
2	0,9063	0,9318	0,9063	0,9189	0,9688	0,9722	0,9688	0,9705
3	0,9063	0,9318	0,9063	0,9189	1,0000	1,0000	1,0000	1,0000
4	0,8750	0,9006	0,8750	0,8876	1,0000	1,0000	1,0000	1,0000
5	0,9688	0,9722	0,9688	0,9705	0,9688	0,9722	0,9688	0,9705
6	0,9688	0,9722	0,9688	0,9705	0,9688	0,9722	0,9688	0,9705
7	0,8750	0,9006	0,8750	0,8876	1,0000	1,0000	1,0000	1,0000
8	0,9063	0,9318	0,9063	0,9189	0,9688	0,9722	0,9688	0,9705
9	0,8750	0,9006	0,8750	0,8876	0,9375	0,9444	0,9375	0,9410
10	0,8750	0,9006	0,8750	0,8876	0,9688	0,9722	0,9688	0,9705
rata-rata	**0,9031**	**0,9243**	**0,9031**	**0,9136**	**0,9750**	**0,9778**	**0,9750**	**0,9764**

Metode yang dipilih merupakan metode yang mendapatkan kinerja klasifikasi terbaik. Hasil dari kinerja klasifikasi sudah lebih dari 90% sehingga kedua metode sangat baik untuk mengklasifikasikan gambar. Dari hasil rata-rata tersebut didapatkan pada metode CNN hasil rata-rata kinerja klasifiksi adalah *accuracy* sebesar 0,9750, *precision* sebesar 0,9778, *sensitivity* senilai 0,9750 dan *Fscore* sebesar 0,9764. Berbeda dengan MLP dengan hasil rata-rata kinerja klasifikasi lebih rendah daripada CNN. *Accuracy* yang didapatkan sebesar 0,9031,

precision senilai 0,9243, *sensitivity* 0,9301 dan *Fscore* sebesar 0,9136. Dari segi parameter yang dihasilkan juga jauh lebih tinggi MLP. Sehingga model dari metode yang akan digunakan pada *Graphic User Interface* adalah model *fold* ke-4 pada metode *Convolutional Neural Network* dengan *convolutional filter* berupa 9×9 dan *pooling filter* berukuran 8×8 serta *fully connected layer* berjumlah 10.

4.3 *Graphic User Interface*

Dalam pembuatan model, terdapat 3 tahap, yaitu *training*, *validation*, dan *testing*. Hasil *model* dari *training* dicobakan ke *validation* yang masih memiliki untuk mendapatkan kinerja klasifikasi terbaik dan *loss* yang dihasilkan. Selanjutnya model yang didapatkan tersebut digunakan untuk *testing* dengan data baru yang ingin diketahui targetnya. Pemodelan ini diaplikasikan pada *Graphic User Interface*. Tampilan awal dari GUI yang dibuat tercantum pada Gambar 4.6.



Gambar 4.6 Tampilan awal GUI

Pada menu utama GUI tersebut terdapat 2 panel, yaitu panel pertama untuk menampilkan citra dari hasil *capture* pada *webcam*

dan panel kedua untuk menampilkan hasil dari identifikasi wajah. Dalam GUI tersebut, juga terdapat 2 tombol yaitu '*Open Camera*' dan '*Save to Database*'. Di GUI tersebut juga disertakan menu yang terdiri dari '*About Us*' yang didalamnya tercantum '*Company*' dan '*Employee*', selanjutnya disertakan pula '*About*', dan '*Quit*'. Penjelasan masing-masing tombol dan menu tersebut disajikan dalam Tabel 4.9.

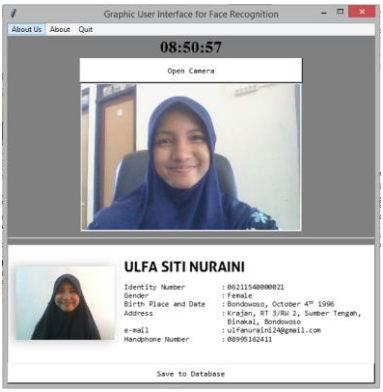
Tabel 4.9 Penjelasan Fungsi Tombol/Menu pada GUI

Nama Tombol/Menu	Fungsi
'*Open Camera*'	Memunculkan *webcam*
'*Save to Database*'	Menyimpan hasil identifikasi wajah pada database yang berekstensi .csv
'*Company*'	Memperkenalkan perusahaan yang menggunakan GUI ini
'*Employee*'	Memunculkan identitas *respon* yang digunakan pada *training*
'*About*'	Menampilkan informasi tentang GUI ini
'*Quit*'	Menutup Aplikasi

Langkah pertama yang dilakukan adalah menampilkan *webcam* dengan menggunakan tombol '*Open Camera*'. Gambar 4.7 merupakan tampilan ketika tombol '*Open Camera*' ditekan.

Gambar 4.7 Tampilan GUI Setelah Memunculkan *Webcam*

Setelah *webcam* muncul, maka selanjutnya menangkap gambar dari *camera webcam* dengan menggunakan tombol spasi pada *keyboard*, jika ingin menutup *webcam* maka menekan tombol ‘esc’ pada *keyboard*. Setelah tombol spasi pada *keyboard* ditekan, maka gambar yang tertangkap akan secara otomatis ditampilkan pada panel pertama. Selanjutnya, klasifikasi akan dijalankan oleh GUI menggunakan model terbaik yang telah didapat pada tahapan sebelumnya, yaitu metode *Convolutional Neural Network*. Proses tersebut secara otomatis dilakukan oleh GUI dan hasil dari klasifikasi akan tercantum pada panel 2 beserta identitas dari target. Gambar 4.8 menunjukkan tampilan *Graphic User Interface* setelah tombol spasi ditekan hingga klasifikasi selesai dilakukan.



Gambar 4.8 Tampilan GUI Setelah Tombol Spasi Ditekan

Hasil dari model pada tahapan sebelumnya menunjukkan bahwa kinerja klasifikasi tidak mencapai 100% atau ada kemungkinan terdapat *mis*-klasifikasi. *Mis*-klasifikasi terjadi disaat hasil dari tangkapan kamera mendapatkan identitas yang berbeda. Contoh hasil mis-klasifikasi terdapat pada Gambar 4.9.

Gambar 4.9 Mis-klasifikasi Hasil Identitas

Jika terjadi *mis*-klasifikasi maka digunakan kembali tombol '*Open Camera*' untuk mendapatkan citra baru. *Mis*-klasifikasi ini dapat dikarenakan oleh tidak sejajarnya pengguna dengan kamera *webcam*, posisi kamera yang tidak sesuai dengan posisi kamera saat pengambilan data *training*, pencahayaan yang sangat kontras, dan lain sebagainya. Oleh karena itu, terdapat tombol '*Save to Database*' untuk menyimpan hasil dari klasifikasi benar terhadap *database* berekstensi '.csv'. Tabel 4.10 merupakan tampilan dari *database* hasil klasifikasi.

Tabel 4.10 Tampilan *Database* Hasil Klasifikasi

Time	Identity_Number	Name
17/06/2019 7:43:52	6211540000021	Ulfa Siti Nuraini

Keseluruhan proses analisis telah selesai ketika hasil dari klasifikasi telah masuk dalam database. Jika ingin keluar dari GUI ini maka ditekan tulisan '*Quit*' pada menu, namun jika ingin melakukan identifikasi wajah kembali maka kembali menekan tombol '*Open Camera*'.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan analisis dan pembahasan yang telah dilakukan pada bab IV, maka diperoleh kesimpulan yaitu metode yang memiliki kinerja klasifikasi yang lebih baik yaitu *Convolutional Neural Network* dengan *convolutional filter* 9×9 dan *pooling filter* berukuran 8×8 dengan rata-rata kinerja klasifikasi yaitu *accuracy* sebesar 0,9750, *precision* sebesar 0,9778, *sensitivity* senilai 0,9750 dan *Fscore* sebesar 0,9764. Sedangkan untuk *Multilayer Perceptron* mendapatkan rata-rata kinerja klasifikasi yaitu *Accuracy* yang didapatkan sebesar 0,9031, *precision* senilai 0,9243, *sensitivity* 0,9301 dan *Fscore* sebesar 0,9136 dengan jumlah *neurons* pada *hidden layer* sebanyak 90. Jumlah parameter untuk model terbaik CNN yaitu sebanyak 9.172 dan untuk MLP yaitu 1.728.454.

5.2 Saran

Berdasarkan kesimpulan yang diperoleh, dapat dirumuskan saran sebagai pertimbangan penelitian selanjutnya adalah sebagai berikut.

1. Menambahkan tahap *preprocessing* seperti menandai wajah atau *cropping* wajah sehingga daerah yang digunakan untuk klasifikasi orang terpusat pada wajah dan tidak menimbulkan *noise* yang banyak.
2. Menggunakan jumlah *neurons* dan *layer* serta ukuran *filter* yang lain agar didapatkan hasil yang maksimal.

(Halaman ini sengaja dikosongkan)

DAFTAR PUSTAKA

Alpaydin, E. (2009). *Introduction to Machine Learning* (2nd ed.). London: MIT Press.

Arianto, Y. (2019). *Dikadali ASN Nakal, Pemkab Lumajang Tambahi Fitur Scan Wajah di SiPERLU*. Diakses pada tanggal 5 Maret 2019, dari Bangsa Online: https:// www.bangsaonline.com/berita/53062/dikadali-asn-nakal-pemkab-lumajang-tambahi-fitur-scan-wajah-di-siperlu

Berthold, M., & Hand, D. (2010). *Intelligent Data Analysis* (2nd ed.). Berlin: Springer Science.

Bishop, C. (2006). *Pattern Recognition and Machine Learning*. New York: Springer Science.

Dunstone, T., & Yager, N. (2009). *Biometric System and Data Analysis*. Berlin : Springer Science.

Faculty of Chemistry. (2019). *Image Science Fundamentals*. Diakses pada tanggal 3 Maret 2019, dari BRNO University of Technology website: https://www2.fch.vut.cz/lectures/ imagesci/

Ferreira, A., & Giraldi, G. (2017). Convolutional Neural Network Approaches to Granite Tiles Classification. *Expert System Application*, Vol 84, 1-11.

Fukushima, K. (1980). *Neocognitron : A self-organizing neural network model fora mechanism of pattern recognition unaffected by shift in position.* Tokyo: Biological Cybernetics.

Ganegedara, T. (2018, Mei 31). *Intuitive Guide to Convolution Neural Networks*. Diakses pada tanggal 29 Juni 2019, dari Towards Data Science: https://towardsdatascience.com /light-on-math-machine-learning-intuitive-guide-to-convolution-neural-networks-e3f054dd5daa

Gokgoz, E., & Subasi, A. (2015). Comparison of Decision Tree Algorithms for EMG Signal Classification Using DWT. *Biomedical Signal Processing and Control*, Vol 18, 138-144.

Gonzalez, R., Woods, R., & Eddins, S. (2009). *Digital Image Processing Using MATLAB* (2nd ed.). USA: Gatesmark Publishing.

GoodFellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning.* Cambridge: The MIT Press.

Iriawan, N., Pravitasari, A., Fithriasari, K., Irhamah, Purnami, S., & Ferriastuti, W. (2018). Comparative study of Brain Tumor Segmentation using Different Segmentation Techniques in Handling Noise. *2018 International Conference on Computer Engineering, Network and Intelligent Multimedia (CENIM)*, (pp. 289-293). Surabaya.

Karpathy, A. (2018). *CS231n Convolutional Neural Netwoerk for Visual Recognition.* Diakses pada tanggal 3 Maret 2019, dari Standford University CS Class: http://cs231m.github.io/

Khaulasari, H. (2016). *Combine Sampling - Least Square Support Vector Machine Untuk Klasifikasi Multi Class Imbalanced Data* (*Masters Thesis*). Surabaya: Fakultas Matematika dan Ilmu Pengetahuan Alam: Institut Teknologi Sepuluh Nopember.

Kingma, D., & Ba, J. (2014). Adam : A Method for Stochastic Optimization. *International Conference on Learning Representations 2015*. San Diego.

Medina, E., Petraglia, M., Gomes, J., & Petraglia, A. (2017). Comparison of CNN and MLP classifiers for algae detection in underwater pipelines. *2017 Seventh International Conference on Image Processing Theory, Tools and Applications (IPTA)* (pp. 1-6). Montreal, QC, Canada: IEEE.

Mounica, C. (2016). Face Detection and Recognition using LBPH. *International Journal of Engineering Research and Science & Technology*, Vol 3, 10-16.

Munir, R. (2004). *Pengolahan Citra Digital.* Bandung: Informatika.

Nwankpa, C., Ijomah, W., Gachagan, A., & Marshall, S. (2018). *Activation Functions: Comparison of trends in Practice and Research for Deep Learning.* Diakses pada tanggal 3 Maret 2019, dari arXiv *website*: https://arxiv.org/abs/1811.03378

Rahmat, Y. (2019). *SiPerlu Diterapkan untuk Tenaga Non PNS Pemkab Lumajang*. Diakses pada tanggal 3 Maret 2019, dari Portal Berita Info Publik: http://www.infopublik.id/kategori/nusantara/333933/siperlu-diterapkan-untuk-tenaga-non-pns-pemkab-lumajang.

Russell, R., & Marksll, R. (1999). *Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks.* Cambridge: The MIT Press.

Sarle, W. (1994). Neural Networks and Statistical Models. *The Nineteenth Annual SAS Users Group International Conference.* NC, USA: SAS Institute Inc.

Setiawan, B., & Rudiyanto. (2004). Aplikasi Neural Networks untuk Prediksi Aliran Sungai. *Prosiding Semiloka Teknologi Simulasi dan Komputasi serta Aplikasi 2004*. Jakarta: BPPT.

Singh, G., & Sachan, M. (2014). Multi-layer perceptron (MLP) neural network technique for offline handwritten Gurmukhi character recognition. *IEEE International Conference* (pp. 1-5). Coimbatore, India: IEEE.

Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information Processing and Management*, 45(2) , 427-437.

Tharwat, A. (2018). Classification Assessment Method. *Applied Computing and Informatics*. doi:doi.org/10.1016/j.aci.2018.08.003.

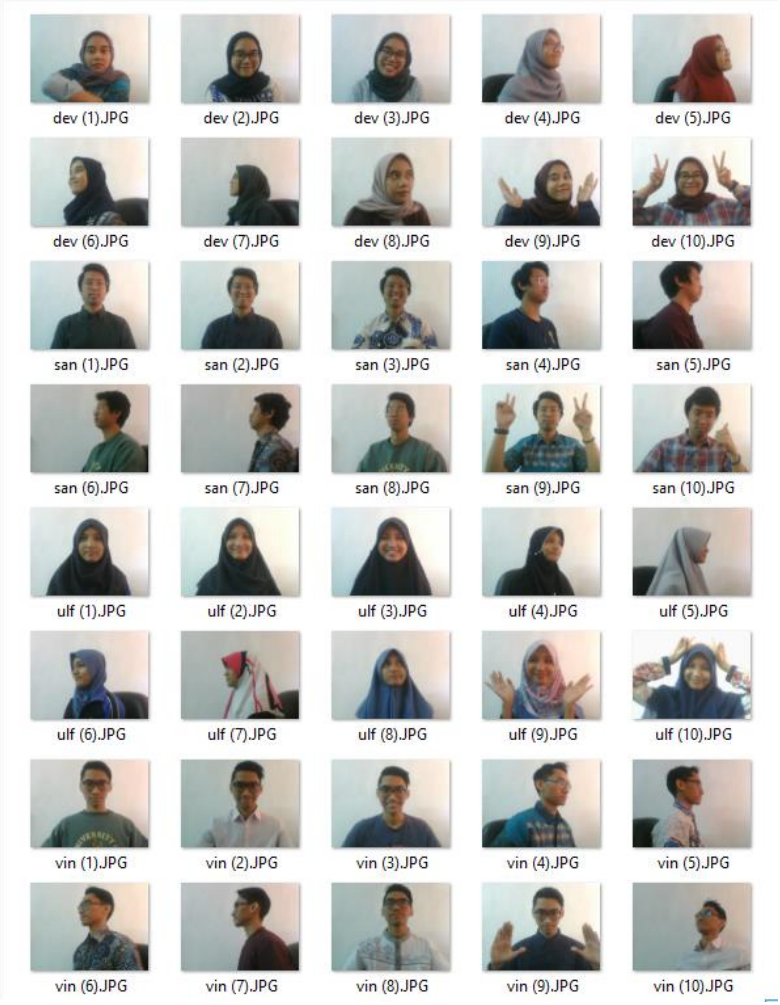
Tivive, F., & Bouzerdoum, A. (2006). A Gender Recognition System Using Shunting Inhibitory Convolutional Neural Networks. *International Joint Conference on Neural Networks* (pp. 5336-5341). New York, USA: IEEE.

Wang, J., & Perez, L. (2017). The Effectiveness of Data Augmentation in Image Classification using Deep Learning. *Computer Vision and Pattern Recognition*. Diakses pada tanggal 3 Maret 2019, dari arXiv *website*: https://arxiv.org/abs/1712.04621.

Woods, R., & Gonzales, R. (2008). *Digital Image Processing* (3rd ed.). New Jersey: Pearson Education.

LAMPIRAN

Lampiran 1. Citra yang Digunakan



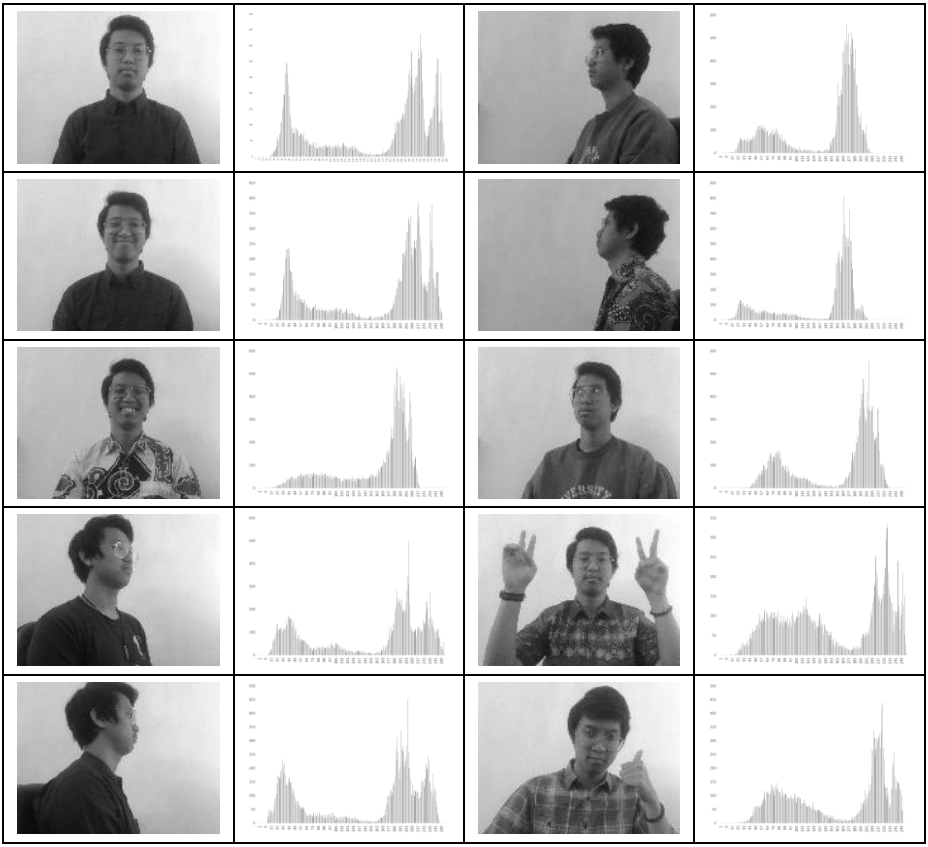
Lampiran 2. Data Hasil *Preprocessing*

No	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}	…	X_{19200}	Y
1	202	203	204	204	206	206	207	208	204	205	…	172	Y_1
2	100	101	101	102	102	103	103	104	103	103	…	85	Y_1
…	…	…	…	…	…	…	…	…	…	…	…	…	…
80	0	3	2	0	0	11	0	0	6	0	…	3	Y_1
81	224	224	224	223	223	222	222	222	220	220	…	175	Y_2
82	111	111	111	111	111	111	111	111	110	110	…	85	Y_2
…	…	…	…	…	…	…	…	…	…	…	…	…	…
160	1	0	12	0	3	5	9	0	3	0	…	4	Y_2
161	232	233	233	233	233	232	231	230	236	236	…	185	Y_3
162	116	116	116	116	116	116	116	116	118	118	…	92	Y_3
…	…	…	…	…	…	…	…	…	…	…	…	…	…
240	0	0	0	0	0	13	0	0	0	6	…	2	Y_3
241	220	218	216	213	212	211	211	212	214	215	…	68	Y_4
242	116	114	111	108	105	104	104	104	107	107	…	35	Y_4
…	…	…	…	…	…	…	…	…	…	…	…	…	…
320	0	5	5	3	0	11	0	0	5	0	…	5	Y_4

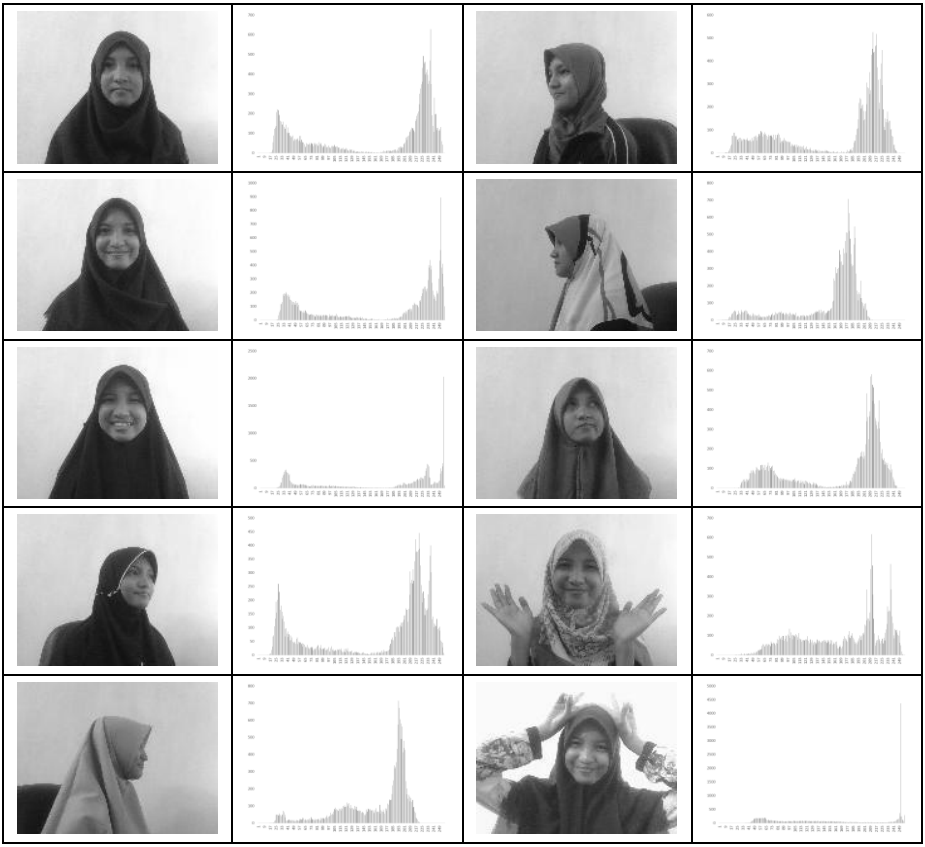
Lampiran 3. *Histogram* setiap citra

Untuk Y_1

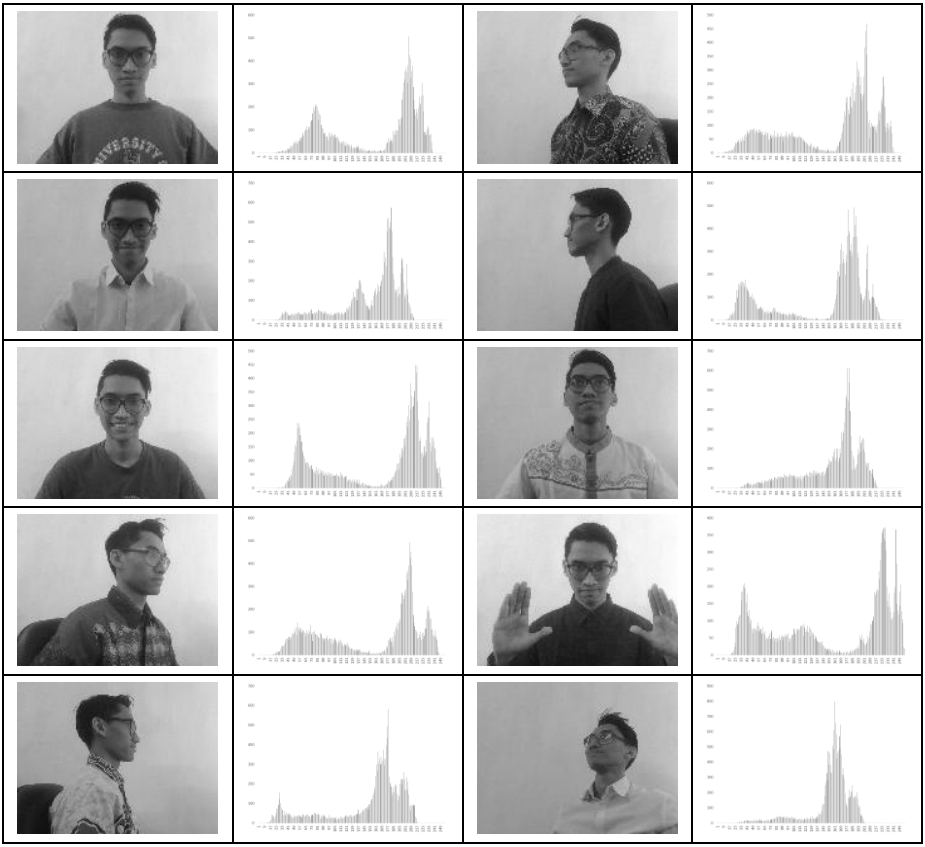
Untuk Y_2



Untuk Y_3



Untuk Y_4



Lampiran 4. *Syntax* untuk Import *Packages*

```
from numpy.random import seed
seed(0)
from random import seed
seed(0)
from PIL import Image, ImageEnhance
from PIL import Image
import os
import cv2
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib
from numpy import *
from sklearn.utils import shuffle
from sklearn.model_selection import train_test_split
from sklearn.metrics import
classification_report,confusion_matrix
from sklearn.model_selection import StratifiedKFold
import theano
from keras import backend as K
from keras.models import Sequential
from keras.layers.core import Dense, Dropout, Activation, Flatten
from keras.layers.convolutional import Convolution2D,
MaxPooling2D
from keras.layers.convolutional import Conv2D, MaxPooling2D
from keras.optimizers import SGD,RMSprop,adam
from keras.utils import np_utils
from keras.initializers import RandomUniform, glorot_uniform,
glorot_normal
from keras.callbacks import EarlyStopping, ModelCheckpoint,
Callback
class EarlyStoppingByLossVal(Callback):
    def __init__(self, monitor='val_loss', value=0.01, verbose=0):
        super(Callback, self).__init__()
```

```
        self.monitor = monitor
        self.value = value
        self.verbose=verbose
    def on_epoch_end(self, epoch,logs={}):
        current=logs.get(self.monitor)
        if current is None:
            warnings.warn("Early stopping requires %$ available!" %
self.monitor, RuntimeWarning)
        if current < self.value:
            if self.verbose>0:
                print("Epoch %05d: early stopping THR" % epoch)
            self.model.stop_training=True
callbacks = [
            EarlyStoppingByLossVal(monitor='loss', value=0.05,
verbose=0),
            ModelCheckpoint('best_model.h5', monitor='loss',
save_best_only=True, verbose=0)]
```

Lampiran 5. *Syntax preprocessing* data dan augmentasi

```
pathgr = 'E:/Tugas Akhir/Tugas Akhir_/data_traininggr2'
path2 = 'E:/Tugas Akhir/Tugas Akhir_/New2'
pathhor= 'E:/Tugas Akhir/Tugas Akhir_/entah2'
factor=0.25
img_rows, img_cols = 160, 120

listing = os.listdir(path1)
for file in listing:
    im = Image.open(path1 + '/' + file)
    img= im.convert('L')
    width = img.size[0]
    height = img.size[1]
    newW=int(width*factor)
    newH=int(height*factor)
    imgnew=Image.new("L", (newW, newH), "white")
    p=img.load()
```

```
    q=imgnew.load()
    for col in range(newW):
       for row in range(newH):
          p=img.load()
          z=p[col/factor,row/factor]
          q=imgnew.load()
          q[col,row]=z
    imgnew.save(pathgr +'/' +  file)

listing = os.listdir(pathgr)
# enhance
for file in listing:
    im = Image.open(pathgr + '/' + file)
    im.save(path2 +'/' +  file)
    contrastinc = ImageEnhance.Contrast(im)
    contrastinc = contrastinc.enhance(1.5)
    contrastinc.save(path2 +'/' +  file + "_contrastplus.JPG")
    contrastdes= ImageEnhance.Contrast(im)
    contrastdes = contrastdes.enhance(0.5)
    contrastdes.save(path2 +'/' +  file + "_contrastmin.JPG")
    enhancer = ImageEnhance.Brightness(im)
    enhanced_im = enhancer.enhance(1.5)
    enhanced_im.save(path2 +'/' +  file + "_brigthnessplus.JPG")
    enhancer = ImageEnhance.Brightness(im)
    enhanced_im = enhancer.enhance(0.5)
    enhanced_im.save(path2 +'/' +  file + "_brigthnessmin.JPG")

# flip
listing = os.listdir(path1)
for file in listing:
    im = Image.open(path1 + '/' + file)
    imhor = np.fliplr(im) # horizontal
    plt.imsave(pathhor +'/' +  file, imhor)
    im = Image.open(pathhor + '/' + file)
    img= im.convert('L')
```

```
    width = img.size[0]
    height = img.size[1]
    newW=int(width*factor)
    newH=int(height*factor)
    imgnew=Image.new("L", (newW, newH), "white")
    p=img.load()
    q=imgnew.load()
    for col in range(newW):
        for row in range(newH):
            p=img.load()
            z=p[col/factor,row/factor]
            q=imgnew.load()
            q[col,row]=z
    imgnew.save(path2 +'/' +  file + "_hor.JPG")
#rotate
for file in listing:
    colorImage = Image.open(pathgr + '/' + file)
    rotated = colorImage.rotate(5)
    rotated.save(path2 +'/' +  file +"_rotatedright.JPG")
for file in listing:
    colorImage = Image.open(pathgr + '/' + file)
    rotated = colorImage.rotate(-5)
    rotated.save(path2 +'/' +  file +"_rotatedleft.JPG")
```

Lampiran 6. *Syntax* Membuat Data Frame ke Tipe *File .csv*

```
# number of channels
img_channels = 1

imlist = os.listdir(path2)
im1 = array(Image.open(path2 + '/'+ imlist[0]))
m,n = im1.shape[0:2]
imnbr = len(imlist) # get the number of images
immatrix = array([array(Image.open(path2 + '/' + im2)).flatten()
            for im2 in imlist],'f')
```

```
num_samples=size(os.listdir(path2) )
label=np.ones((num_samples,),dtype = int)
label[0:80]=0
label[81:160]=1
label[160:240]=2
label[240:]=3

data,Label = (immatrix,label)
data,Label = shuffle(immatrix,label, random_state=4)
train_data = [data,Label]

# number of output classes
kategori = 4
# number of epochs to train
epochmax = 300
batch_size=288
seed = 0
np.random.seed(seed)
(X, y) = (train_data[0],train_data[1])
dimData = np.prod(X.shape[1:])
X = X.reshape(X.shape[0], dimData)
X = X.astype('float64')
X /= 255
print(X.shape)
kfold = StratifiedKFold(n_splits=10, shuffle=True,
random_state=seed)

pd.DataFrame([label,imlist]).transpose().to_csv('foto.csv')
data=pd.DataFrame(train_data[0])
data['y']=train_data[1]
data.transpose().to_csv('data.csv')
```

Lampiran 7. *Syntax* klasifikasi dengan metode MLP

```
for train, test in kfold.split(X, y):
    (X[train],X[test])
    (y[train],y[test])
    Yb1_train = np_utils.to_categorical(y[train], kategori)
    Yb1_test = np_utils.to_categorical(y[test], kategori)
    #MLP
    mlp =()
    mlp = Sequential()
    mlp.add(Dense(10,input_dim=19200, activation = 'relu')))
    mlp.add(Dense(kategori, activation = 'softmax',
bias_initializer='zeros',
kernel_initializer=RandomUniform(minval=-0.05, maxval=0.05,
seed=None)))
    mlp.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])
    mlp.fit(X[train], Yb1_train, batch_size=batch_size,
epochs=epochmax,verbose=0, callbacks=callbacks,
validation_data=(X[test], Yb1_test))
    Y1_pred = mlp.predict(X[train])
    y1_pred = np.argmax(Y1_pred, axis=1)
    print(confusion_matrix(np.argmax(Yb1_train,axis=1),
y1_pred))
    #print(classification_report(np.argmax(Yb1_train,axis=1),
y1_pred))
    Y1_pred = mlp.predict(X[test])
    y1_pred = np.argmax(Y1_pred, axis=1)
    print(confusion_matrix(np.argmax(Yb1_test,axis=1), y1_pred))
    #print(classification_report(np.argmax(Yb1_test,axis=1),
y1_pred))
```

Lampiran 8. *Syntax* klasifikasi dengan metode CNN

```
seed = 0
np.random.seed(seed)
kfold = StratifiedKFold(n_splits=10, shuffle=True,
random_state=seed)
(X, y) = (train_data[0],train_data[1])
X = X.reshape(X.shape[0], 1, img_cols, img_rows)
X = X.astype('float32')
X /= 255

conv1 = 7
pool1 = 2
batch_size = 288
kategori = 4
epochmax = 300
filters1 = 8

cvscores = []
for train, test in kfold.split(X, y):
   cnn=()
   cnn = Sequential()
   cnn.add(Convolution2D(filters1, (conv1, conv1),
                  padding='valid',
                  input_shape=(1,img_cols,img_rows),
                  data_format='channels_first',
                  activation='relu'))
   cnn.add(MaxPooling2D(pool_size=(pool1, pool1)))
   cnn.add(Flatten())
   cnn.add(Dense(10, activation = 'relu'))
   cnn.add(Dense(kategori, activation = 'softmax',
bias_initializer='zeros',
kernel_initializer=RandomUniform(minval=-0.05, maxval=0.05,
seed=123)))
   cnn.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])
```

```
    hist = cnn.fit(X[train], np_utils.to_categorical(y[train],
kategori), batch_size=batch_size, epochs=epochmax,
              verbose=0, validation_data=(X[test],
np_utils.to_categorical(y[test], kategori)),
                    callbacks=callbacks, shuffle=False)
    # evaluate the cnn
    scores = cnn.evaluate(X[train], np_utils.to_categorical(y[train],
kategori), verbose=0)
    #print("%s: %.2f%%" % (cnn.metrics_names[1],
scores[1]*100))
    scores = cnn.evaluate(X[test], np_utils.to_categorical(y[test],
kategori), verbose=0)
    #print("%s: %.2f%%" % (cnn.metrics_names[1],
scores[1]*100))
    cvscores.append(scores[1] * 100)

    Y_pred = cnn.predict(X[train])
    y_pred = np.argmax(Y_pred, axis=1)
    p=cnn.predict_proba(X[train]) # to predict probability
    target_names = ['class 0', 'class 1', 'class 2', 'class 3']

#print(classification_report(np.argmax(np_utils.to_categorical(y[t
rain], kategori),axis=1), y_pred,target_names=target_names))
print(confusion_matrix(np.argmax(np_utils.to_categorical(y[train
], kategori),axis=1), y_pred))

    Y_pred = cnn.predict(X[test])
    y_pred = np.argmax(Y_pred, axis=1)
    p=cnn.predict_proba(X[test]) # to predict probability
    target_names = ['class 0', 'class 1', 'class 2', 'class 3']

#print(classification_report(np.argmax(np_utils.to_categorical(y[t
est], kategori),axis=1), y_pred,target_names=target_names))
```

```
print(confusion_matrix(np.argmax(np_utils.to_categorical(y[test],
kategori),axis=1), y_pred))

print("%.2f%% (+/- %.2f%%)" % (np.mean(cvscores),
np.std(cvscores)))
```

Lampiran 9. Surat Pernyataan Data

SURAT PERNYATAAN

Saya yang bertanda tangan di bawah ini, mahasiswa Departemen Statistika FMKSD ITS:

Nama : Ulfa Siti Nuraini
NRP : 06211540000021

menyatakan bahwa data yang digunakan dalam Tugas Akhir ini benar-benar merupakan data primer yang dikumpulkan dengan cara mengambil gambar menggunakan kamera dengan objek mahasiswa Statistika ITS angkatan 2015.

Surat pernyataan ini dibuat dengan sebenarnya. Apabila terdapat pemalsuan data, maka saya siap menerima sanksi sesuai aturan yang berlaku.

Mengetahui
Pembimbing Tugas Akhir

(Dr. Dra. Kartika Fithriasari, M.Si)
NIP. 19691212 199303 2 002

Surabaya, 28 Juni 2019

(Ulfa Siti Nuraini)
NRP. 06211540000021

BIODATA PENULIS

Penulis dilahirkan di Bondowoso, 4 Oktober 1996 dengan identitas Ulfa Siti Nuraini, biasa dipanggil Ulfa. Penulis merupakan putri kedua dari Bapak Ali dan Ibu Tatuk Eko Sihwinantu yang beralamat di Krajan, RT 3 RW 2 Desa Sumber Tengah, Kecamatan Binakal, Kabupaten Bondowoso. Penulis menempuh pendidikan formal di SDN Kotakulon 1 Bondowoso, SMPN 1 Bondowoso, dan SMAN 1 Jember. Kemudian penulis diterima sebagai mahasiswa Departemen Statistika ITS pada tahun 2015. Selama masa perkuliahan, penulis aktif di Divisi Statistics Computer Course (SCC) Himpunan Mahasiswa Statistika ITS (HIMASTA-ITS) sebagai staff Training Development 2016-2017 dan manajer Training Development pada periode 2017-2018. Bagi pembaca yang ingin berdiskusi, memberikan saran, dan kritik mengenai Tugas Akhir ini dapat disampaikan melalui email ulfanuraini24@gmail.com atau melalui nomor 08995162411.