TUGAS AKHIR – IF184802

PENGENALAN WAJAH PADA APLIKASI SISTEM KEHADIRAN MAHASISWA (SIKEMAS) DENGAN METODE *CONVOLUTIONAL NEURAL NETWORK*

ANISAH PUTRI DIANA
NRP 05111540000135

Dosen Pembimbing
Dr. Eng. Chastine Fatichah, S.Kom., M.Kom.
Dwi Sunaryono, S.Kom., M.Kom.

DEPARTEMEN INFORMATIKA
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2019

[Halaman ini sengaja dikosongkan]

TUGAS AKHIR – IF184802

PENGENALAN WAJAH PADA APLIKASI SISTEM KEHADIRAN MAHASISWA (SIKEMAS) DENGAN METODE *CONVOLUTIONAL NEURAL NETWORK*

ANISAH PUTRI DIANA
NRP 05111540000135

Dosen Pembimbing
Dr. Eng. Chastine Fatichah, S.Kom., M.Kom.
Dwi Sunaryono, S.Kom., M.Kom.

DEPARTEMEN INFORMATIKA
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2019

[Halaman ini sengaja dikosongkan]

FINAL PROJECT – IF184802

FACE RECOGNITION FOR STUDENT ATTENDANCE SYSTEM (SIKEMAS) WITH CONVOLUTIONAL NEURAL NETWORK METHOD

ANISAH PUTRI DIANA
NRP 05111540000135

Supervisors
Dr. Eng. Chastine Fatichah, S.Kom., M.Kom.
Dwi Sunaryono, S.Kom., M.Kom.

INFORMATICS DEPARTMENT
Faculty of Information and Communication Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2019

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

PENGENALAN WAJAH PADA APLIKASI SISTEM KEHADIRAN MAHASISWA (SIKEMAS) DENGAN METODE *CONVOLUTIONAL NEURAL NETWORK*

TUGAS AKHIR

Diajukan Guna Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Komputasi Cerdas dan Visi
Program Studi S-1 Departemen Informatika
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember

Oleh :
ANISAH PUTRI DIANA
NRP. 05111540000135

Disetujui oleh Pembimbing Tugas Akhir

1. Dr.Eng. Chastine Fatichah, S.Kom., M.Kom. .....
   NIP. 197512202001122002 (Pembimbing 1)

2. Dwi Sunaryono, S.Kom., M.Kom. .....
   NIP. 197205281997021001 (Pembimbing 2)

SURABAYA

JANUARI 2019

[Halaman ini sengaja dikosongkan]

PENGENALAN WAJAH PADA APLIKASI SISTEM KEHADIRAN MAHASISWA (SIKEMAS) DENGAN METODE *CONVOLUTIONAL NEURAL NETWORK*

Nama Mahasiswa	: Anisah Putri Diana
NRP	: 05111540000135
Departemen	: Informatika FTIK-ITS
Dosen Pembimbing 1	: Dr. Eng. Chastine Fatichah, S.Kom., M.Kom.
Dosen Pembimbing 2	: Dwi Sunaryono, S.Kom., M.Kom.

ABSTRAK

Teknologi machine learning saat ini sudah banyak diimplementasikan di berbagai aplikasi seperti aplikasi desktop, web, maupun mobile. Salah satu aplikasi yang mengimplementasikan machine learning adalah sistem kehadiran mahasiswa (SIKEMAS) yaitu menggunakan pengenalan wajah. Pada penelitian sebelumnya, machine learning yang digunakan di aplikasi SIKEMAS menggunakan klasifikasi SVM dan LDA. Terdapat beberapa kelemahan pada penelitian sebelumnya seperti tingkat akurasi yang rendah dan waktu komputasi yang lama. Evaluasi pada metode SVM pada dataset SIKEMAS menghasilkan akurasi sebesar 88,5% dengan waktu prediksi model 15,32 detik sementara ukuran ruang model sebesar 73,2 MB. Sementara pada LDA menghasilkan akurasi 93,4% dengan waktu prediksi 0,50 detik *dan ukuran ruang model sebesar 70,1 MB. Sehingga diusulkan sebuah metode untuk memperbaiki masalah dalam klasifikasi dan pengenalan wajah pada SIKEMAS menggunakan metode deep learning Convolutional Neural Network (CNN). Metode ini digunakan karena kemampuan CNN yang terbukti pada beberapa penelitian yang ada, baik dalam mengenali citra atau gambar, mengklasterkan berdasarkan kemiripannya, dan menampilkan pengenalan objek dengan layer. Hasil evaluasi pada dataset wajah sistem kehadiran mahasiswa (SIKEMAS) dengan uji coba parameter optimizer rmsprop menghasilkan nilai akurasi*

model terbaik sebesar 97,8% dengan loss sebesar 0,101, waktu prediksi 0.73 detik dan ukuran ruang model sebesar 1,8 MB.

Kata kunci: SIKEMAS, pengenalan wajah, klasifikasi, convolutional neural networks, deep learning

FACE RECOGNITION FOR STUDENT ATTENDANCE SYSTEM (SIKEMAS) WITH CONVOLUTIONAL NEURAL NETWORK METHOD

Student Name	: Anisah Putri Diana
NRP	: 05111540000135
Departement	: Informatika FTIK-ITS
Supervisor 1	: Dr. Eng. Chastine Fatichah, S.Kom., M.Kom.
Supervisor 2	: Dwi Sunaryono, S.Kom., M.Kom.

ABSTRACT

Machine learning technology is now widely implemented in various applications such as desktop, web, and mobile applications. One application that implements machine learning is the student attendance system (SIKEMAS), which uses face recognition. In previous studies, machine learning used in the SIKEMAS application used the SVM and LDA classifications. There are several weaknesses in previous studies such as low accuracy and long computing time. The evaluation on the SVM method in the SIKEMAS dataset resulted in an accuracy of 88.5% with a model prediction time of 15.32 seconds while the model size was 73.2MB. While the LDA produces an accuracy of 93.4% with a prediction time of 0.50 seconds and the model size is 70.1MB. So that a method was proposed to fix problems in face classification and recognition in SIKEMAS using the deep learning method Convolutional Neural Network (CNN). This method is used because of the ability of CNN which is proven in several existing studies, both in recognizing images, clustering based on their similarity, and displaying object recognition with layers. The evaluation results on the student attendance system (SIKEMAS) dataset with the rmsprop optimizer parameter test resulted in the best model accuracy value of 97.8% with a loss of 0.101, prediction time of 0.73 seconds and model space size of 1.8 MB.

Keywords: SIKEMAS, face recognition, classification, convolutional neural networks, deep learning

[Halaman ini sengaja dikosongkan]

KATA PENGANTAR

Puji syukur saya ucapkan kepada Allah swt. Yang Maha Kuasa. Karena atas karunia serta rahmat-Nya saya dapat menyelesaikan tugas akhir yang berjudul :

PENGENALAN WAJAH PADA APLIKASI SISTEM KEHADIRAN MAHASISWA (SIKEMAS) DENGAN METODE *CONVOLUTIONAL NEURAL NETWORK*

Bagi penulis, penyusunan dan pengerjaan tugas akhir ini merupakan sebuah pengalaman yang berharga. Selama pengerjaan tugas akhir, penulis dapat belajar dan mengeksplorasi lebih dalam materi yang telah didapatkan selama penulis menjalani perkuliahan di Departemen Informatika ITS. Tugas akhir ini merupakan implementasi dari apa yang sudah penulis pelajari. Melalui lembar ini, penulis ingin menyampaikan ucapan terima kasih dan penghormatan sebesar-besarnya kepada :

1. Allah swt.
2. Ayah, Mama, Bang Hanif, Irfan, Ammar, Nadya dan keluarga penulis yang telah dan selalu memberikan dukungan moral dan material serta doa yang tulus kepada penulis. Serta selalu memberikan semangat dan motivasi kepada penulis dalam mengerjakan tugas akhir ini.
3. Ibu Dr. Eng. Chastine Fatichah, S.Kom., M.Kom. selaku dosen pembimbing I yang telah membimbing dan membantu penulis serta selalu memberikan motivasi dan masukan dalam penyelesaian tugas akhir ini dengan sabar.
4. Bapak Dwi Sunaryono, S.Kom., M.Kom. selaku dosen pembimbing II yang telah membimbing dan membantu penulis serta selalu memberikan motivasi dan masukan dalam penyelesaian tugas akhir ini dengan sabar.

5. Bapak dan Ibu civitas akademika Departemen Informatika ITS yang telah membagikan ilmu serta pengalamannya kepada penulis selama berkuliah di Informatika ITS.
6. Nahda, Purina, dan Hania yang sudah mau menjadi teman susah-senang penulis selama berkuliah di Informatika ITS.
7. Nisa Aulia dan Dwi Hidayatti teman yang selalu memberikan dukungan kepada penulis selama masa sekolah sampai sekarang.
8. Teman-teman admin lab MIS Informatika ITS. Purina, Rafi, Fajar, Ariya, Muhajir, Fino, Zevi, Fasma, Ayas, Andre, Naja, Shania, dan Yuda yang selalu memberikan semangat dan dukungan kepada penulis selama berada di lab.
9. Rezky Alamsyah yang telah banyak memberikan ilmu kepada penulis dalam mengerjakan tugas akhir ini dan Rakhma Rufaida selaku teman seperjuangan kerja praktik di GMF.
10. Sirria dan Hendry teman seperjuangan bimbingan tugas akhir Bu Chastine.
11. Teman-teman TC angkatan 2015.
12. Teman-teman BPH Schematics 2016-2017, teman-teman Danus Schematics 2016-2017, teman-teman Medfo 5th gen, teman-teman External Affairs BEM FTIf 2016 – 2017, teman-teman saman tc, teman-teman lab KCV.
13. Serta pihak-pihak lain yang namanya tidak dapat penulis sebutkan satu per satu.

Penulis menyadari banyaknya kekurangan dalam tugas akhir ini, sehingga penulis sangat terbuka akan kritik dan saran yang membangun dari pembaca untuk perbaikan ke depannya.

Surabaya, Januari 2019

Anisah Putri Diana

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

[*Halaman ini sengaja dikosongkan*]

DAFTAR KODE SUMBER

[*Halaman ini sengaja dikosongkan*]

BAB I
PENDAHULUAN

1.1 Latar Belakang

Perkembangan *machine learning* saat ini sangat pesat dan hampir semua aplikasi berbasis teknologi mulai mengimplementasikan bantuan *machine learning* dalam penerapannya. Kemampuan *machine learning* dalam belajar dan memprediksi pola yang diberikan tentunya akan sangat membantu kehidupan manusia dalam penyelesaian masalah yang ada. *Machine learning* bekerja dengan memahami struktur data yang diberikan dan mencocokan data tersebut ke dalam sebuah model sehingga dapat dimanfaatkan oleh manusia. *Machine learning* dapat ditemui di banyak aplikasi seperti rekomendasi pada situs amazon, netflix dan beberapa media sosial. Perbankan menggunakan *machine learning* dalam mendeteksi *fraud* pada transaksi kartu kredit pun dalam dunia medis *machine learning* digunakan untuk mendiagnosa penyakit.

Pengaplikasian *machine learning* juga sudah ditanamkan pada aplikasi Sistem Kehadiran Mahasiswa (SIKEMAS) Informatika ITS. Aplikasi SIKEMAS menggunakan metode pengenalan wajah dalam proses presensi kehadiran mahasiswa. Metode pengenalan wajah pada aplikasi SIKEMAS sebelumnya menggunakan metode klasifikasi *Support Vector Machine (SVM)* dan *Linear Discriminant Analysis (LDA)* [1]. Pada praktiknya, metode klasifikasi SVM dan LDA ini masih mengalami beberapa kendala dalam pengenalan wajah mahasiswa yang akan melakukan presensi antara lain: kecilnya akurasi yang dihasilkan dalam pengenalan wajah mahasiswa dan ukuran model yang besar sehingga saat melakukan presensi membutuhkan waktu yang cukup lama untuk mencocokan wajah mahasiswa dengan model yang sudah ada. Hasil pada SVM menggunakan dataset SIKEMAS memiliki akurasi sebesar 88,5% dengan waktu prediksi model

selama 15,3194 *seconds* dan ukuran ruang (*space*) model sebesar 73,2 MB. Sementara pada LDA tingkat akurasi yang dihasilkan sebesar 93,4% dan model membutuhkan waktu prediksi selama 0,500933 *seconds* dengan ukuran ruang (*space*) model sebesar 70,1 MB. Untuk menangani hal ini, penulis mengajukan usulan untuk memperbaiki metode pengenalan wajah yang dipakai dalam aplikasi SIKEMAS dengan menggunakan metode *Convolutional Neural Network* (CNN). Tingkat keberhasilan CNN

Convolutional Neural Network (CNN) adalah *deep artificial neural network* yang sering digunakan untuk klasifikasi citra atau gambar, mengklasterkan berdasarkan kemiripannya, dan menampilkan pengenalan objek dengan layar. Dalam pengenalan gambar CNN dapat mengidentifikasi wajah, individu, tanda-tanda yang ada di jalan, tumor, dan berbagai data visual lainnya. Secara garis besar, CNN tidak berbeda dengan *Neural Network* lainnya. CNN terdiri dari *neuron* yang memiliki *weight*, *bias* dan *activation function*. Hanya saja, yang membedakan CNN dengan *Neural Network* biasanya adalah arsitektur CNN dibagi menjadi dua bagian besar yaitu *Feature Extraction Layer* dan *Fully Connected Layer*.

1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam tugas akhir ini adalah sebagai berikut:

1. Bagaimana hasil komputasi metode Convolutional Neural Network pada pengenalan wajah di aplikasi Sistem Kehadiran Mahasiswa?
2. Bagaimana evaluasi pengenalan wajah dengan menggunakan metode Convolutional Neural Network pada aplikasi Sistem Kehadiran Mahasiswa?

1.3 Batasan Masalah

Permasalahan yang dibahas pada Tugas Akhir ini memiliki beberapa batasan, yaitu sebagai berikut :

Dataset yang digunakan adalah data gambar wajah mahasiswa aktif Departemen Informatika.

Implementasi program menggunakan bahasa pemrograman Python.

Dataset terdiri dari 3 jenis untuk setiap uji coba skenario yang dilakukan.s

1.4 Tujuan

Tujuan dari pembuatan Tugas Akhir ini adalah untuk mengaplikasikan metode *Convolutional Neural Network* dalam pengenalan wajah pada aplikasi Sistem Kehadiran Mahasiswa (SIKEMAS).

1.5 Manfaat

Dengan adanya usulan Tugas Akhir ini, diharapkan dapat menemukan metode pengenalan wajah yang lebih baik pada aplikasi SIKEMAS sehingga proses presensi mahasiswa bisa lebih efisien.

1.6 Metodologi

Langkah-langkah yang harus ditempuh dalam pengerjaan tugas akhir ini adalah sebagai berikut:

1. **Penyusunan proposal tugas akhir**

   Proposal tugas akhir ini berisi tentang penjelasan mengenai pendahuluan dari tugas akhir yang dibuat. Pendahuluan ini terdiri dari hal-hal yang melatarbelakangi tugas akhir, rumusan masalah yang diangkat, batasan masalah yang ada, tujuan, dan manfaat dari tugas akhir ini. Selain itu dijabarkan pula tinjauan pustaka yang digunakan sebagai referensi dalam pengerjaan tugas akhir ini.

2. **Studi Literatur**

   Pada studi literatur tugas akhir ini telah dipelajari sejumlah referensi yang relevan terhadap tugas akhir yang telah dikerjakan. Studi literatur ini didapatkan dari buku, publikasi ilmiah, artikel serupa, internet serta dokumentasi-dokumentasi metode yang digunakan.

Studi literatur tentang CNN berpusat pada publikasi-publikasi ilmiah serta dokumentasi-dokumentasi penerapan arsitektur CNN untuk kasus pengenalan wajah. Dikarenakan sifat dari CNN yang cenderung eksperimental dalam menentukan desain yang tepat untuk arsitekturnya, maka arsitektur CNN pada implementasi tugas akhir ini akan mulai dibangun dengan cara mengombinasikan beberapa referensi.

3. Desain metode

Desain metode dalam tugas akhir ini terbagi menjadi dua garis besar, yaitu perancangan sistem pengenalan wajah dengan aristektur CNN dan penerapan sistem pengenalan wajah di aplikasi Sistem Kehadiran Mahasiswa (SIKEMAS).

Pada perancangan sistem pengenalan wajah terdapat beberapa skenario pengembangan dan uji coba. Perancangan metode CNN yang digunakan untuk mendapatkan model yang baik dan optimal untuk diimplementasikan ke dalam Sistem Kehadiran Mahasiswa (SIKEMAS).

4. Implementasi metode

Implementasi metode dalam tugas akhir ini menggunakan bahasa pemrograman python 3.6 dan kakas alat bantu Jupyter Notebook 3.6 dan Spyder 3.6 serta menggunakan perangkat lunak pendukung seperti pustaka-pustaka yang terdapat pada bahasa pemrograman Python.

5. Uji Coba dan Evaluasi

Uji coba pada tugas akhir ini dilakukan untuk mengetahui apakah metode yang digunakan sudah tepat dan sesuai dengan tujuan tugas akhir ini. Uji coba dilakukan menggunakan beberapa skenario.

Uji coba pertama dilakukan saat pengembangan program pengenalan wajah. Skenario ini dibagi menjadi dua sub-skenario, yaitu dengan melakukan uji coba pada dataset yang sama dengan masalah yang terdapat pada latar belakang dan menggunakan dataset lain.

Evaluasi dalam tugas akhir ini bermanfaat untuk mendapatkan nilai kuantitatif dari skenario uji coba yang dilakukan. Metrik yang digunakan dalam tugas akhir ini adalah metrik *accuracy*, *logarithmic loss*, *precision*, *recall* dan *F-measure.*

6. Penyusunan buku Tugas Akhir

Pada tahap ini dilakukan penyusunan laporan yang menjelaskan dasar teori dan metode yang digunakan dalam tugas akhir ini serta hasil dari implementasi metode yang telah dibuat.

1.7 Sistematika Penulisan

Buku tugas akhir ini merupakan laporan secara lengkap mengenai Tugas Akhir yang telah dikerjakan. Sistematika penulisan buku tugas akhir ini secara garis besar antara lain:

Bab I Pendahuluan

Bab ini berisi penjelasan latar belakang, rumusan masalah, batasan masalah, dan tujuan pembuatan tugas akhir. Selain itu, metodologi pengerjaan dan sistematika penulisan laporan tugas akhir.

Bab II Dasar Teori

Bab ini berisi penjelasan secara detail mengenai dasar-dasar penunjang dan teori-teori yang digunakan untuk mendukung pembuatan tugas akhir ini.

Bab III Perancangan Perangkat Lunak

Bab ini berisi penjelasan tentang rancangan dari sistem yang akan dibangun. Rancangan dalam tugas akhir ini seperti penjelasan arsitektur yang digunakan dalam implementasi tugas akhir ini.

Bab IV Implementasi

Bab ini berisi penjelasan implementasi dari rancangan yang telah dibuat pada bab sebelumnya. Implementasi

disajikan dalam bentuk *code* secara keseluruhan disertai dengan penjelasannya.

Bab V Uji Coba dan Evaluasi

Bab ini berisi penjelasan mengenai data hasil percobaan dan pembahasan mengenai percobaan yang telah dilakukan.

Bab VI Kesimpulan dan Saran

Bab ini merupakan bab terakhir yang berisi kesimpulan dari hasil uji coba yang dilakukan dan saran untuk pengembangan sistem ke depannya.

BAB II
DASAR TEORI

Pada bab ini akan dibahas mengenai teori-teori yang menjadi dasar tugas akhir ini.

2.1 Convolutional Neural Networks (CNN)

Convolutional Neural Networks (CNN) merupakan salah satu algoritma *deep learning* yang banyak dipakai dalam *machine learning* khususnya pada kasus citra. Di antara banyaknya model *deep learning* yang ada, *Convolutional Neural Networks* terbukti memiliki performa tinggi dalam klasifikasi citra.

CNN mirip dengan *neural network* yang terbentuk dari *neurons* yang memiliki *weight* (bobot) dan *biases* (bias) yang bisa dipelajari. Sebuah CNN terdiri dari satu atau lebih lapisan yang terkoneksi penuh seperti dalam jaringan saraf *multilayer* standar. Berbeda dengan neural network biasanya, lapisan pada CNN memiliki neuron yang diatur dalam 3 dimensi: lebar, tinggi dan kedalaman dimana kedalaman merujuk pada dimensi ketiga dari sebuah volume aktivasi, bukan kedalaman dari *neural network* penuh yang mengacu pada jumlah total dari lapisan dalam sebuah jaringan [2].

Untuk membangun arsitektur CNN terdapat tiga tipe layer utama yaitu c*onvolutional layer, pooling layer*, dan *fully-connected layer*. Dalam kasus citra, layer konvolusi dan layer-layer lain yang mengikutinya berukuran dua atau tiga dimensi.

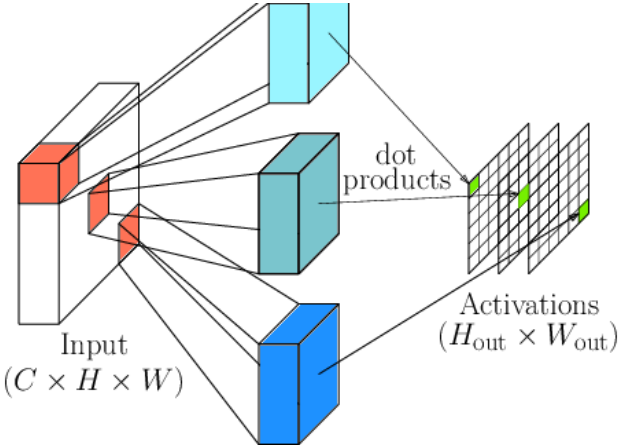
2.1.1 Convolution Layer

Konvolusi merupakan sebuah operasi yang mengubah suatu fungsi menjadi sesuatu lain, seperti mentransformasikan fungsi asli ke bentuk lain untuk mendapatkan informasi yang lebih banyak. Dalam kasus pemrosesan citra, konvolusi dipakai untuk mengaburkan citra (*blurred*), menajamkan citra, dan operasi lain seperti meningkatkan garis tepi dan bentuk timbul [3]. Dengan kata lain, konvolusi memanfaatkan apa yang disebut sebagai filter.

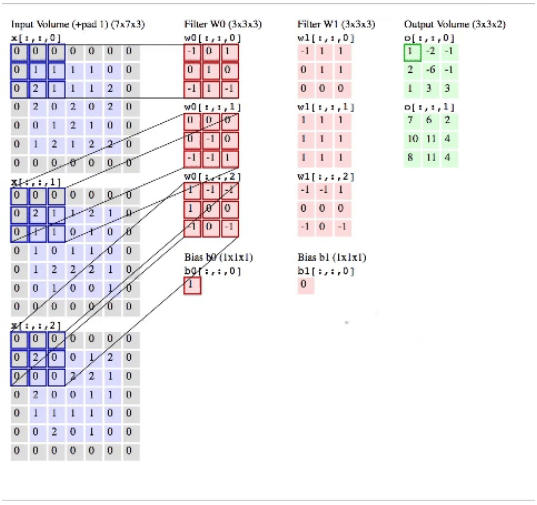
Seperti pada Gambar 2.1, filter memiliki ukuran tinggi, lebar, dan tebal tertentu. Filter ini diinisialisasi dengan nilai tertentu (*random* atau menggunakan teknik tertentu seperti Glorot), dan nilai dari filter inilah yang menjadi parameter yang akan di-*update* dalam proses *learning* [4].

Pada Gambar 2.1 terdapat bentuk input CNN yang berbentuk kotak dan setiap filternya merupakan sebuah matriks angka. Selanjutnya filter mengikuti karakteristik kotak tersebut. Pada setiap posisi gambar, dihasilkan sebuah angka yang merupakan *dot product* antara bagian gambar tersebut dengan filter yang digunakan. Dengan menggeser (*convolve*) filter di setiap kemungkinan posisi filter pada gambar, dihasilkan sebuah *activation map*.

Dalam kasus citra, layer konvolusi yang digunakan adalah *Convolution 2D* atau biasa disingkat dengan Conv2D. Hal ini dikarenakan input citra memiliki panjang dan lebar. Filter ini berukuran lebih kecil dari ukuran citra input sehingga filter dapat digeser (*convolve*) di seluruh area citra input.



Gambar 2.1 Proses pada convolutional layer [5]



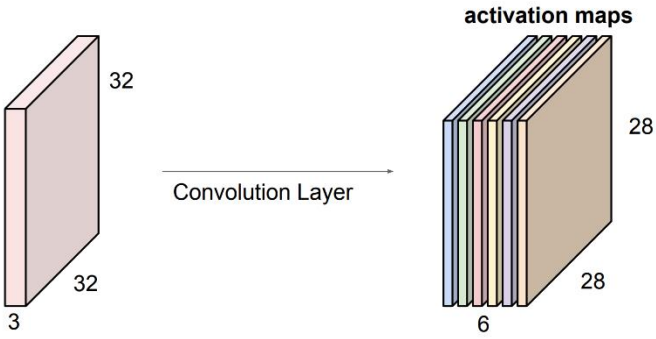
Gambar 2.2 Ilustrasi proses konvolusi dengan dua filter, W0 dan W1 [4]

Layer konvolusi pada input citra bekerja dengan memanjang melalui tiga *channel* warna yaitu *RGB* atau *Red, Green, Blue*. Setelah konvolusi dilakukan pada setiap *channel* warna, hasilnya akan dijumlahkan untuk mendapatkan citra akhir yang sudah dikonvolusi.

Ukuran spasial dai output proses pada Gambar 2.2 dapat dihitung berdasarkan formula :

$$output = \frac{N - F + 2P}{S} + 1 \qquad (2.1)$$

Di mana *N* adalah ukuran spasial (tinggi H_1 = lebar W_1) dari citra input, *F* merupakan ukuran spasial filter, *P* merupakan jumlah penambahan angka (umumnya nol) untuk menyesuaikan ukuran gambar, dan *S* adalah besaran pergeseran filter di setiap proses komputasi.



Gambar 2.3 Hasil dari convolutional layer [5]

Proses ini diulang dengan beberapa filter berbeda, hingga menghasilkan citra baru yang merupakan kumpulan dari *activation maps* yang dapat dilihat pada Gambar 2.3.

2.1.2 Pooling Layer

Pooling layer merupakan layer yang berada setelah layer konvolusi. Pada prinsipnya, pooling layer terdiri dari sebuah filter dengan ukuran dan stride tertentu yang akan bergeser pada seluruh area *feature map*.

Pooling yang biasa digunakan adalah *Max Pooling* dan *Average Pooling*. Penggunaan Max Pooling 2x2 dengan stride 2, maka pada setiap pergeseran setiap filter, nilai maksimum pada area piksel 2x2 tersebut yang akan dipilih. Sedangkan pada penggunaan Average Pooling akan dipilih nilai rata-ratanya. Ilustrasi Max Pooling dapat dilihat pada Gambar 2.4.

Tujuan dari penggunaan pooling layer adalah mengurangi dimensi dari *feature map* (*downsampling*), sehingga mempercepat komputasi karena parameter yang harus di-*update* semakin sedikit dan mengatasi *overfitting*.

2.1.3 Fully-Connected Layer

Setelah dilakukan filter pada layer konvolusi dan layer *pooling*, *feature map* yang dihasilkan masih berbentuk *multidimensional array*, sehingga harus dilakukan *flatten* atau *reshape feature map* menjadi sebuah vektor agar bisa digunakan sebagai input dari fully-connected layer.

Layer ini berfungsi untuk menghitung nilai setiap kelas. Saraf-saraf yang ada pada layer ini memiliki koneksi penuh ke seluruh aktivasi layer sebelumnya. Aktivasinya dapat dihitung dengan perkalian matriks dengan sebuah *offset* bias.

Fully-connected layer memiliki beberapa layer *hidden*, fungsi aktivasi, layer output dan fungsi *loss*. Untuk menghindari *overfitting*, CNN memiliki regularisasi Dropout. Dropout akan memberikan nilai 0 pada sebuah *neuron* secara acak ketika proses *backpropagation* dan *forward learning*. Ini adalah teknik simpel agar Neural Network tidak terjebak dalam *overfitting* [6].

Jika ditentukan regularisasi nilai Dropout adalah p, maka sebuah neuron akan di-"dropout" dengan mengganti nilai prediksinya menjadi 0 ketika hasil prediksinya bernilai p. Jadi, setiap proses *learning* akan menggunakan arsitektur CNN yang berbeda-beda sehingga menghasilkan model yang lebih *robust*. Pada saat proses *testing*, setiap prediksi dari neuron yang layernya diregularisasi dengan Dropout akan dikali dengan p sehingga hasilnya sama dengan saat proses *training*. Ilustrasi dari Dropout dijelaskan pada Gambar 2.5.

2.1.4 Optimizer

Gradient descent merupakan salah satu algoritma populer yang digunakan untuk menampilkan *optimizer* dan sejauh ini merupakan cara yang paling banyak digunakan untuk meng*optimizer neural networks* [8]. Jenis *optimizer* pada *deep learning* ada banyak seperti adagrad, sgd, adadelta, adam, rmsprop dan lainnya. Perbandingan setiap optimizer tersebut dapat dilihat

pada Gambar 2.6. Tugas akhir ini menggunakan dua parameter optimizer yaitu adam dan rmsprop.

2.1.4.1 Adam

Adaptive Moment Estimation (Adam) merupakan salah satu metode yang menghitung *learning rate* yang adaptif pada setiap parameter [8]. Adam menjaga rata-rata peluruhan secara eksponensial pada gradien sebelumnya m_t, serupa dengan momentum :

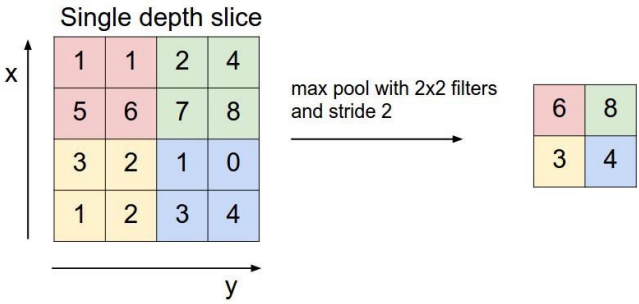
$$\widehat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{2.2}$$

$$\hat{v}t = \frac{v_t}{1 - \beta_2^t} \tag{2.3}$$

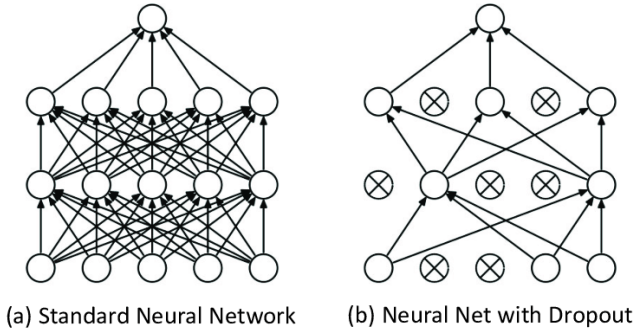
Pada Persamaan 2.2 dan 2.3 menunjukan m_t dan v_t adalah estimasi dari momen pertama (rata-rata) dan momen kedua (varian yang tidak terpusat) dari masing-masing gradien. Kemudian rumus akhir untuk parameter di-*update* menjadi :

$$\theta_t + 1 = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} - \epsilon}\,\widehat{m}_t \tag{2.4}$$

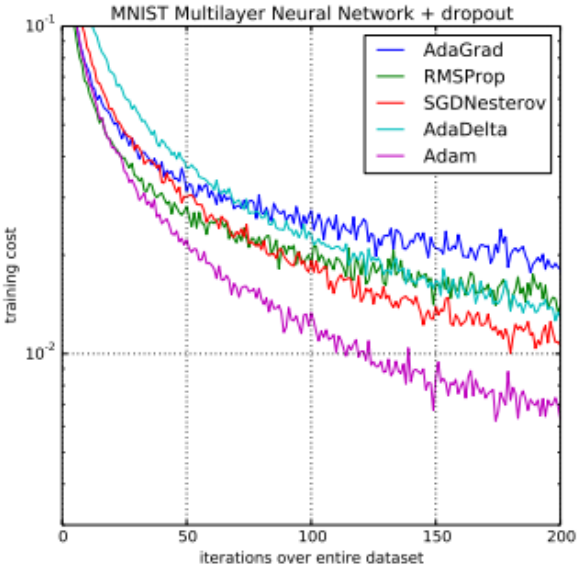
Nilai β1 sama dengan 0.9, β2 sama dengan 0.999, dan ϵ sama dengan 10^{-8}.



Gambar 2.4 Ilustrasi max pooling [7]



Gambar 2.5 Regularisasi dropout (a). Standar neural network, (b). Neural network dengan dropout. [6]



Gambar 2.6 Visualisasi perbandingan algoritma *optimizer* [9]

2.1.4.2 RMSprop

RMSprop merupakan metode tingkat pembelajaran adaptif yang diusulkan oleh Geoff Hinton pada kuliahnya di kelas *coursera* [8].

RMSprop dan Adadelta keduanya telah dikembangkan secara mandiri di waktu yang sama dari kebutuhan untuk menyelesaikan tingkat pembelajaran yang menurun secara radikal dari adagrad. Faktanya rmsprop identik dengan vektor *update* pertama dari adadelta yang diturunkan :

$$\mathrm{E}[g^2]_t = 0.9\mathrm{E}[g^2]_{t-1} + 0.1g_t^2 \tag{2.5}$$

$$\theta_t + 1 = \theta_t - \frac{\eta}{\sqrt{\mathrm{E}[g^2]_t + \epsilon}}\, g_t \tag{2.6}$$

RMSprop juga membagi tingkat pembelajaran dengan rata-rata meluruh gradien kuadrat secara eksponensial. Hinton menyarankan untuk mengatur γ sama dengan 0.9, sementara nilai default untuk tingkat pembelajaran η adalah 0.001 [8].

2.1.5 Fungsi Aktivasi

Fungsi aktivasi merupakan fitur yang sangat penting dalam dari neural network. Pada dasarnya, fungsi aktivasi berperan untuk memutuskan apakah suatu *neuron* harus diaktifkan atau tidak. Apakah informasi yang diterima oleh *neuron* relevan untuk memberikan informasi atau harus diabaikan.

$$y = Activation(\sum(weight * input) + bias) \tag{2.7}$$

Fungsi aktivasi adalah transformasi non linier yang dilakukan terhadap sinyal. Output transformasi ini kemudian dikirim dari sebuah neuron sebagai input layer selanjutnya.

Beberapa fungsi aktivasi yang digunakan pada convolutional neural network untuk tugas akhir ini adalah sebagai berikut :

2.1.5.1 ReLU (Rectified Linear Units)

ReLU (Rectified Linear Units) merupakan sebuah fungsi aktivasi yang memiliki fondasi biologis dan matematis yang kuat. Persamaan 2.8 menunjukan persamaan ReLU yang didemonstrasikan pada tahun 2011 untuk meningkatkan *training* pada *deep neural network*. Berjalan dengan nilai *treshold* 0. Di mana jika *x* lebih kecil dari 0, maka nilai *f(x)* adalah 0 dan jika lebih besar dari 0, maka nilai *f(x)* adalah nilai *x* itu sendiri.

$$f(x) = max(0, x) \tag{2.8}$$

2.1.5.2 Softmax

Solusi klasifikasi pada *deep learning* umumnya menggunakan funsgi softmax. Softmax berfungsi untuk mendistribusikan hasil *learning* atau prediksi dari layer-layer sebelumnya pada rentang [0,1] dengan hasil akhir total seluruh nilai elemennya adalah 1. Nilai tertinggi yang dihasilkan Softmax adalah kelas hasil prediksi dari CNN.

Rumus dari fungsi fungsi softmax dapat dilihat pada Persamaan 2.9 di mana *j* adalah index dari setiap elemen dari z dan K adalah jumlah seluruh elemen z, persamaan Softmax dijabarkan menjadi :

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{N} e^{z_k}}, \quad j = \{1,2, \dots, N\}. \tag{2.9}$$

2.2 Tensorflow

Tensorflow adalah sistem pembelajaran mesin (*machine learning*) yang beroperasi pada skala besar dan di lingkungan yang heterogen. *Tensorflow* menggunakan trafik aliran data untuk merepresentasikan komputasi, *state* bersama, dan operasi yang memutasi *state* tersebut [10].

Dalam tugas akhir ini, pustaka *tensorflow* digunakan sebagai *back-end* dari arsitektur CNN yang akan dibuat dan juga akan digunakan untuk memberikan visualisasi grafik dari model CNN yang telah dibuat dengan bantuan fungsi *tensorboard.*

Tensorboard merupakan sebuah aplikasi web untuk memeriksa dan memahami bagaimana *tensorflow* berjalan dan membentuk sebuah graf. Saat ini tensorboard mendukung lima bentuk visualisasi yaitu skalar, gambar, audio, histogram dan graf [11]. Pada tugas akhir ini bentuk visualisasi yang digunakan adalah bentuk skalar.

2.3 Confusion Matrix

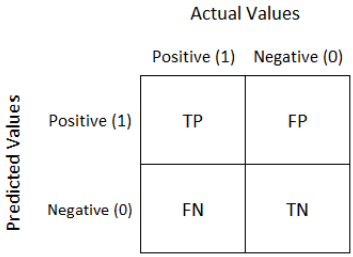
Confusion matrix adalah sebuah tabel yang dapat di-*generate* untuk *classifier* pada dataset dan dapat digunakan untuk menggambarkan performa *classifier* [12]. Tabel *confusion matrix* dijelaskan pada Gambar 2.7 yang terdiri dari istilah-istilah berikut TP, FP, FN, dan TN.

Di mana, TP adalah *True Positive* yang artinya adalah jumlah kebenaran antara hasil kalsifikasi dengan jumlah seluruh data. TN adalah *True Negative* yang menyatakan jumlah hasil klasifikasi yang diindikasikan benar, tetapi sesungguhnya salah. FP merupakan *False Positive* yaitu jumlah dari hasil klasifikasi yang diindikasi salah, tetapi sesungguhnya benar dan FN atau *False Negative* merupakan jumlah seluruh data prediksi dan sebenarnya yang salah.

Penjelasan metrik evaluasi *accuracy, precision, recall* dan *F-Measure* yang lebih rinci dipaparkan sebagaimana berikut:

2.4.1 Accuracy

Akurasi adalah tingkat kedekatan antara nilai prediksi dengan nilai aktualnya. Akurasi dapat dihitung dengan membagi nilai $TP + TN$ dengan jumlah seluruh data sehingga akurasi dapat dirumuskan sesuai dengan Persamaan 2.10.



Gambar 2.7 Confusion matrix [13]

$$accuracy = \frac{TP + \sum TN}{TP + \sum TN + \sum FP + \sum FN} \tag{2.10}$$

2.4.2 Precision

Precision atau presisi merupakan tingkat ketepatan antara gambar wajah yang diinputkan oleh pengguna dengan hasil yang dikeluarkan oleh sistem. Precision dapat dihitung dengan membagi TP dengan jumlah dari TP dan FP sehingga precision dapat dirumuskan sesuai dengan Persamaan 2.11.

$$precision = \frac{TP}{\sum TP + \sum FP} \tag{2.11}$$

2.4.3 Recall

Recall adalah nilai dari ketepatan informasi yang diprediksi relevan pada suatu kelas terhadap data asli pada kelas tersebut. *Recall* dapat dihitung dengan membagi TP dengan jumlah TP dan FN sehingga recall dapat dirumuskan sesuai dengan Persamaan 2.12.

$$recall = \frac{TP}{\sum TP + \sum FN} \tag{2.12}$$

2.4.4 F-Measure

F-Measure memanfaatkan nilai presisi dan recall. Perhitungan untuk mencari *F-Measure* atau F_1 dapat dilihat pada Persamaan 2.13.

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{2.13}$$

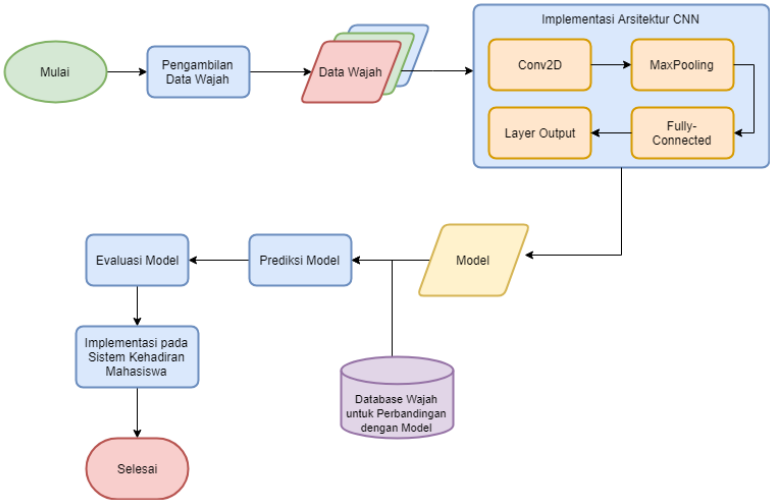
BAB III
ANALISIS DAN PERANCANGAN SISTEM

Pada bab ini akan dijabarkan desain tugas akhir yang meliputi tahap-tahap penyelesaian tugas akhir baik secara umum, maupun algoritma-algoritma yang mendukungnya secara khusus.

3.1 Analisis Metode Secara Umum

Pada tugas akhir ini akan dibangun sebuah sistem pengenalan wajah yang akan diimplementasikan pada aplikasi absensi mahasiswa. Metode yang digunakan dalam pembangunan model pengenalan wajah adalah metode *deep learning* yaitu Convolutional Neural Networks (CNN). CNN digunakan karena memiliki tingkat akurasi yang baik dalam kasus pengenalan citra [14].

Dalam tugas akhir ini, akan dirancang dalam beberapa skenario uji coba. Parameter uji coba yang dilakukan berdasarkan variasi dataset dan optimizer saat proses *learning*.



Gambar 3.1 Alir diagram perancangan sistem pengenalan wajah

Tahapan yang dilakukan dalam perancangan tugas akhir dengan skenario ini meliputi pengambilan data wajah, implementasi dataset pada arsitektur CNN, mengevaluasi model CNN, dan implementasi model ke sistem kehadiran mahasiswa. Diagram alir dari tahapan skenario ini dapat dilihat pada Gambar 3.1.

3.2 Pengambilan Dataset

Dataset merupakan kumpulan data yang saling terikat yang dapat diakses secara terpisah atau kombinasi atau dikelola sebagai kesuluruhan entitas [15]. Sebuah dataset diorganisir ke dalam suatu tipe data.

Dataset yang digunakan dalam Tugas Akhir ini adalah data wajah dari mahasiswa aktif Departemen Informatika ITS. Terdapat tiga jenis dataset yang digunakan. Pada dataset pertama, data dengan 1 variasi pengambilan (hanya menghadap ke depan) dengan jumlah kelas sebanyak 423 kelas dan masing-masing kelas terdapat 10 gambar. Beberapa *sample* gambar wajah pada dataset 1 dapat dilihat pada Gambar 3.2.

Data yang kedua merupakan data dengan 4 variasi pengambilan sudut pandang yang berbeda seperti: foto wajah dari depan tanpa ekspresi, foto wajah dari depan dengan ekspresi senyum, foto wajah dari samping kanan, dan foto wajah dari samping kiri dimana setiap sudut pandang diambil sebanyak 5 buah foto dengan jumlah data pada setiap kelas berjumlah 20 data wajah. Jumlah kelas yang terdapat pada dataset kedua dalah sebanyak 82 kelas. Beberapa *sample* gambar wajah pada dataset 2 dapat dilihat pada Gambar 3.3.

Dataset yang ketiga diambil dengan acak dengan jumlah kelas sebanyak 74 dan masing-masing kelas terdapat 20 gambar. Beberapa *sample* gambar wajah pada dataset 3 dapat dilihat pada Gambar 3.4.

Gambar 3.2 Contoh gambar wajah pada dataset 1

Gambar 3.3 Gambar sample dataset 2

Masing-masing jenis dataset wajah akan digunakan untuk data latih dan data uji model.

3.3 Desain Aplikasi CNN

Pada sub-bab ini akan dijelaskan desain umum aplikasi CNN yang akan digunakan pada tugas akhir ini. Desain umum aplikasi CNN dalam tugas akhir ini meliputi pemisahan dataset, CNN *learning*, prediksi, dan evaluasi model CNN yang sudah dibuat.

Gambar 3.4 Gambar sample dataset 3

3.3.1 Pemisahan Dataset

Sebelum dilakukan pelatihan CNN, dataset yang sudah terkumpul akan dipisahkan menjadi dua bagian, yaitu data latih dan data uji. Data latih adalah data yang menjadi bahan CNN untuk dilakukan *learning*. Komposisi pada data latih lebih banyak daripada data uji. Data latih ini saat di-*load* pada program akan dibagi lagi menjadi dua *subset*, yaitu *subset training* dan *subset validation* dengan nilai *validation split* sebesar 25%. Setelah proses *learning* selesai, model CNN terbaik akan di-*load* kembali untuk memprediksi dari setiap data uji.

3.3.2 CNN *Learning*

Sebelum melakukan proses *learning* pada CNN, penting untuk menentukan arsitektur CNN yang akan digunakan. Arsitektur yang cocok digunakan pada dataset akan memberikan performa terbaik. Arsitektur CNN yang akan digunakan pada tugas akhir ini dibahas pada sub-bab 3.4.2. Secara garis besar, aristektur CNN pada tugas akhir ini modifikasi dari arsitektur AlexNet yang

digunakan pada dataset ILSVRC, yaitu menggunakan dua layer konvolusi, dua layer *fully-connected* dan satu layer output.

Melakukan pemantauan terhadap proses *learning* CNN apakah terjadi *overfitting* atau tidak, bisa melakukan *monitoring accuracy* dan *loss* menggunakan visualisasi *tensorboard*. Penjelasan *tensorboadrd* dapat dilihat pada sub-bab 2.2.

Dalam proses *learning* dengan menggunakan *neural network*, model terbaik seringkali bukan didapatkan pada *epochs* terakhir. Untuk itu, proses *learning* pada tugas akhir ini akan menggunakan *checkpoints* dan *early stopping*.

Chekpoints adalah teknik penyimpanan model CNN setiap nilai *loss* turun pada selisih tertentu. Dengan begitu, ketika nilai *loss* yang sudah konstan atau cenderung naik, model CNN yang berhasil mencapai nilai *loss* terendah akan tersimpan.

Early stopping adalah teknik untuk memberhentikan proses *learning* CNN ketika nilai *loss* sudah tidak menunjukan penurunan yang signifikan dalam jumlah *epochs* tertentu atau model dikatakan *convergence*. Cara ini digunakan karena dapat mengoptimalkan jumlah *epochs* semaksimal mungkin, tetapi lebih hemat waktu *training* dengan memberhentikan *training* CNN ketika sudah tidak menunjukan peningkatan *learning*.

Algoritma paling penting dalam proses *learning* terdapat pada parameter-parameter yang digunakan pada *neural network*, yaitu Gradient Descent (*optimizer*). Algoritma inilah yang akan melakukan *update* pada bobot-bobot setiap *neurons*. Pada tugas akhir ini, *optimizer* yang digunakan adalah Adam dan RMSprop dengan nilai *learning rate* yang dipakai masing-masing 0,001 dan 0,0005.

3.3.3 Prediksi

Langkah ini dilakukan untuk mengukur kemampuan model CNN terbaik dari hasil *training* sebelumnya dalam memprediksi. Model CNN diberi masukan dari data uji. Pada setiap gambar

wajah yang masuk, layer ouput dari CNN akan memberikan probabilitas dari masing-masing kelas dengan rumus fungsi aktivasi *softmax*. Sebuah neuron ke-i dengan nilai probalilitas output tertinggi adalah hasil dari prediksi CNN.

3.3.4 Evaluasi

Setelah dilakukan prediksi pada model, langkah selanjutnya adalah mengukur performa model. Pada tahap ini, hasil prediksi model dibandingkan dengan *groudtruth*. Untuk memudahkan, dibuatlah sebuah *confusion matrix* untuk menghitung nilai akurasi, *precision*, *recall* dan *F-Measure* dari setiap evaluasi model.

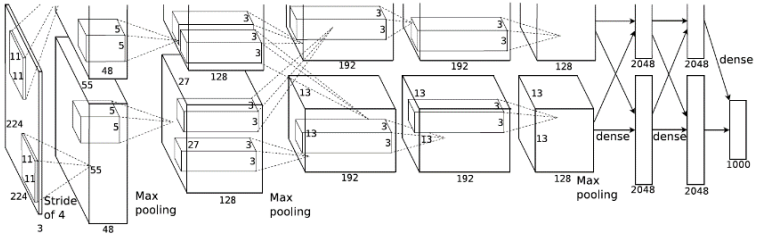
3.4 Desain Arsitektur CNN

Pada sub-bab ini akan dijelaskan desain arsitektur CNN. Terdapat banyak arsitektur CNN dalam pengenalan wajah yang sudah cukup *robust*. Arsitektur CNN yang dimaksud seperti *LeNet-5* yang pertama kali dipublikasi pada tahun 1998, *AlexNet* (2012), *ZFNet* (2013), *GoogleNet/Inception* (2014), *VGGNet* (2014) dan lain sebagainya.

Arsitektur CNN yang digunakan sangat berpengaruh sekali dalam performa klasifikasi. Maka dari itu, untuk menentukan jumlah layer, parameter, atau nilai variabel dalam CNN perlu melalui eksperimen. Arsitektur CNN yang digunakan dalam tugas akhir ini adalah arsitektur *AlexNet* dan arsitektur CNN biasa yang digunakan untuk dataset Sikemas.

3.4.1 Arsitektur AlexNet pada Dataset ILSVRC

Arsitektur AlexNet menerima penghargaan sebagai model terbaik dalam melatih dataset dalam kompetisi *ImageNet Large-Scale Visual Recognition Competition (ILSVRC)* 2010 dan 2012. Arsitektur AlexNet yang dimaksud berupa lima layer konvolusi dan tiga layer *fully-connected*. Ilustrasi dari arsitektur AlexNet dapat dilihat pada Gambar 3.5.



Gambar 3.5 Arsitektur AlexNet [17]

Fungsi aktivasi ReLU digunakan setelah layer konvolusi dan dan layer *fully-connected*. Regularisasi dropout diaplikasikan layer fully-connected pertama dan kedua [16].

Ukuran gambar input yang digunakan oleh arsitektur asli AlexNet adalah 224 x 224. Layer konvolusi pertama memfilter gambar input 224 x 224 x 3 dengan 96 kernel yang berukuran 11 x 11 x 3 dengan ukuran stride 4 piksel. Layer max pooling dilakukan setelah layer konvolusi pertama dan sebelum layer konvolusi kedua. Layer konvolusi kedua mengambil input berupa output dari layer konvolusi pertama dan melakukan filter 256 kernel yang berukuran 5 x 5 x 48. Layer max pooling dilakukan setelah layer konvolusi kedua. Layer konvolusi ketiga, keempat dan kelima terhubung satu sama lain tanpa penambahan layer max pooling dan layer normalisasi. Layer konvolusi ketiga memiliki 384 kernel dengan ukuran 3 x 3 x 256 terhubung dengan output dari layer konvolusi kedua. Layer konvolusi keempat memiliki 384 kernel yang berukuran 3 x 3 x 192 dan layer konvolusi ke lima memiliki 256 kernel dengan ukuran 3 x 3 x 192. Layer fully-connected masing-masing memiliki 4096 neurons. Fungsi aktivasi relu diaplikasikan pada output dari setiap layer konvolusi dan layer *fully-connected* [17].

Untuk meminimalisasi terjadinya *overfitting* pada model, AlexNet melakukan regularisasi *dropout*. Neuron yang di”drop-

out" dalam hal ini tidak berkontribusi dalam operasi *pass forward* dan *backpropagation*. Nilai *dropout* yang diaplikasikan dalam arsitektur AlexNet sebesar 0.5.

3.4.2 Arsitektur AlexNet pada Dataset Sikemas

Arsitektur CNN yang digunakan pada tugas akhir ini mengadopsi arsitektur AlexNet yang sudah ada dengan jumlah layer yang lebih sedikit. Rancangan arsitektur CNN yang digunakan berjumlah empat layer, yaitu dua layer konvolusi dan dua layer *fully-connected* dengan satu layer output.

Ukuran gambar yang digunakan sebagai input adalah 96 x 96 RGB piksel. Filter pada layer konvolusi pertama bernilai 16 dengan ukuran kernel 3 x 3, dan menggunakan fungsi aktivasi relu. Layer MaxPooling pertama setelah layer Conv2D pertama berukuran 2 x 2. Sementara ukuran filter untuk layer konvolusi kedua bernilai 32 dengan ukuran kernel 3 x 3 dan ukuran strides sebesar 1 piksel. Layer konvolusi kedua mengambil input dari output layer konvolusi pertama dan pooling sebelumnya. Ukuran pooling pada layer *maxpooling* kedua sebesar 2 x 2. Neuron pada layer *hidden* jumlahnya sebanyak jumlah kelas yang ada, dan nilai masing-masing *dropout* 0.1. Fungsi aktivasi yang digunakan pada layer *hidden* yaitu relu. Gambaran umum aristektur secara lebih jelas dapat dilihat pada Gambar 3.6 dan untuk arsitektur modifikasi AlexNet yang digunakan pada tugas akhir ini dapat dilihat pada Gambar 5.1.

Percobaan dengan arsitektur ini menghasilkan nilai akurasi terbaik sebesar 97,8% dengan *loss* sebesar 0.103 untuk dataset 2.

Arsitektur CNN untuk semua dataset sampai pada layer *flatten* menghasilkan jumlah input dan output yang sama, spesifikasi dapat dilihat pada Tabel 3.1. Output dari layer *flatten* ini akan digunakan untuk input pada layer *dense*. Output yang dihasilkan dari layer *dense* sampai layer output setelah dilakukan regularisasi *dropout*

terakhir jumlahnya akan menyesuaikan jumlah kelas yang ada pada dataset yang digunakan untuk pembuatan model.

Tabel 3.1 menunjukan spesifikasi dari arsitektur CNN yang dipakai dalam implementasi sistem pengenalan wajah.

3.5 Desain Aplikasi Sistem Kehadiran Mahasiswa

Pada sub-bab ini akan dijelaskan desain dari aplikasi kehadiran mahasiswa yang akan digunakan untuk implementasi pengenalan wajah menggunakan metode CNN.

3.5.1 Definisi Sistem Kehadiran Mahasiswa

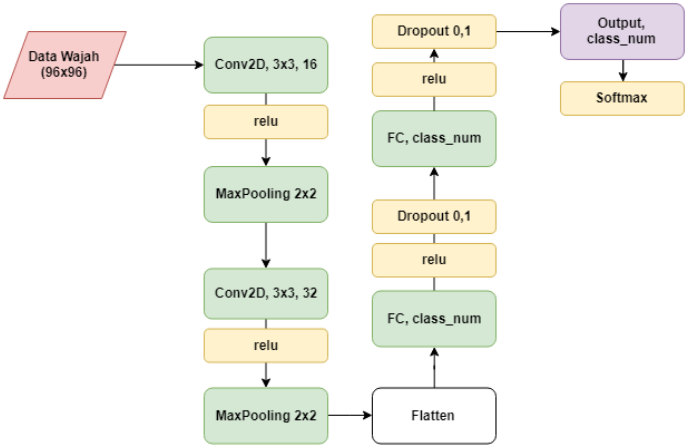
Sistem Kehadiran Mahasiswa (SIKEMAS) merupakan sebuah aplikasi sistem kehadiran yang mengimplementasikan pencocokan wajah (*face recognition*) ke dalam aplikasi mobile. Aplikasi ini diimplementasikan dengan menggunakan pustaka (*library*) Android Face Recognition with Deep Learning di mana di dalamnya terdapat algoritma dan pustaka-pustaka lainnya yang berhubungan dengan proses pengenalan wajah [18].

3.5.2 Diagram Alir Sistem Kehadiran Mahasiswa

Gambar 3.7 menunjukan alur kerja sistem kehadiran mahasiswa secara umum dalam melakukan absensi dengan pengenalan wajah. Berdasarkan diagram alir pada Gambar 3.7, mahasiswa yang akan melakukan absensi harus *log in* terlebih dahulu kemudian melakukan validasi lokasi kelas dengan memindai *QR Code* kelas yang tersedia.

Apabila kelas sesuai, maka mahasiswa dapat melakukan pengambilan wajah. Setelah wajah diambil akan dilakukan proses konvolusi oleh model CNN yang sudah dipasang pada sistem.Kemudian wajah mahasiswa yang sudah pernah dilakukan training sebelumnya akan diprediksi. Apabila wajah sesuai dengan pemilik akun dan terdapat pada data latih yang ada pada model CNN, maka *record* akan disimpan pada server. Apabila wajah tidak sesuai dengan pemilik akun, pengguna akan ditolak dan

diminta untuk melakukan pengambilan wajah kembali hingga proses prediksi dilakukan sampai benar. Penjelasan terkait arsitektur sistem secara lebih rinci terdapat pada sub-bab 3.5.3.

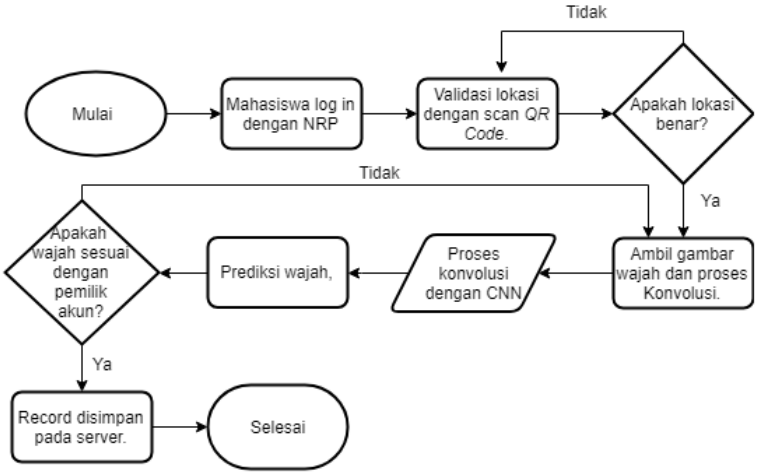


Gambar 3.6 Arsitektur CNN pada dataset SIKEMAS

Tabel 3.1 Spesifikasi arsitektur CNN

Layer	Input	Output	Spesifikasi
Input layer	(None, 96, 95, 3)		-
Conv2D Layer 1	(None, 96,96, 3)	(None, 94, 94, 16)	Filter : 3 x 3 – 16 Strides : - Activation : relu
MaxPooling2D Layer 1	(None, 94, 94, 16)	(None, 47, 47, 16)	Kernel : 2 x 2
Conv2D Layer 2	(None, 47, 47, 16)	(None, 45, 45, 32)	Filter : 3 x 3 – 32 Strides : 1 x 1 Activation : relu

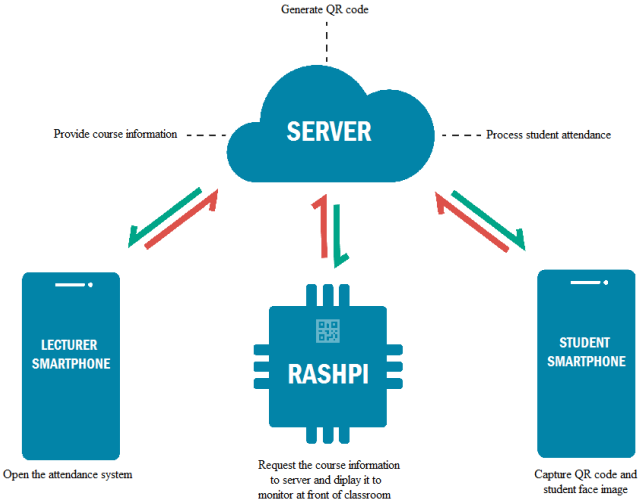
MaxPooling2D Layer 2	(None, 45, 45, 32)	(None, 22, 22, 32)	Kernel : 2 x 2
Flatten	(None, 22, 22, 32)	(None, 15488)	-
FC Layer 1	(None, 15488)	(None, Jumlah Kelas)	Activation : relu
Dropout Layer	(None, Jumlah Kelas)		0,1
FC Layer 2	(None, Jumlah Kelas)		Activation : relu
Dropout Layer	(None, Jumlah Kelas)		0,1
Output Layer	(None, Jumlah Kelas)		Activation : softmax



Gambar 3.7 Diagram alir sistem kehadiran

3.5.3 Arsitektur Sistem Kehadiran Mahasiswa

Gambar 3.8, menunjukan arsitektur sistem kehadiran mahasiswa lebih spesifik. Pengguna sistem adalah dosen pengampu mata kuliah dan mahasiswa.



Gambar 3.8 Arsitektur aplikasi sistem kehadiran

Dosen pengampu mata kuliah dapat melakukan aktivasi sistem kehadiran sesuai dengan mata kuliah yang sedang diampu melalui ponsel. Sementara ponsel mahasiswa dapat melakukan pemindaian *QR code* dan melakukan pengambilan wajah untuk absensi.

Server menyediakan layanan untuk me-*generate QR code* kelas, menyediakan informasi kelas, dan memroses data kehadiran mahasiswa. *QR code* yang di-*generate* oleh server dikirimkan ke *raspberry pi* untuk ditampilkan pada monitor yang terdapat pada tiap kelas. Pada monitor yang tersambung dengan *raspberry pi* ini akan ditampilkan *QR code* untuk keperluan pemindaian lokasi kelas oleh mahasiswa yang akan melakukan absensi.

Model CNN yang akan melakukan pengenalan wajah mahasiswa dipasang pada server. Saat mahasiswa melakukan absensi data wajah yang diambil akan dikirim ke server supaya proses konvolusi yang dilakukan tidak lama dan membebani

resource pada ponsel. Hasil prediksi dari server kemudian akan dikirim kembali oleh server ke ponsel.

[Halaman ini sengaja dikosongkan]

BAB IV
IMPLEMENTASI

Pada bab ini akan dijelaskan implementasi pada tugas akhir ini. Bab ini juga akan merinci *tools* yang digunakan pada tugas akhir ini beserta langkah-langkah pengerjaannya.

4.1 Lingkungan Implementasi

Pada sub bab ini akan dijelaskan lingkungan implementasi pada tugas akhir ini baik lingkungan implementasi perangkat keras dan perangkat lunak yang digunakan.

4.1.1 Lingkungan Implementasi Perangkat Keras

Pada implementasi metode tugas akhir ini, perangkat keras yang digunakan adalah perangkat keras yang mendukung *graphical processing unit* (GPU) NVIDIA GeForce. Sementara, sistem operasi yang digunakan adalah berbasis Linux demi kemudahan instalasi berbagai macam perangkat lunak yang dibutuhkan.

Spesifikasi perangkat keras yang digunakan pada lingkungan pengembangan aplikasi ini adalah sebagai berikut:

- Intel Core i7-7700 3.6GHz (8 Cores)
- NVIDIA GeForce GTX 1060 3GB
- 8GB of RAM
- Ubuntu 16.04 64-bit

4.1.2 Lingkungan Implementasi Perangkat Lunak

Sedangkan untuk environment perangkat lunak, digunakan bahasa pemrograman Python 3.6 dengan berbagai pustaka pendukung seperti *NumPy, Scikit-learn, Pandas* dan seluruh pustaka *built-in* pada Python 3.6. Semua pustaka tersebut digunakan untuk melakukan manipulasi dan rekayasa data-data mentah sehingga memenuhi kualifikasi untuk menjadi masukan pada pustaka utama.

Untuk mengimplementasikan algoritma CNN dilakukan terlebih dahulu instalasi *TensorFlow* untuk GPU yang langkah-langkahnya instalasinya pada komputer dapat diikuti pada alamat: *https://www.tensorflow.org/install*. Sangat dianjurkan untuk menggunakan GPU dikarenakan proses yang jauh lebih cepat dibandingkan dengan menggunakan CPU.

Kemudian, dilakukan instalasi *library* Keras yang prosesnya dapat diikuti di alamat: *https://keras.io.* Keras adalah pustaka yang dibangun di atas *TensorFlow*. Keras bertugas mempermudah dalam melakukan pembangunan pada arsitektur Neural Networks sehingga tidak perlu melakukan *coding-from-scratch* pada *TensorFlow*.

Pada Tabel 4.1 diuraikan *tools* dan perangkat lunak yang digunakan untuk pengerjaan tugas akhir ini.

Tabel 4.1. tools yang digunakan pada tugas akhir.

No.	Tools	Deskripsi
1	Python	Bahasa Python digunakan untuk menangani *task Image Processing* dan Machine Learning.
2	TensorFlow	Pustaka yang digunakan untuk melakukan training menggunakan Neural Networks.
3	Keras	Pustaka yang digunakan untuk menyusun tensor-tensor dari TensorFlow sehingga lebih mudah dalam membangun arsitektur CNN.
4	Scikit-Learn	Pustaka ini digunakan untuk melakukan evaluasi model. Baik *accuracy*, *precision*, *recall*, maupun *F-Measure*.
5	Panda	Pustaka yang digunakan untuk melakukan analisa data dan evaluasi model.
6	OpenCV	Pustaka ini digunakan untuk menangani kasus pemrosesan gambar.
7	Jupyter Notebook	*IDE* yang digunakan untuk membangun kode program berbasis python.

8	Spyder	*IDE* yang digunakan untuk membangun kode program berbasis python.

4.2 Tahap-Tahap Implementasi

Tahap-tahap implementasi pada tugas akhir ini secara garis besar ada 5 tahap :

- Skenario Data
- Implementasi arsitektur CNN
- Implementasi Prediksi Model
- Implementasi Evaluasi Model
- Implementasi Model pada SIKEMAS

Tabel 4.1 menjelaskan skenario dataset yang digunakan dalam tugas akhir ini.

Tabel 4.1 Skenario dataset

Skenario Data	**Nama Data**
Data Wajah 1 Variasi	Gasal-2017
Data Wajah 4 Variasi	Genap-2017
Data Wajah Pengambilan Bebas	Gasal-2018

4.2.1 Implementasi *Load* Dataset

Tahap pertama untuk membangun model *machine learning* adalah mengambil dataset sebagai input. Berikut ini Kode Sumber 4.1 merupakan kode sumber untuk mengambil dataset pada skenario data 1.

Pada implementasi load dataset ini, menggunakan *class* `ImageDataGenerator` yang berfungsi untuk membaca dan mengambil langsung data dari direktorinya. Data gambar yang sudah diambil kemudian skala warna dikonversi menjadi dari 0-255 menjadi 0-1 dengan parameter `rescale` yang terdapat pada

fungsi `ImageDataGenerator`. Selanjutnya, dataset dipisahkan menjadi data latih dan data uji dengan parameter `validation_split` dengan perbandingan 80% data untuk data latih dan 20% data untuk data uji.

Class `ImageDataGenerator` memiliki dua *method()* yaitu *`flow()`* dan *`flow_from_directory()`* untuk membaca gambar dari sebuah *numpy array* dan folder yang berisi data gambar. Pada tugas akhir ini, *method* *`flow_from_directory()`* dipilih sebagai metode untuk membaca data gambar wajah yang dapat dilihat pada Kode Sumber 4.1 baris 8 dan baris 15.

Method *`flow_from_directory()`* memiliki beberapa parameter diantaranya, lokasi direktori gambar, `target_size` untuk menentukan ukuran gambar yang dibaca, `batch_size` untuk menentukan ukuran setiap *batch* yang digunakan, `class_mode` untuk menentukan jenis kelas yang terdapat pada dataset, dan `subset` yaitu sebuah parameter untuk menentukan dataset akan digunakan sebagai data latih dan data uji.

```
1.  # load dataset dengan ImageDataGenerator
2.  image_path = os.getcwd()+"/2017-gasal/training-images"
3.  train_data = ImageDataGenerator(
4.                      rescale=1./255,
5.                      validation_split=0.25
6.                      )
7.
8.  train_generator = train_data.flow_from_directory(
9.                      image_path,
10.                     target_size=(img_size, img_size),
11.                     batch_size=batch_size,
12.                     class_mode='categorical',
13.                     subset='training')
```

```
14.
15.  test_generator = train_data.flow_from_directory(
16.                      image_path,
17.                      target_size=(img_size, img_size),
18.                      batch_size=batch_size,
19.                      class_mode='categorical',
20.                      subset='validation')
```

Kode Sumber 4.1 Implementasi load dataset

4.2.2 Implementasi *Checkpoints* untuk Menyimpan Model Terbaik

Untuk mendapatkan model terbaik dari hasil *training* CNN, salah satu cara yang bisa digunakan adalah dengan menggunakan *checkpoints*. *Checkpoints* berisi informasi yang dibutuhkan untuk menyimpan kondisi terakhir dari eksperimen sehingga kita dapat mengulang *training* dari poin tersebut [19]. Model disimpan dengan format h5.

Terdapat 6 model terbaik dari setiap skenario data. Kode sumber untuk implementasi checkpoints terdapat pada Kode Sumber 4.3.

Pada implementasi *checkpoint* untuk menyimpan model terbaik ini, pustaka *keras* sudah menyediakan dengan memasukkan fungsi `EarlyStopping` pada baris kedua dan `ModelCheckpoint` pada baris ketiga Kode Sumber 4.3.

`EarlyStopping` akan berhenti melakukan *training* saat jumlah yang dimonitor telah berhenti bertambah. Argumen yang terdapat pada `EarlyStopping` antara lain `monitor` untuk menentukan jumlah yang akan dimonitor, `patience` untuk menentukan jumlah data akan berhenti pada *epoch* ke-n yang sudah ditentukan setelah tidak ada perbaikan lagi, dan `verbose` digunakan untuk mode verbositas.

`ModelCheckpoint` memiliki argumen `filepath` untuk menentukan lokasi model yang akan disimpan, `verbose` untuk mode verbositas, dan `save_best_only` apabila diatur `True`, maka model terbaik terbaru sesuai dengan jumlah yang dimonitor tidak akan ditimpa.

```
1.  # membuat checkpoint untuk menyimpan model terbaik
2.  checkpoint_path = 'checkpoints/alexnet-model-sk2-
    rmsprop.h5'
3.  early_stopper = EarlyStopping(monitor='val_loss',
                              patience=15, verbose=0,
    mode='auto')
4.  checkpointer = ModelCheckpoint(
                    filepath=checkpoint_path,
                    verbose=1,
                    save_best_only=True)
```

Kode Sumber 4.2 Implementasi checkpoints

4.2.3 Implementasi Visualiasi Tensorboard

Visualisasi dilakukan untuk mengetahui bagaimana *tensorflow* berjalan dan membentuk sebuah graf. Kode Sumber 4.3 adalah implementasi visualisasi dengan tensorboard. Class `TensorBoardWrapper()` berfungsi untuk membuat sebuah class tensorboard yang akan dipanggil pada implementasi di Kode Sumber 4.4.

Fungsi `on_epoch_end` pada class `TensorBoardWrapper()` adalah untuk mengisi properti `validation_data` yang secara jelas sepesifik terhadapa bagaimana generator bekerja. Setelah terisi, metode `on_epoch_end` memiliki akses terhadap `validation_data`.

```
1.  class TensorBoardWrapper(TensorBoard):
2.      '''Sets the self.validation_data property for use with
        TensorBoard callback.'''
3.
4.      def __init__(self, batch_gen, nb_steps,
            **kwargs):
5.          super().__init__(**kwargs)
6.          self.batch_gen = batch_gen
7.          self.nb_steps = nb_steps
8.
9.      def on_epoch_end(self, epoch, logs):
10.         imgs, tags = None, None
11.         for s in range(self.nb_steps):
12.             ib, tb = next(self.batch_gen)
13.             if imgs is None and tags is None:
14.                 imgs = np.zeros((self.nb_steps *
                    ib.shape[0], *ib.shape[1:]),
                    dtype=np.float32)
15.                 tags = np.zeros((self.nb_steps *
                    tb.shape[0], *tb.shape[1:]),
                    dtype=np.uint8)
16.             imgs[s * ib.shape[0]:(s + 1) *
                            ib.shape[0]] = ib
17.             tags[s * tb.shape[0]:(s + 1) *
                    tb.shape[0]] = tb
18.         self.validation_data = [imgs, tags,
            np.ones(imgs.shape[0]), 0.0]
19.
20.         return super().on_epoch_end(epoch, logs)
```

Kode Sumber 4.3 Implementasi visualisasi tensorboard

Penggunaan class `TensorBoardWrapper()` diimplementasikan pada Kode Sumber 4.4.

```
1.  #implementasi visualisasi dengan tensorboard
2.  tensorboard = TensorBoardWrapper(test_generator,
    log_dir="tensorboard/{}".format(time()),
    histogram_freq=1, batch_size=batch_size,
    write_graph=True, write_grads=False, write_images=False,
    embeddings_freq=0, embeddings_layer_names=None,
    embeddings_metadata=None, embeddings_data=None,
    nb_steps=1)
```

Kode Sumber 4.4 Implementasi visualisasi tensorboard

4.2.4 Implementasi Arsitektur AlexNet

Pada sub bab ini akan dijelaskan tahapan implementasi arsitektur AlexNet yang digunakan pada tugas akhir ini. Implementasi artsitektur AlexNet dapat dilihat pada Kode Sumber 4.5.

Model yang dipakai pada tugas akhir ini adalah `Sequential()`. Sesuai dengan arsitektur pada sub-bab 3.3.2. implementasi CNN menggunakan empat hidden layer, yaitu dua konvolusi layer dan dua *fully-connected* layer. Fungsi yang digunakan pada hidden layer adalah relu dengan nilai dropout masing-masing 0.1 yang dapat dilihat pada baris 5 sampai dengan baris 28 Kode Sumber 4.5.

Sementara output layer menggunakan fungsi aktivasi softmax dengan jumlah *Dense* sejumlah kelas yang ada, yang dapat dilihat pada baris 32 dan 32 Kode Sumber 4.5.

```
1.  # membuat arsitektur model
```

```
2.  model = Sequential()
3.
4.  # 1st Convolutional Layer
5.  model.add(Conv2D(filters=16, kernel_size=(3,3),
    input_shape=(img_size, img_size, 3), padding='valid',
    data_format='channels_last'))
6.  model.add(Activation('relu'))
7.  # Max Pooling
8.  model.add(MaxPooling2D(pool_size=(2,2)))
9.
10. # 2nd Convolutional Layer
11. model.add(Conv2D(filters=32, kernel_size=(3,3),
    strides=(1,1), padding='valid'))
12. model.add(Activation('relu'))
13. # Max Pooling
14. model.add(MaxPooling2D(pool_size=(2,2)))
15.
16. # Passing it to a Fully Connected layer
17. model.add(Flatten())
18. # 1st Fully Connected Layer //4096
19. model.add(Dense(num_class))
20. model.add(Activation('relu'))
21. # Add Dropout to prevent overfitting
22. model.add(Dropout(0.1))
23.
24. # 2nd Fully Connected Layer
25. model.add(Dense(num_class))
26. model.add(Activation('relu'))
27. # Add Dropout
```

```
28.  model.add(Dropout(0.1))
29.
30.  # Output Layer
31.  model.add(Dense(num_class))
32.  model.add(Activation('softmax'))
```

Kode Sumber 4.5 Implementasi arsitektur CNN

4.2.5 Implementasi *Compile* Model

Setelah model dibuat pada arsitektur CNN kemudian model di*compile* dengan tiga parameter yaitu *optimizer*, *loss* dan *metrics*. Pada tugas akhir ini menggunakan dua parameter *optimizer* yang berbeda yaitu adam dan rmsprop. Implementasi dari kompilasi model dengan parameter *optimizer* adam dapat dilihat pada Kode Sumber 4.6 sementara implementasi kompilasi model dengan paramter *optimizer* rmsprop terdapat pada Kode Sumber 4.7.

Pada Kode Sumber 4.6 baris 2 merupakan inisialisasi *optimizer* yang akan digunakan untuk kompilasi yaitu *optimizer* adam. Sementara baris 3 adalah fungsi untuk melakukan kompilasi model dengan memanggil `model.compile()`.

```
1.  # compile & summary model
2.  adam = optimizers.Adam(lr = 0.0005,
                           beta_1 = 0.9,
                  beta_2 = 0.9,
                           decay = 0,
                           amsgrad=False)
3.  model.compile(optimizer=adam,
            loss='categorical_crossentropy',
            metrics=['accuracy'])
```

Kode Sumber 4.6 Implementasi kompilasi model dengan parameter optimizer adam

Pada Kode Sumber 4.7 baris 2 menjelaskan inisialisasi *optimizer* rmsprop dan baris 3 merupakan fungsi untuk melakukan kompilasi model menggunakan `model.compile()`.

```
1.  # compile & summary model
2.  rmsprop = optimizers.RMSprop(lr=0.0005, rho=0.9,
    epsilon=None, decay = 0)
3.  model.compile(optimizer=rmsprop,
            loss='categorical_crossentropy',
            metrics=['accuracy'])
```

Kode Sumber 4.7 Implementasi kompilasi model dengan parameter optimizer rmsprop

Selanjutnya untuk melihat *summary* model yang sudah dikompilasi, dapat menggunakan fungsi `model.summary()` seperti yang ada pada Kode Sumber 4.8.

```
1.  # compile & summary model
2.  model.summary()
```

Kode Sumber 4.8 Implementasi summarize model

4.2.6 Implementasi *Training* Model CNN

Tahap implementasi selanjutnya adalah melakukan *training* pada model. Pada pustaka keras, terdapat fungsi *fit_generator* untuk melakukan training pada model. Implementasi *training* model terdapat pada Kode Sumber 4.9.

Fungsi `fit_generator()` memiliki beberapa argumen. Pada baris 3 Kode Sumber 4.8 adalah argumen untuk menentukan data yang akan digunakan saat *training* model, baris 4 merupakan argumen `steps_per_epoch` yang berisi jumlah data untuk satu kali epoch. Pada baris 5 adalah argumen `epochs` yang berisi jumlah epoch untuk satu kali running, argumen `validation_data` pada

baris 6 berisi data *testing*, `validation_steps` menentukan jumlah step *testing* yaitu sejumlah `sum_data` yang didapat dari pembagian jumlah seluruh data dengan jumlah batch yang ada. Baris 8 adalah argumen callbacks yang berisi `checkpointer`, `early_stopper` dan `tensorboard`.

```
1.  # fit model
2.  model.fit_generator(
3.      train_generator,
4.      steps_per_epoch=sum_data,
5.      epochs=epochs,
6.      validation_data=test_generator,
7.      validation_steps=sum_data,
8.      callbacks=[checkpointer, early_stopper,
            tensorboard])
```

Kode Sumber 4.9 Implementasi training model

4.2.7 Implementasi Prediksi Model

Seperti yang dijabarkan pada sub-bab 3.3.3, implementasi untuk memprediksi model dilakukan untuk mengukur kemampuan model hasil *training*. Pustaka keras, telah menyediakan fungsi untuk melakukan prediksi model yaitu fungsi `predict()` yang dapat dilihat pada baris 16 Kode Sumber 4.10.

```
1.  #prediksi model
2.  def load_image(img_path, show=False):
3.       img = image.load_img(img_path,
             target_size=(img_size, img_size))
4.       img_tensor = image.img_to_array(img)
5.       img_tensor = np.expand_dims(img_tensor, axis=0)
6.
7.       if show:
```

```
8.          plt.imshow(img_tensor[0])
9.          plt.axis('off')
10.         plt.show()
11.
12.     return img_tensor
13.
14. def predict(modelnya, imgFile):
15.     test_image = load_image(imgFile, False)
16.     result = modelnya.predict(test_image) #predict keras
17.
18.     return result
```

Kode Sumber 4.10 Implementasi prediksi pada model

4.2.8 Implementasi Evaluasi Model

Tahap ini adalah evaluasi model yang sudah dilakukan *training* pada tahap sebelumnya. Evaluasi model dilakukan untuk mengetahui nilai *loss* dan *accuracy* yang terdapat pada model. Implementasi evaluasi model menggunakan fungsi `evaluate_generator()` yang sudah disediakan pustaka keras dapat dilihat pada Kode Sumber 4.11.

```
1.  # evaluasi model
2.  score = model.evaluate_generator(
3.                  test_generator,
4.                  sum_data
5.                  )
6.  print("Validation Loss : ", score[0], "Accuracy : ",
    score[1])
```

Kode Sumber 4.11 Implementasi evaluasi pada model

Selanjutnya adalah implementasi kode untuk menentukan kelas sebenarnya dan kelas prediksi. Implementasi dapat dilihat pada **Error! Reference source not found.** dengan menggunakan f ungsi `predict()` yang disediakan oleh pustaka Keras.

```
1.  y_true, y_pred = [],[]
2.  image_path = os.getcwd()+"image location"
3.  for kelas in os.listdir(image_path):
4.          kelas_path = os.path.join(image_path,kelas)
5.          for img in os.listdir(kelas_path):
6.                  r=predict(model,single_image_path)
7.                  y_classes = r.argmax(axis=-1)[0]
8.                  label = inv_map[y_classes]
9.                  print("Name={} Image ID: {}, Label: {}
                    Prosentase={}".format(single_image_path,
                    y_classes, label,r[0][y_classes]))
10.                 y_true.append(kelas)
11.                 y_pred.append(label)
```

Kode Sumber 4.12 Implementasi penentuan kelas sebenarnya dan kelas prediksi

Pada tahap ini dilakukan implementasi untuk mendapatkan hasil *confusion matrix* dari training yang dilakukan yaitu dengan menampilkan hasil prediksi model dalam bentuk *Accuracy*, *Precision*, *Recall* dan *F-Measure* sesuai dengan rumus yang terdapat pada sub-bab 2.3 sebelumnya.

Implementasi kode untuk *confusion matrix* dapat dilihat pada **Error! Reference source not found.** dengan menggunakan b eberapa fungsi yang disediakan oleh pustaka Panda dan Scikit-learn.

```
1.  cf = confusion_matrix(y_true, y_pred)
2.  className = np.arange(0,len(np.unique(y_true)))
3.  cfPandas = ConfusionMatrix(y_true,y_pred)
4.  cfPandas.print_stats()
5.  print("Accuracy : " +
    str(100*metrics.accuracy_score(y_true, y_pred)))
6.  print("Precision : " +
    str(100*metrics.precision_score(y_true, y_pred,
    average="weighted")))
```

```
7.  print("Recall : " + str(100*metrics.recall_score(y_true,
    y_pred, average="weighted")))
8.  print("f1_score : " + str(100*metrics.f1_score(y_true,
    y_pred, average="weighted")))
```

Kode Sumber 4.13 Implementasi *confusion matrix*

[Halaman ini sengaja dikosongkan]

BAB V
UJI COBA DAN EVALUASI

Pada bab ini, akan dijabarkan hasil uji coba beserta evaluasi dari sistem yang telah dibangun. CNN diuji kemampuannya memprediksi wajah dengan menggunakan metrik akurasi, *precision*, *recall*, dan *F-Measure*. Selain itu, hasil uji coba akan dievaluasi kinerjanya sehingga dapat diputuskan apakah sistem ini mampu menyelesaikan permasalahan yang telah dirumuskan di awal.

5.1 Lingkungan Uji Coba

Pada Sub-bab ini akan dijelaskan lingkungan uji coba dari sistem pengenalan wajah ini, baik perangkat keras maupun perangkat lunak yang digunakan pada tugas akhir ini. Lingkungan tersebut ditunjukan pada Tabel 5.1.

Tabel 5.1 Spesifikasi lingkungan uji coba

Perangkat	Spesifikasi
Perangkat Keras	Prosesor : Intel Core i7-7700 3.6GHz (8 Cores) GPU : NVIDIA GeForce GTX 1060 3GB Memori : 8GB
Perangkat Lunak	Sistem Operasi : Ubuntu 16.04 64-bit Perangkat Pengembang : - Spyder 3.6 - Jupyter Notebook Perangkat Pendukung : - Tensorboard 1.12.1 - Microsoft Excel

5.2 Data Uji Coba

Data yang digunakan dalam uji coba metode tugas akhir ini ada tiga jenis. Yang pertama data dengan 1 variasi pengambilan (hanya

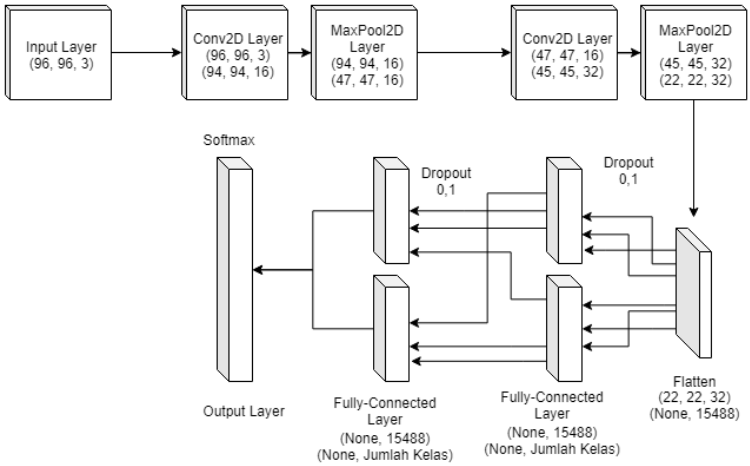
menghadap ke depan) dengan jumlah kelas sebanyak 423 kelas dan masing-masing kelas terdapat 10 gambar. Data yang kedua merupakan adata dengan 4 variasi pengambilan (menghadap ke depan tanpa ekspresi, menghadap ke depan senyum, menghadap ke kiri dan menghadap ke kanan) dengan jumlah kelas sebanyak 82 kelas dan masing-masing kelas terdapat 20 gambar. Sedangkan data yang ketiga diambil dengan acak dengan jumlah kelas sebanyak 74 dan masing-masing kelas terdapat 20 gambar.

5.3 Arsitektur Uji Coba

Penentuan arsitektur yang dipakai pada tugas akhir ini tidak secara langsung diimplementasikan pada dataset SIKEMAS. Melainkan telah melalui beberapa uji coba berdasarkan jumlah layer yang akan dipakai sehingga mendapatkan model CNN yang optimal. Karena arsitektur pada tugas akhir ini merupakan modifikasi dari model AlexNet yang ada, penulis mencoba melakukan penelitian mulai dari penggunaan 1 layer sampai 4 layer yang merupakan arsitektur paling optimal pada model CNN ini. Berikut akan dijelaskan hasil dari uji coba penggunaan layer pada tugas akhir ini terhadap dataset pertama.

Hasil uji coba pembuatan model menggunakan hanya 1 layer konvolusi menghasilkan nilai *loss* pada model sebesar 0,098 dan nilai akurasi model sebesar 97,9%. Dilihat dari nilai *loss* dan akurasi model sekilas model merupakan paling optimal, tetapi pada kenyataannya model ini tidak digunakan karena pada saat proses *learning* berlangsung model mengalami *overfitting* disebabkan tidak adanya regularisasi *dropout* pada proses *learning*. Akurasi *training* model mendekati 99% sementara akurasi validasi model hanya sebesar 73,4%.

Hasil uji coba pembuatan model menggunakan 2 layer yaitu 1 layer konvolusi dan 1 layer *fully-connected* menghasilkan nilai *loss* model sebesar 0,317 dan nilai akurasi model sebesar 94,3%. Arsitektur dengan 2 layer ini sudah menggunakan regularisasi



Gambar 5.1 Arsitektur CNN menggunakan 4 layer

d*ropout* sebesar 0,1 pada layer *fully-connected.* Meskipun output model yang dikeluarkan sudah baik, namun pada saat proses *learning* model masih mengalami *overfitting* dengan nilai akurasi *training* sebesar 86,3% dan akurasi validasi sebesar 53,8%. Maka dari itu, arsitektur ini tidak digunakan dalam tahap implementasi tugas akhir.

Hasil uji coba pembuatan model berdasarkan arsitektur selanjutnya adalah dengan menggunakan 3 layer, yaitu 2 layer konvolusi dan 1 layer *fully-connected* dengan nilai regularisasi *dropout* sebesar 0,1. Nilai *loss* pada model yang dihasilkan sebesar 0,297 dan nilai akurasi model sebesar 95,0%. Pada uji coba menggunakan 3 layer ini, masih terjadi *overfitting* pada saat proses *learning* berlangsung. Model memiliki nilai akurasi *training* sebesar 95,4% sementara nilai akurasi validasi model sebesar 65,4%. Oleh karena itu, arsitektur menggunakan 3 layer ini tidak diimplementasikan pada tugas akhir ini.

Maka dari itu, arsitektur yang diimplementasikan pada tugas akhir ini adalah arsitektur dengan menggunakan 4 layer yaitu 2

layer konvolusi dan 2 layer *fully-connected* yang dapat dilihat pada Gambar 5.1.

5.4 Skenario Uji Coba

Parameter yang digunakan pada skenario uji coba yang terdapat pada tugas akhir ini ada beberapa macam yaitu, nilai *epoch* yang digunakan, *optimizer* yang digunakan, ukuran *batch* yang dipakai, *loss function* yang dipakai, dan penggunaan *learning rate*. Tabel 5.2 menunjukan parameter uji coba yang digunakan pada tugas akhir ini.

Nilai *learning rate* dipilih 0,001 dan 0,0005 berdasarkan beberapa penelitian, nilai optimum model didapatkan pada rentang *learning rate* antara 0,001 sampai 0,0005. Sementara pemilihan *Categorical Cross Entropy* sebagai *loss function* karena dataset yang digunakan adalah dataset multilabel.

Tabel 5.2 Parameter uji coba yang dipakai

Jumlah *Epochs* (*patience*)	500 (100 *patiences*)
Optimizer	Adam dan RMSprop
Ukuran *Batch*	16
Loss Function	*Categorical Cross Entropy*
Learning Rate	0,001 dan 0,0005

5.4.1 Skenario Uji Coba Dataset 1

Pada uji coba dataset 1 nilai akurasi model tertinggi didapatkan pada percobaan dengan optimizer rmsprop dengan nilai *learning rate* 0,001 yaitu sebesar 76%, hanya saja *loss* yang dihasilkan paling besar pada percobaan ini yaitu sebesar 3,02. Sementara, nilai *loss* paling rendah yaitu sebesar 2,42 pada percobaan dengan optimizer adam dengan nilai *learning rate* sebesari 0,0005. Data hasil uji coba dapat dilihat pada

Tabel **5.3**.

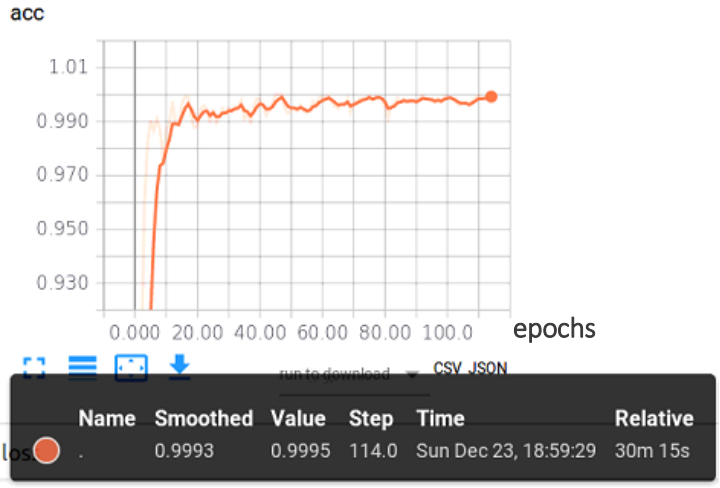
Tabel 5.3 Skenario uji coba dataset 1

Optimizer	*Learning rate*	Loss	Accuracy (%)
Adam	**0,001 (default)**	**0,239**	**95,5**
	0,0005	0,317	94,0
Rmsprop	0,001 (default)	0,525	90,1
	0,0005	0,299	93,6

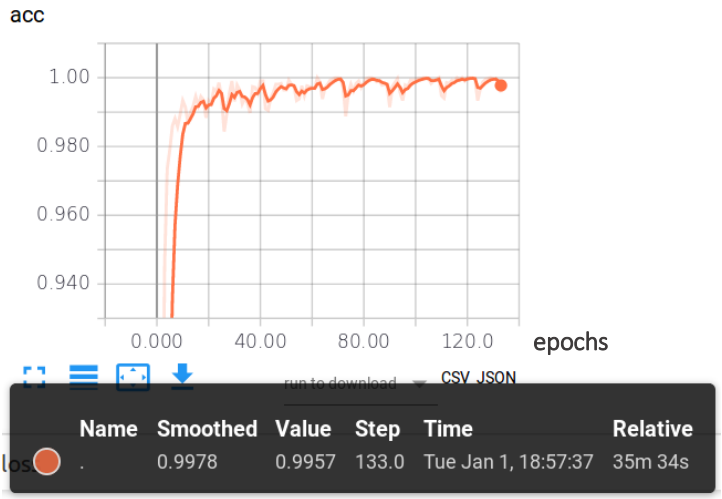
5.4.1.1 Adam Optimizer

Pada Tabel 5.3, hasil uji coba menggunakan *optimizer* adam dengan nilai *learning rate* 0,001 (*default*) menghasilkan akurasi dan *loss* masing-masing sebesar 95,5% dan 0,239. Kemudian, setelah dilakukan *tuning* terhadap nilai *learning rate*, akurasi bertambah dan *loss* berkurang menjadi 94,0% dan 0,317. Hasil uji coba dengan Untuk melihat perbandingan nilai akurasi *training*, *loss training*, akurasi validasi dan *loss* validasi ditunjukan pada grafik yang ditampilkan oleh *tensorboard*.

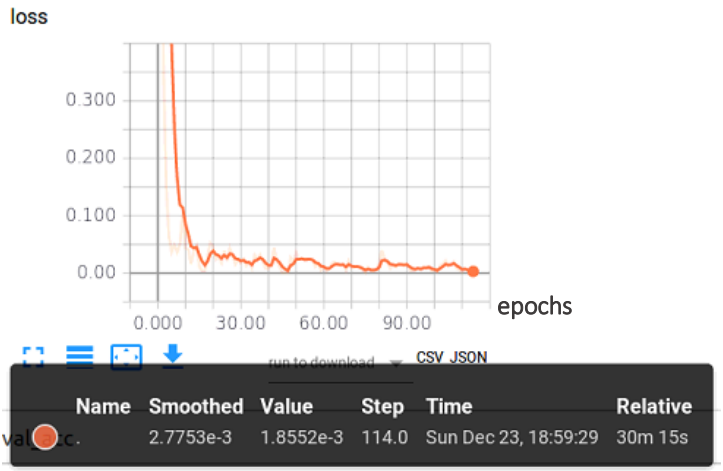
Gambar 5.2 dan Gambar 5.3 merupakan visualisasi dari akurasi masing-masing data latih, Gambar 5.4 dan Gambar 5.5 menunjukan visualisasi dari nilai *loss* pada masing-masing data latih, Gambar 5.6 dan Gambar 5.7 menunjukan visualisasi akurasi validasi pada masing-masing data uji, dan Gambar 5.8 dan Gambar 5.9 menunjukan visualiasi dari nilai validasi *loss* pada masing-masing data uji.



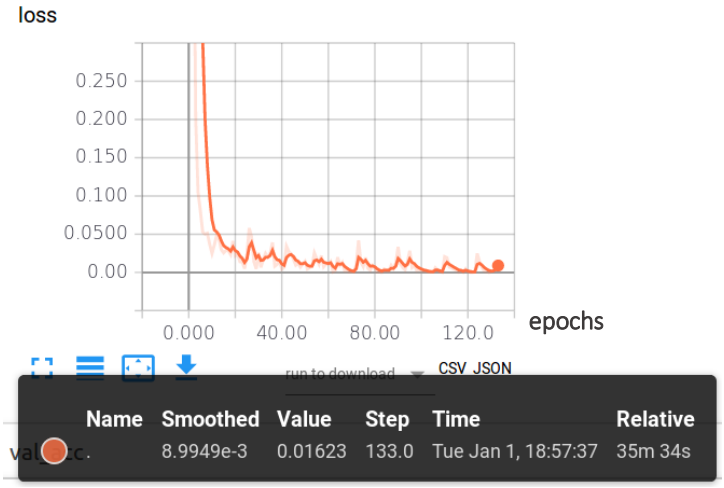
Gambar 5.2 Akurasi data latih dengan nilai *learning rate* 0,001



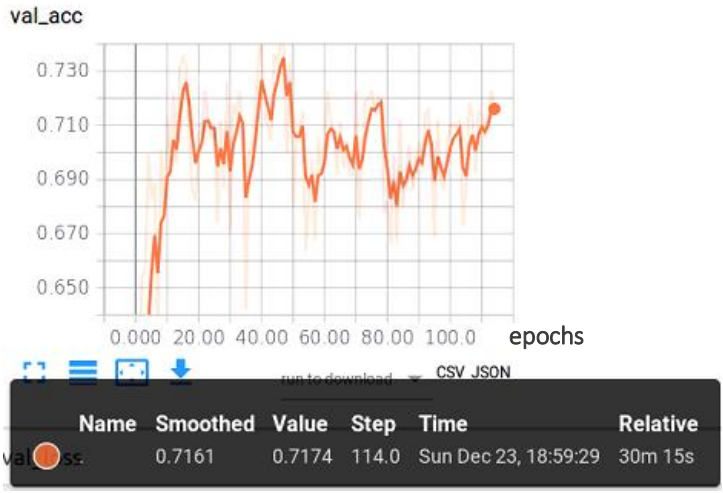
Gambar 5.3 Akurasi data latih dengan nilai *learning rate* 0,0005



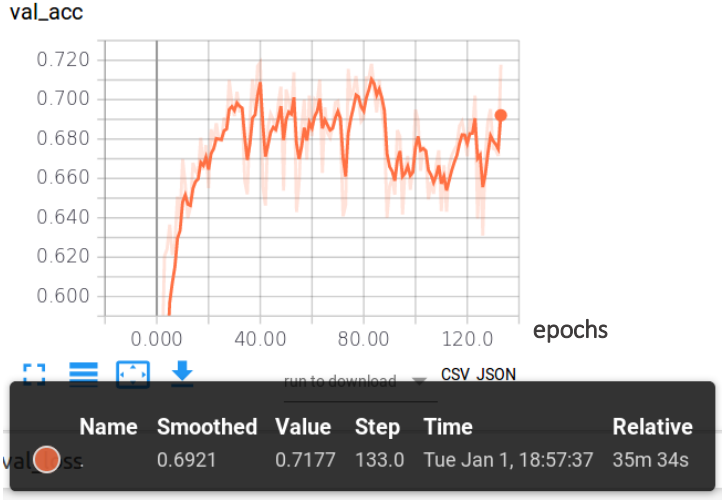
Gambar 5.4 *Loss* pada data latih dengan nilai *learning rate* 0,001



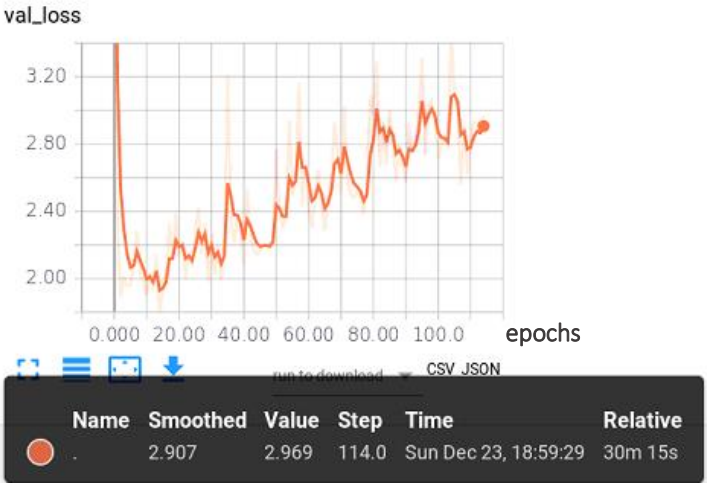
Gambar 5.5 *Loss* pada data latih dengan nilai *learning rate* 0,0005



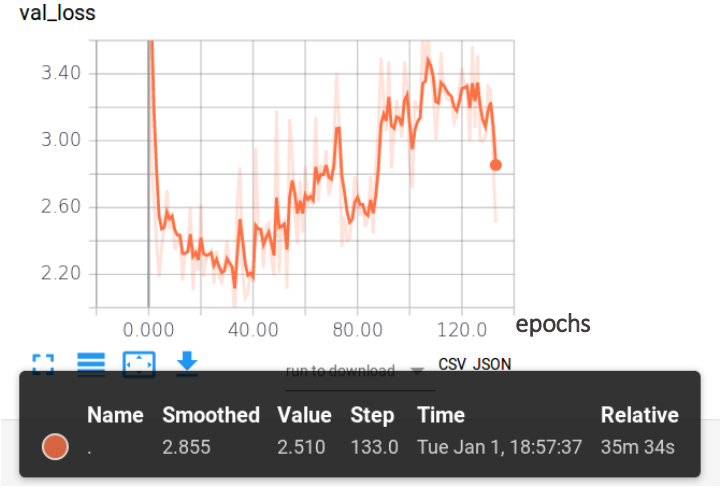
Gambar 5.6 Akurasi validasi data uji dengan nilai *learning rate* 0,001



Gambar 5.7 Akurasi validasi data uji dengan nilai *learning rate* 0,0005



Gambar 5.8 *Loss* pada data uji dengan nilai *learning rate* 0,001

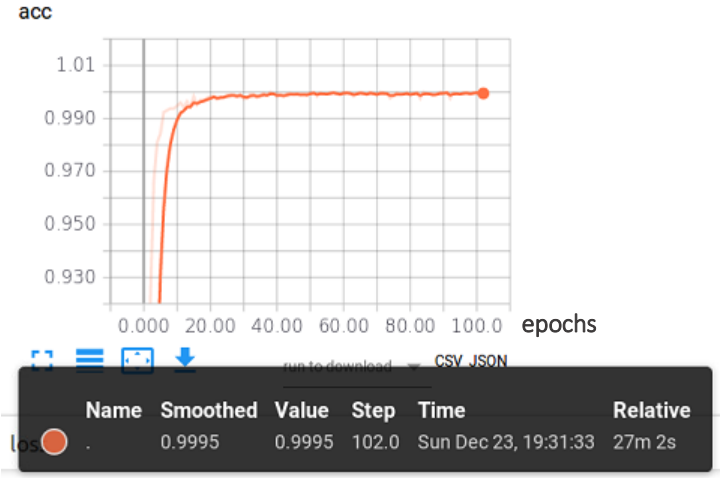


Gambar 5.9 *Loss* pada data uji dengan nilai *learning rate* 0,0005

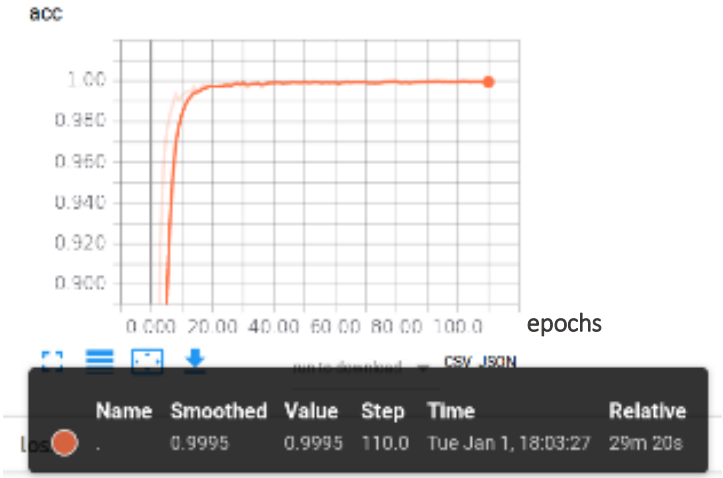
5.4.1.2 Rmsprop Optimizer

Berdasarkan Tabel 5.3, implementasi *optimizer* rmsprop memiliki nilai akurasi tidak lebih baik dibandingkan dengan menggunakan *optimizer* Adam. Nilai akurasi dan *loss* dengan *learning rate* 0,001 adalah sebesar 90,1% dan 0,525. Sementara dengan *learning rate* 0,0005 memiliki akurasi model sebesar 93,6% dan 0,299. Untuk melihat perbandingan nilai akurasi training, *loss* training, akurasi validasi dan *loss* validasi ditunjukan pada grafik yang ditampilkan oleh tensorboard.

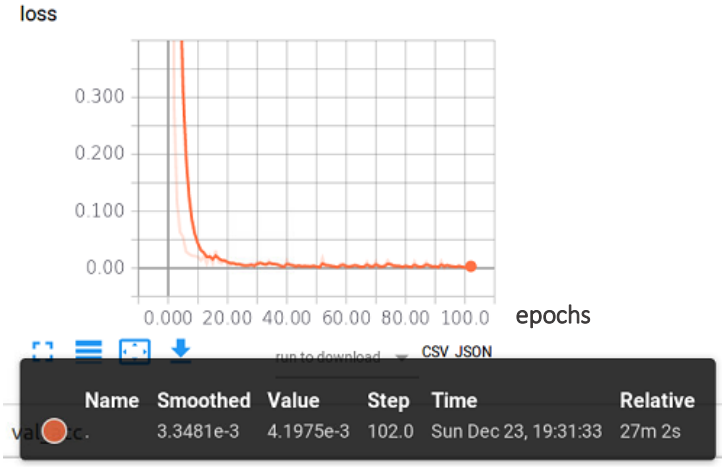
Gambar 5.10 dan Gambar 5.11 merupakan visualisasi dari akurasi data latih, Gambar 5.12 dan Gambar 5.13 menunjukan visualisasi dari nilai *loss* pada data latih, Gambar 5.14 dan Gambar 5.15 menunjukan visualisasi akurasi validasi pada data uji, dan Gambar 5.16 dan Gambar 5.17 menunjukan visualiasi dari nilai validasi *loss* pada data uji.



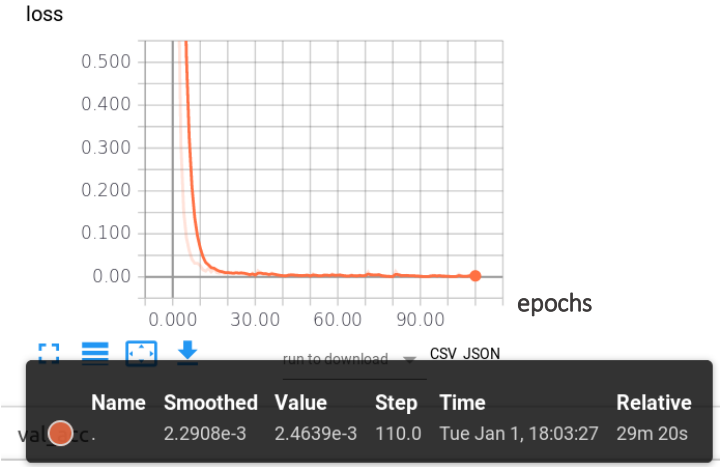
Gambar 5.10 Akurasi data latih dengan nilai *learning rate* 0,001



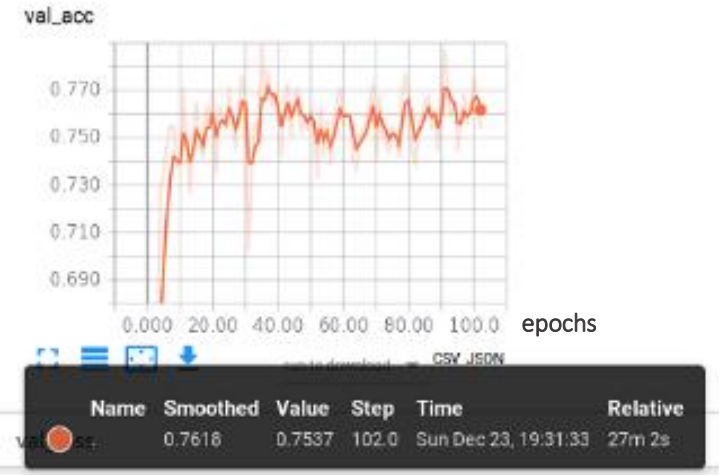
Gambar 5.11 Akurasi data latih dengan nilai *learning rate* 0,0005



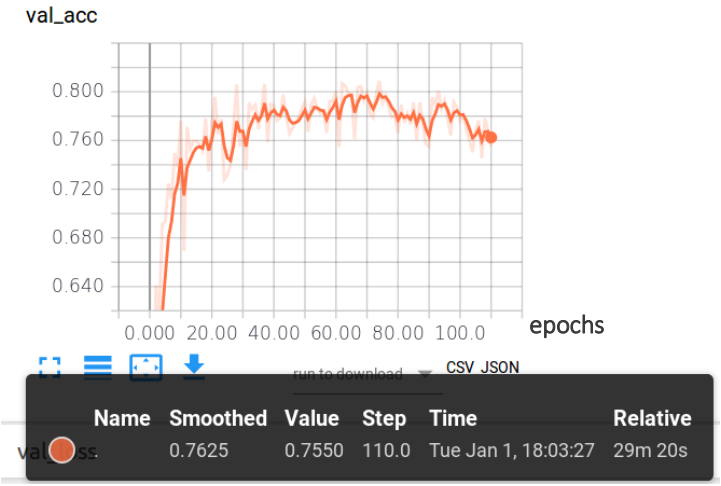
Gambar 5.12 *Loss* pada data latih dengan nilai *learning rate* 0,001



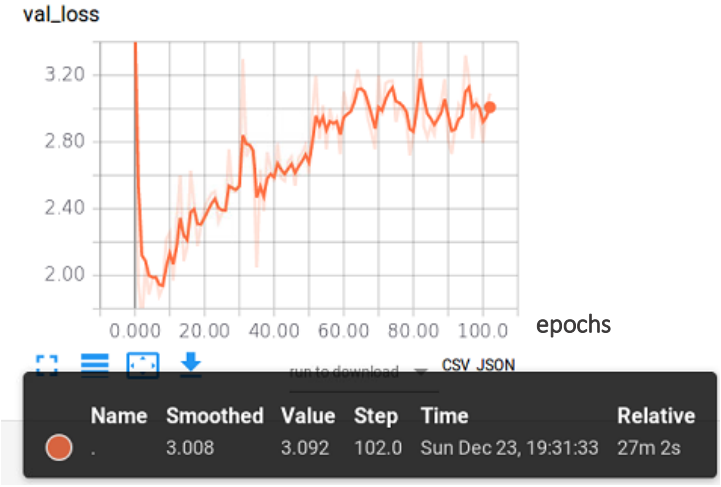
Gambar 5.13 *Loss* pada data latih dengan nilai *learning rate* 0,0005



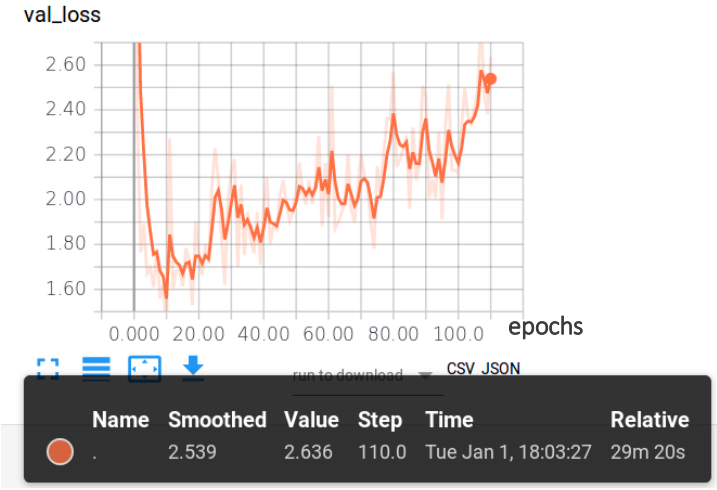
Gambar 5.14 Akurasi validasi data uji dengan nilai *learning rate* 0,001



Gambar 5.15 Akurasi validasi data uji dengan nilai *learning rate* 0,001



Gambar 5.16 *Loss* pada data uji dengan nilai *learning rate* 0,001



Gambar 5.17 *Loss* pada data uji dengan nilai *learning rate* 0,0005

5.4.2 Skenario Uji Coba Dataset 2

Tabel 5.3 menunjukan perbandingan hasil uji coba antara *optimizer* adam dan rmsprop pada dataset 2. Berdasarkan uji coba yang telah dilakukan, nilai akurasi model dan *loss* pada dataset 2 dengan *optimizer* rmsprop dengan nilai *learning rate* sebesar 0,0005 (*default*) adalah yang terbaik yaitu 97,8% dan 0,101. Dengan ini, *optimizer* rmsprop dengan nilai *learning rate* 0,0005 pada dataset 2 yang paling optimal.

Tabel 5.4 Skenario uji coba dataset 2

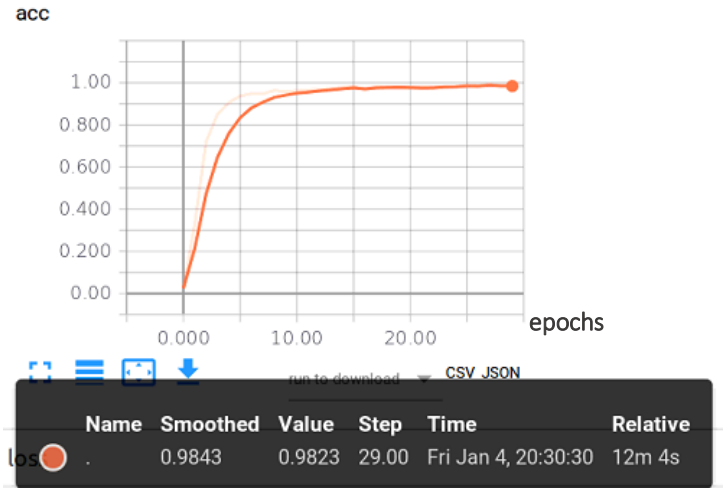
Optimizer	*Learning rate*	Loss	Accuracy (%)
Adam	0,001 (default)	0,356	91,4
	0,0005	0,103	96,7
Rmsprop	0,001 (default)	0,203	96,6
	0,0005	**0,101**	**97,8**

5.4.2.1 Adam Optimizer

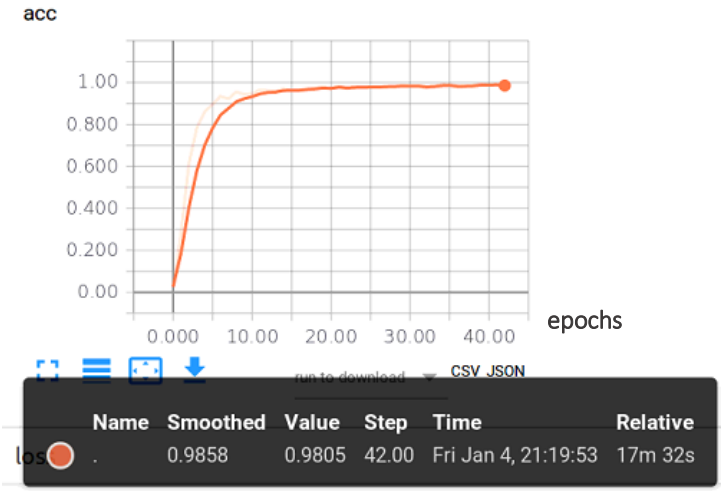
Berdasarkan data uji coba pada Tabel 5.3, meskipun akurasi dan nilai *loss* pada *optimizer* adam sudah termasuk tinggi yaitu 91,4%; 0,356 pada *learning rate* 0,001 dan 96,7%; 0,103 pada *learning rate* 0,005 namun tidak lebih baik dibandingkan menggunakan *optimizer* rmsprop. Untuk melihat perbandingan nilai akurasi training, *loss* training, akurasi validasi dan *loss* validasi ditunjukan pada grafik yang ditampilkan oleh tensorboard.

Visualisasi tensorboard untuk *optimizer* adam dengan nilai *learning rate default* dan setelah di*tune* dapat dilihat pada Gambar 5.18 sampai dengan Gambar 5.25.

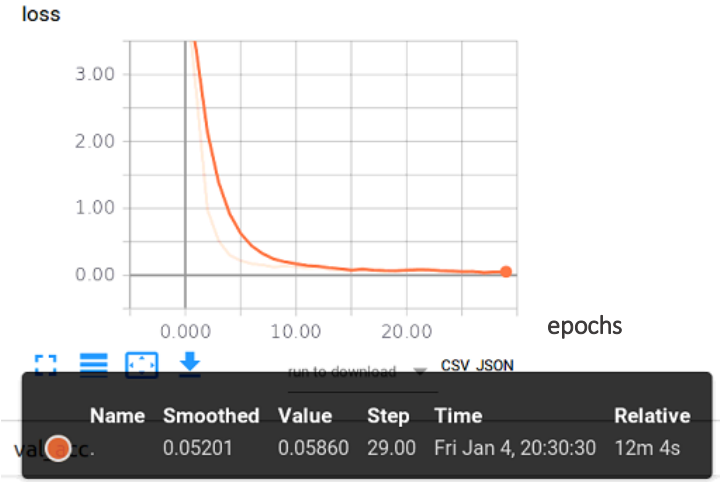
Gambar 5.18 dan Gambar 5.19 merupakan nilai akurasi pada data latih pada dua nilai *learning rate* yaitu 0,001 dan 0,0005.



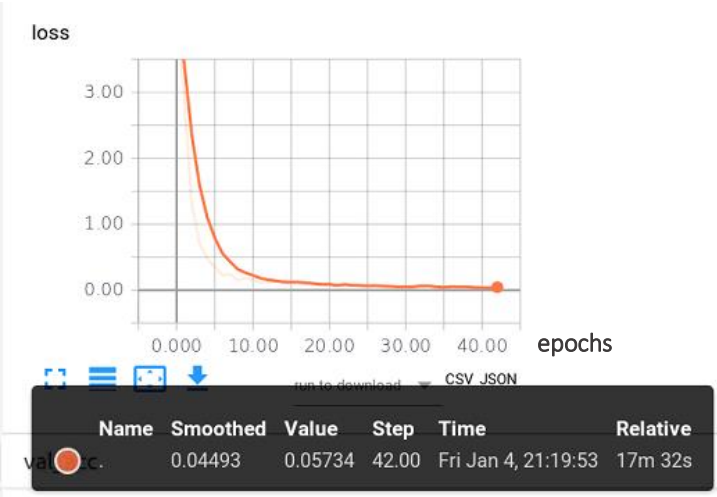
Gambar 5.18 Akurasi data latih dengan nilai *learning rate* 0,001



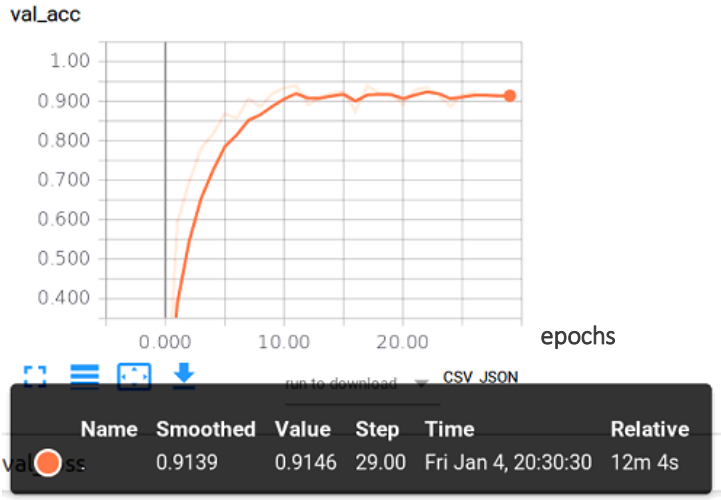
Gambar 5.19 Akurasi data latih dengan nilai *learning rate* 0,0005



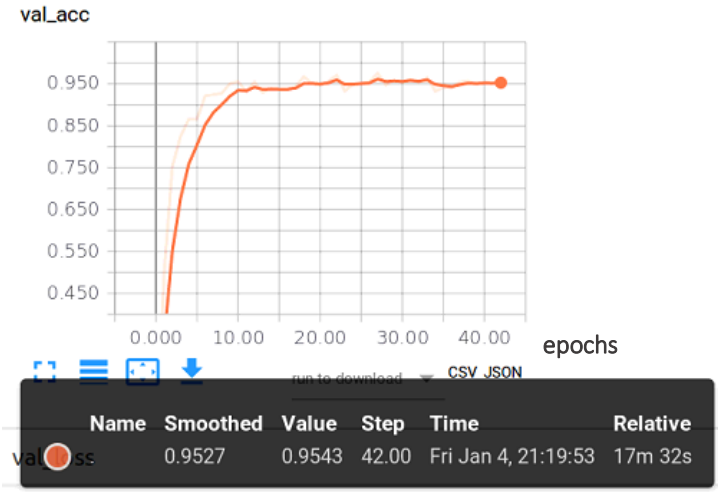
Gambar 5.20 *Loss* pada data latih dengan nilai *learning rate* 0,001



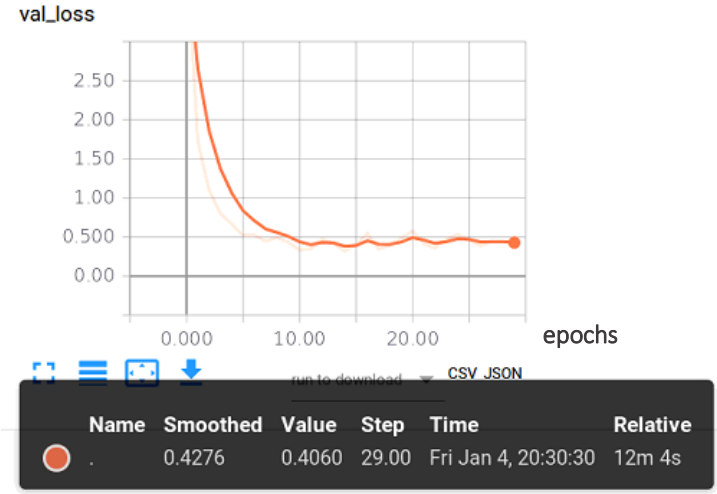
Gambar 5.21 *Loss* pada data latih dengan nilai *learning rate* 0,0005



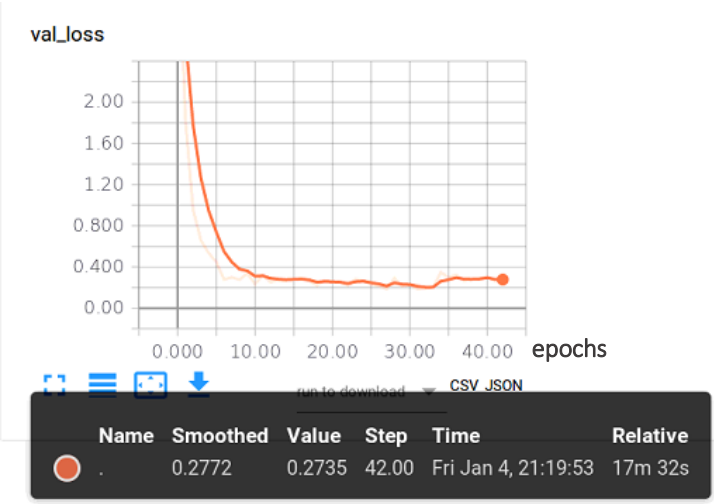
Gambar 5.22 Akurasi validasi data uji dengan nilai *learning rate* 0,001



Gambar 5.23 Akurasi validasi data uji dengan nilai *learning rate* 0,0005



Gambar 5.24 *Loss* pada data uji dengan nilai *learning rate* 0,001



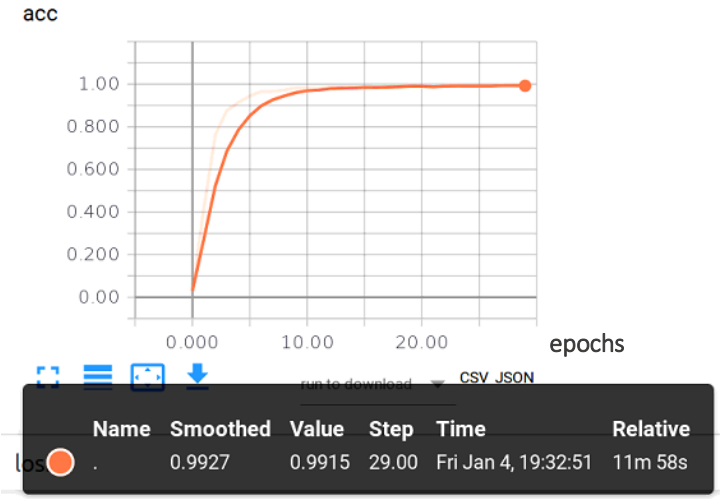
Gambar 5.25 *Loss* pada data uji dengan nilai *learning rate* 0,0005

Gambar 5.20 dan Gambar 5.21 menunjukan nilai *loss* pada masing-masing *learning rate* terhadap *optimizer* adam.

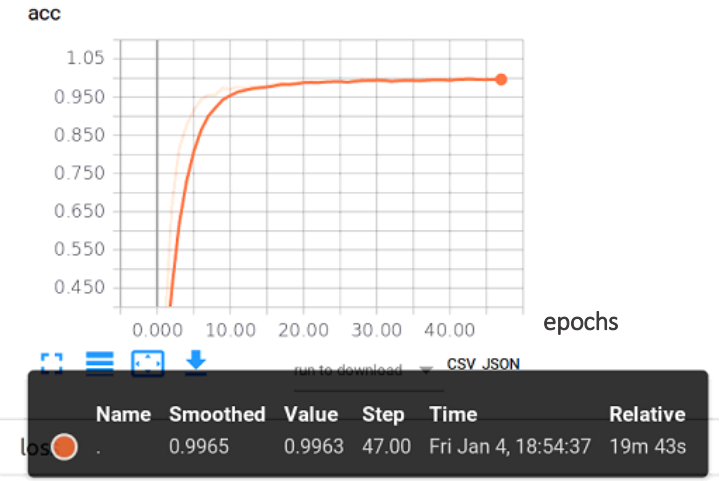
5.4.2.2 RMSprop Optimizer

Berdasarkan data uji coba pada Tabel 5.4, percobaan menggunakan *optimizer* rmsprop dengan nilai *learning rate* default lebih baik dibanding dengan nilai *learning rate* dengan nilai *learning rate* yang di*tuning* yaitu akurasi dan *loss* masing-masing sebesar 97,8% dan 0,101. Untuk melihat perbandingan nilai akurasi training, *loss* training, akurasi validasi dan *loss* validasi ditunjukan pada grafik yang ditampilkan oleh *tensorboard*.

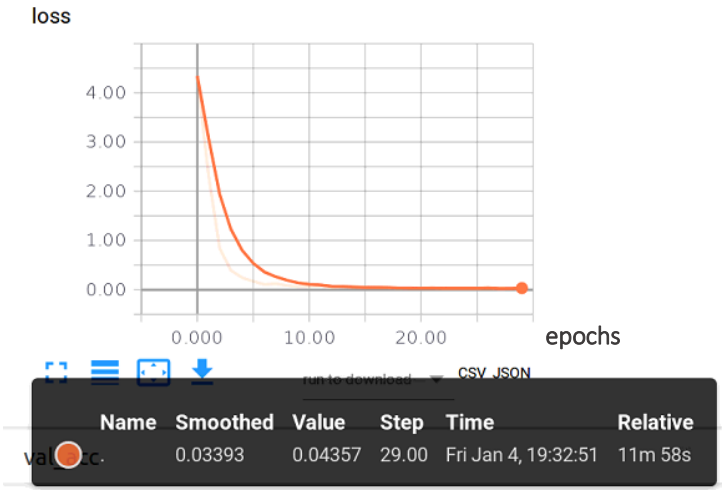
Visualisasi tensorboard pada *optimizer* rmsprop dapat dilihat pada Gambar 5.26 sampai dengan Gambar 5.33.



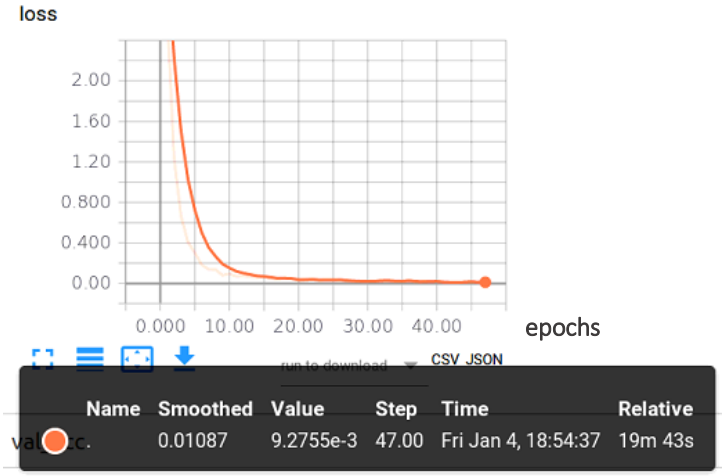
Gambar 5.26 Akurasi data latih dengan nilai *learning rate* 0,001



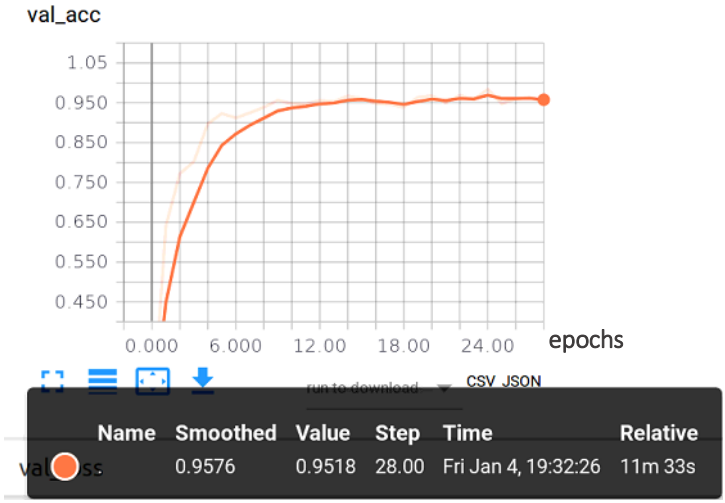
Gambar 5.27 Akurasi data latih dengan nilai *learning rate* 0,0005



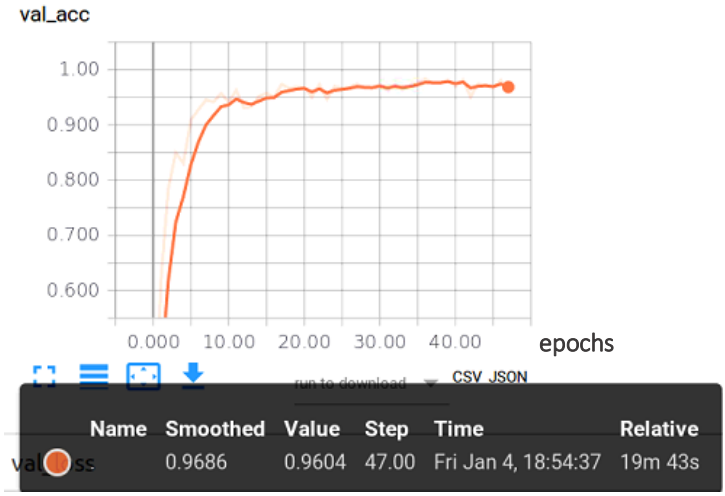
Gambar 5.28 *Loss* pada data latih dengan nilai *learning rate* 0,001



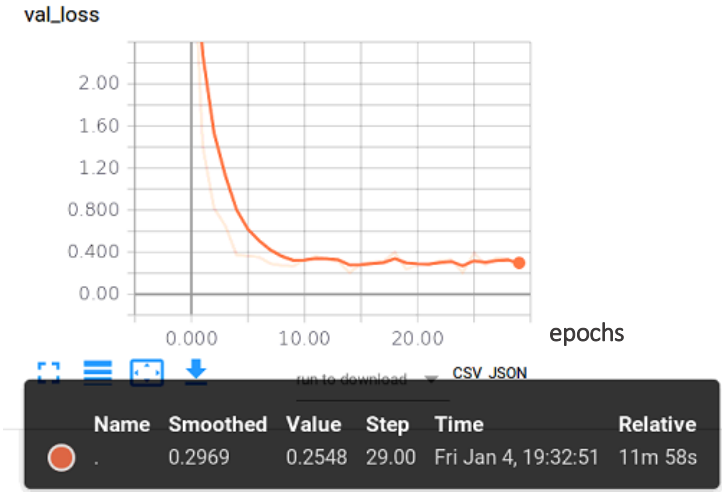
Gambar 5.29 *Loss* pada data latih dengan nilai *learning rate* 0,0005



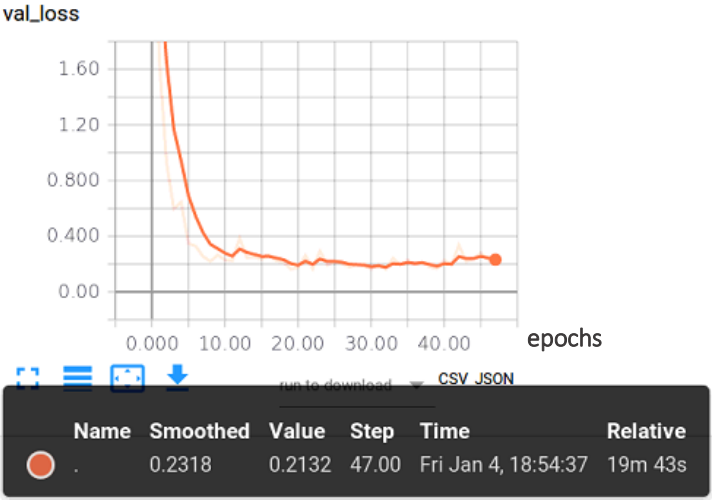
Gambar 5.30 Akurasi validasi pada data uji dengan nilai *learning rate* 0,001



Gambar 5.31 Akurasi validasi pada data uji dengan nilai *learning rate* 0,0005



Gambar 5.32 *Loss* pada data uji dengan nilai *learning rate* 0,001s



Gambar 5.33 *Loss* pada data uji dengan nilai *learning rate* 0,0005

5.4.3 Skenario Uji Coba Dataset 3

Tabel 5.4 menunjukan hasil uji coba skenario pada dataset 3. *Optimizer* adam dengan nilai *learning rate* 0,0005 memiliki nilai akurasi dan *loss* model lebih tinggi dibanding dengan *optimizer* adam dengan nilai *learning rate* 0,0005 dan *optimizer* rmsprop. Nilai akurasi dan *loss* masing-masing adalah 84,9% dan 1,18.

Tabel 5.5 Skenario uji coba dataset 3

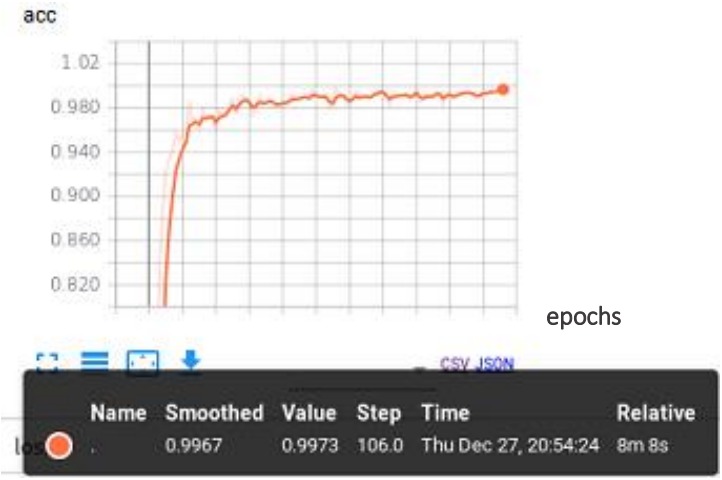
Optimizer	*Learning rate*	Loss	Accuracy (%)
Adam	0,001 (default)	1,19	77,6
	0,0005	**1,18**	**84,9**
Rmsprop	0,001 (default)	1,08	75,2
	0,0005	1,23	71,8

5.4.3.1 Adam Optimizer

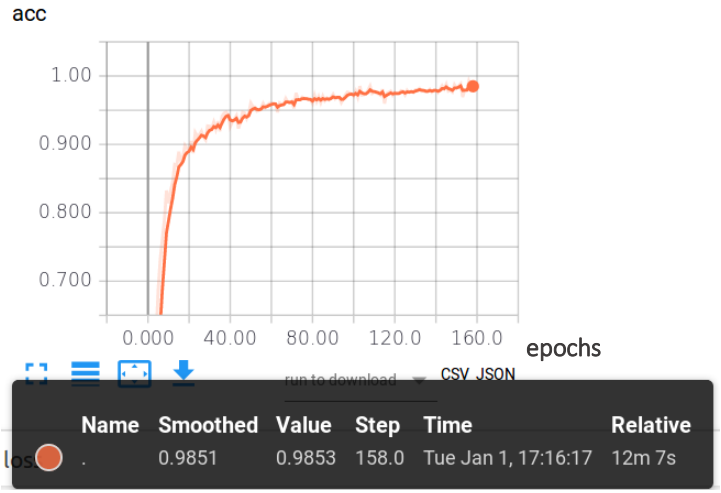
Berdasarkan data uji coba pada Tabel 5.4, nilai akurasi menggunakan *learning rate* 0,001 dan 0,0005 berbeda cukup signifikan yaitu 77% dan 83,6%. Untuk melihat perbandingan nilai akurasi training, *loss* training, akurasi validasi dan *loss* validasi ditunjukan pada grafik yang ditampilkan oleh tensorboard.

Visualisasi tensorboard untuk *optimizer* adam dengan nilai *learning rate* default dan setelah di*tune* dapat dilihat pada Gambar 5.33 sampai dengan Gambar 5.40.

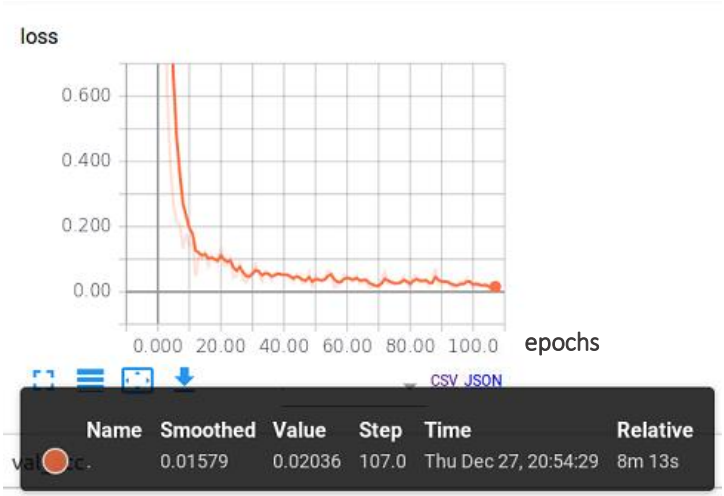
Gambar 5.33 dan Gambar 5.34 merupakan visualisasi dari akurasi masing-masing data latih, Gambar 5.35 dan 5.36 menunjukan visualisasi dari nilai *loss* pada masing-masing data latih, Gambar 5.37 dan Gambar 5.38 menunjukan visualisasi akurasi validasi pada masing-masing data uji, dan Gambar 5.39 dan Gambar 5.40 menunjukan visualiasi dari nilai validasi *loss* pada masing-masing data uji.



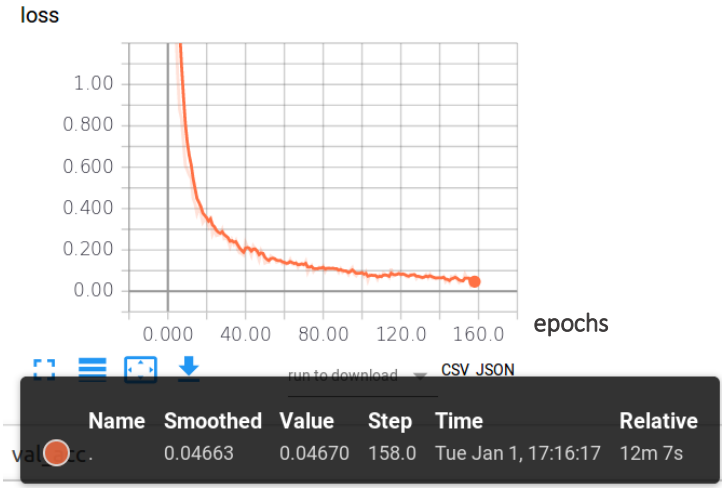
Gambar 5.34 Akurasi data latih dengan nilai *learning rate* 0,001



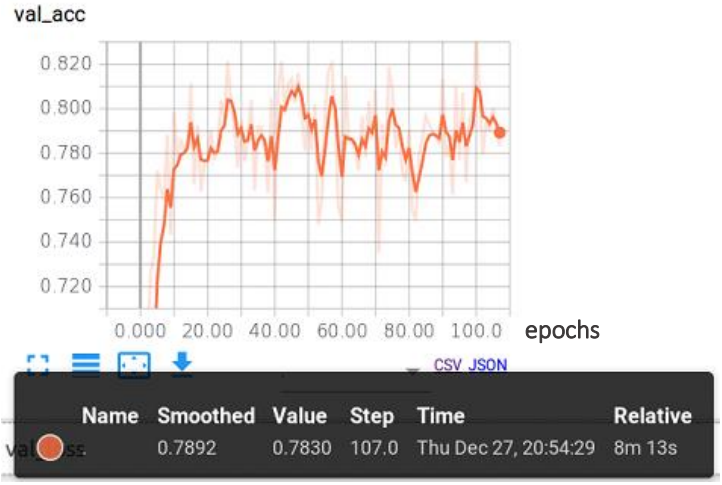
Gambar 5.35 Akurasi data latih dengan nilai *learning rate* 0,0005



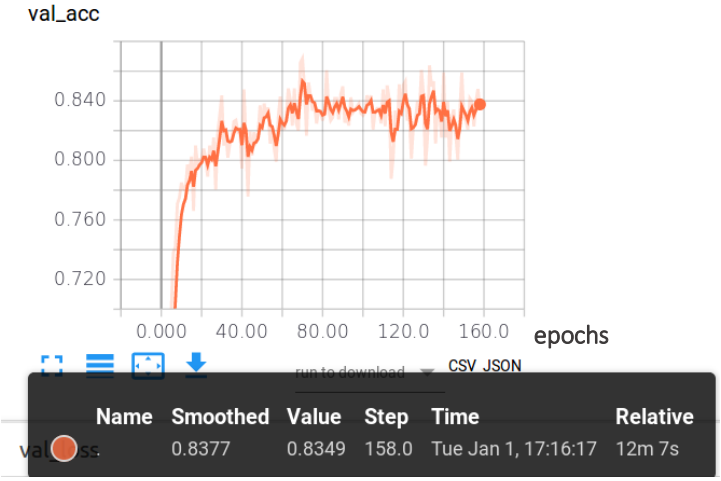
Gambar 5.36 Loss data latih dengan nilai *learning rate* 0,001



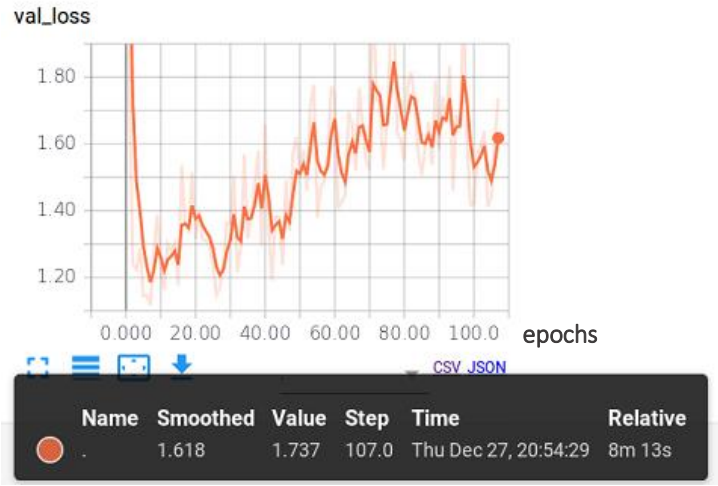
Gambar 5.37 *Loss* data latih dengan nilai *learning rate* 0,0005



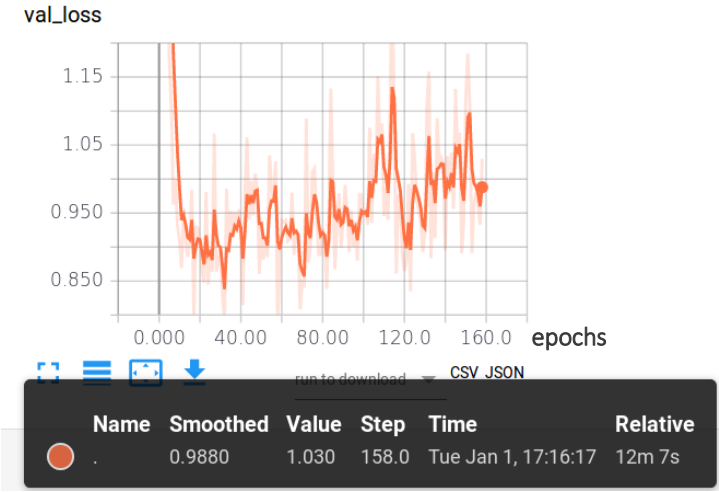
Gambar 5.38 Akurasi validasi data uji dengan nilai *learning rate* 0,001



Gambar 5.39 Akurasi validasi data uji dengan nilai *learning rate* 0,0005



Gambar 5.40 *Loss* data uji dengan nilai *learning rate* 0,001

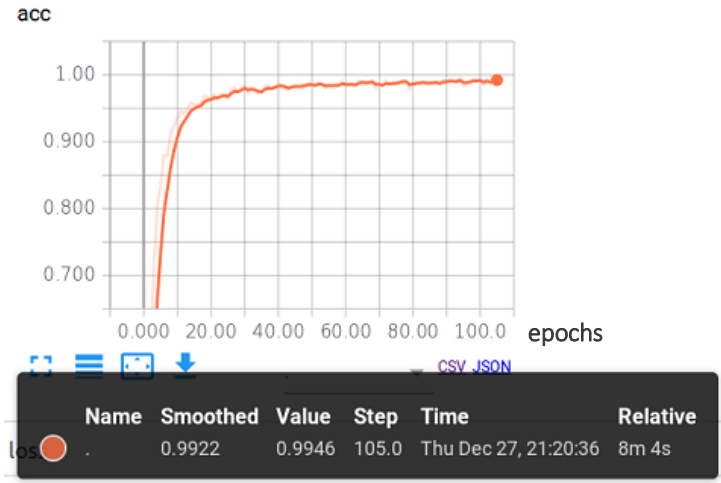


Gambar 5.41 *Loss* data uji dengan nilai *learning rate* 0,0005

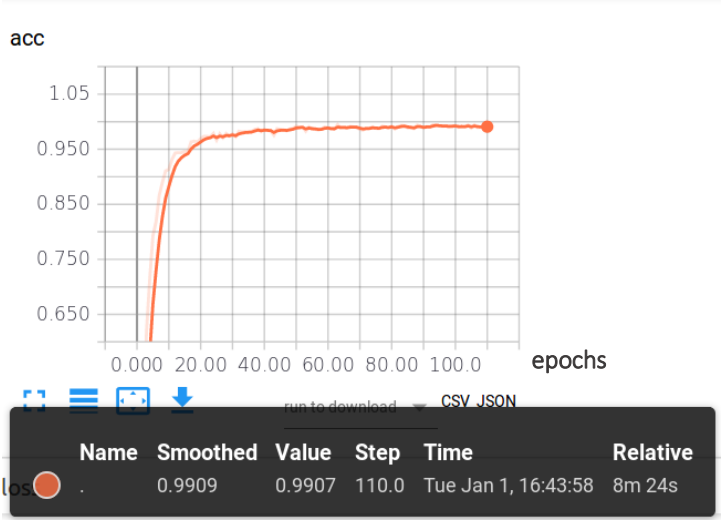
5.4.3.2 Rmsprop Optimizer

Berdasarkan data uji coba pada Tabel 5.4, percobaan menggunakan *optimizer* rmsmprop dengan nilai *learning rate* 0,001 dan 0,0005 memiliki tingkat akurasi yang sama sebesar 83,4%. Namun, nilai *loss* yang dihasilkan oleh *optimizer* rmsprop dengan nilai *learning rate* 0,0005 lebih rendah yaitu 1,53. Untuk melihat perbandingan nilai akurasi training, *loss* training, akurasi validasi dan *loss* validasi ditunjukan pada grafik yang ditampilkan oleh tensorboard.

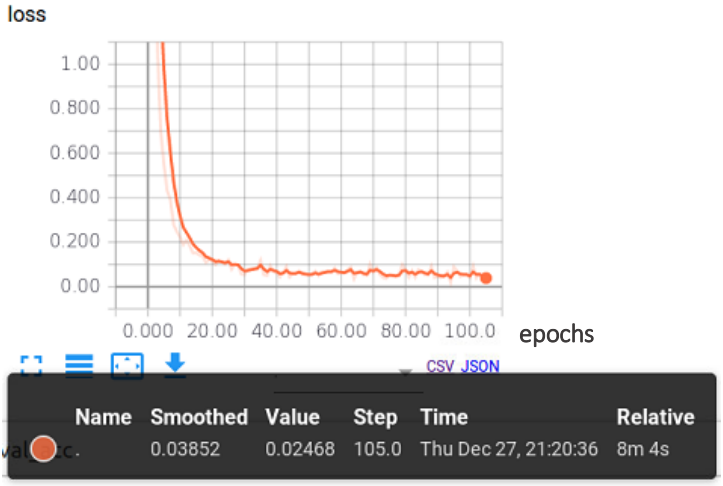
Visualisasi tensorboard untuk *optimizer* adam dengan nilai *learning rate* default dan setelah di*tune* dapat dilihat pada Gambar 5.41 sampai dengan Gambar 5.48.



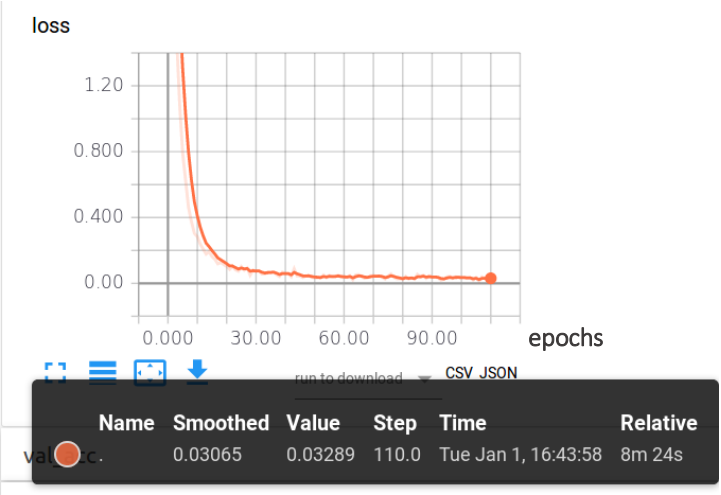
Gambar 5.42 Akurasi data latih dengan nilai *learning rate* 0,001



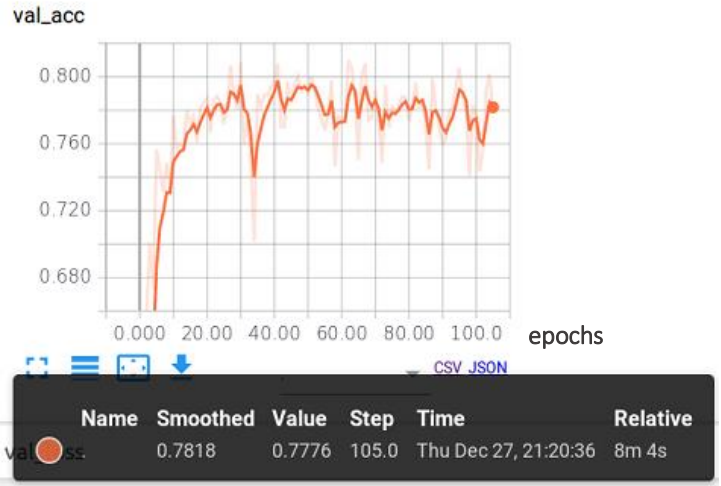
Gambar 5.43 Akurasi data latih dengan nilai *learning rate* 0,0005



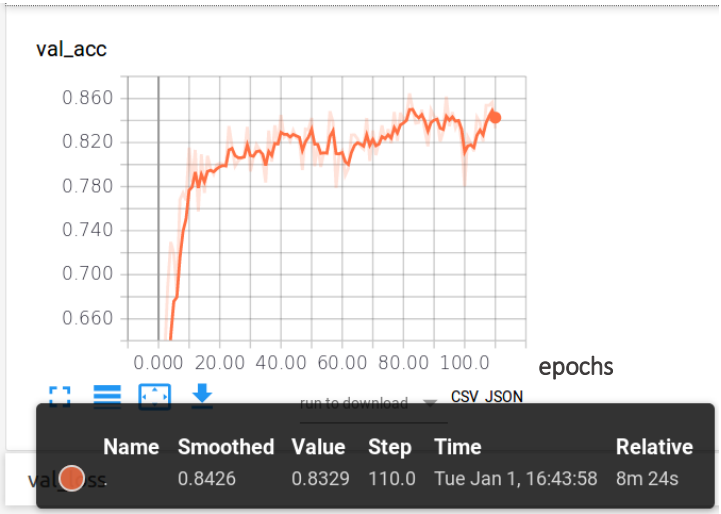
Gambar 5.44 *Loss* data latih dengan nilai *learning rate* 0,001



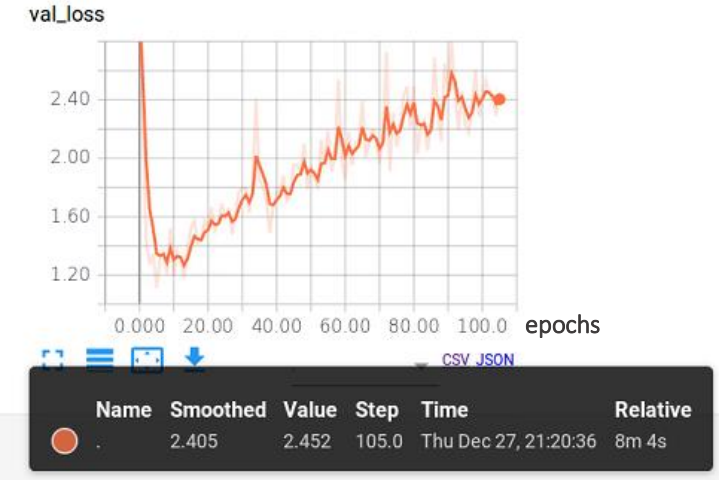
Gambar 5.45 *Loss* data latih dengan nilai *learning rate* 0,0005



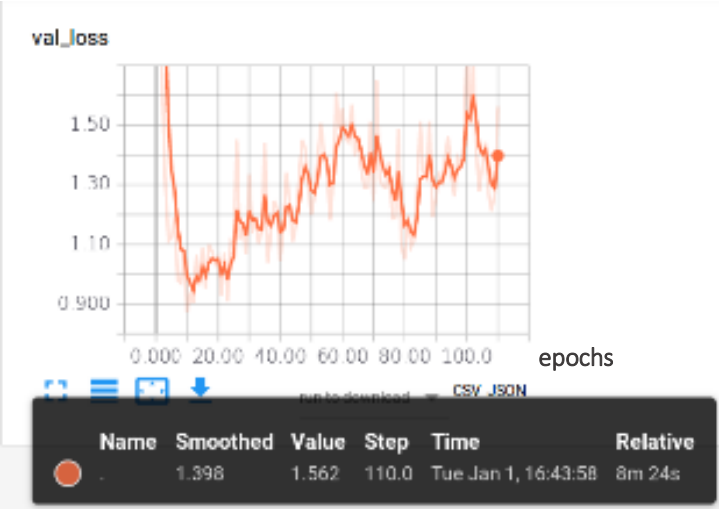
Gambar 5.46 Akurasi validasi data uji dengan nilai *learning rate* 0,001



Gambar 5.47 Akurasi validasi data uji dengan nilai *learning rate* 0,0005



Gambar 5.48 *Loss* data uji dengan nilai *learning rate* 0,001



Gambar 5.49 *Loss* data uji dengan nilai *learning rate* 0,0005

5.5 Evaluasi Kemampuan Prediksi Model

Skenario uji coba yang terdapat pada tugas akhir ini ada dua terhadap masing-masing dataset yang ada. Parameter uji coba yang digunakan pada tugas akhir ini adalah dengan membandingkan performa model menggunakan dua jenis *optimizer* yang berbeda pada setiap dataset.

Matriks konfusi yang digunakan untuk menguju kemampuan model adalah *accuracy, precision, recall* dan *f-measure* sebagaimana sudah dijelaskan pada sub-bab 2.3.

5.5.1 Evaluasi Skenario Uji Coba Dataset 1

Berikut beberapa contoh keluaran dari hasil prediksi model yang salah dari uji coba pada dataset 1 menggunakan model CNN terbaik yaitu *optimizer* Adam dengan *learning rate* 0,001 sesuai dengan hasil uji coba model pada subbab 5.4. Tabel 5.6 menunjukan kemampuan model dalam memprediksi terhadap dataset 1.

Tabel 5.6 Kemampuan prediksi model dataset 1

Optimizer	*Learning rate*	Accuracy (%)	Precision (%)	Recall (%)	F-Measure (%)
Adam	**0,001**	**95,5**	**93,3**	**95,5**	**94,1**
	0,0005	94,0	94,1	94,0	91,5
RMSprop	0,001	90,1	86,2	90,1	87,4
	0,0005	93,6	90,8	90,8	91,7

Gambar 5.50 merupakan hasil prediksi benar yang dikeluarkan oleh model dengan prosentase keyakinan model memprediksi gambar sebesar 99,9%.

Gambar 5.51 memprediksi gambar dengan label 05111740000026 di mana gambar sebenarnya memiliki label 05111540000168 dengan prosentase keyakinan model memprediksi gambar sebesar 43%. Berdasarkan analisa, model salah memprediksi dikarenakan region wajah pada Gambar 5.50 memiliki kemiripan.

```
Name=/home/mis-admin/Documents/pede/Tugas Akhir/DATA-ALL/2017-gasal/test-images/
05111740000079/05111740000079 - NAJATUL MUSLIM DINATRA_9.png Image ID: 276, Label: 05111740000079
Prosentase=0.9999868869781494
```

Gambar 5.50 Contoh prediksi benar pada dataset 1

```
Name=/home/mis-admin/Documents/pede/Tugas Akhir/DATA-ALL/2017-gasal/test-images/
05111440000124/05111440000124 - Aufar Rizqi_9.png Image ID: 165, Label: 05111640000070
Prosentase=0.4346519410610199
```

Gambar 5.51 Contoh prediksi salah pada dataset 1

5.5.2 Evaluasi Skenario Uji Coba Dataset 2

Berikut beberapa contoh keluaran dari hasil prediksi model yang salah dari uji coba pada dataset 2 menggunakan model CNN terbaik yaitu *optimizer* RMSprop dengan nilai *learning rate* 0,0005 yang dapat dilihat pada Tabel 5.7.

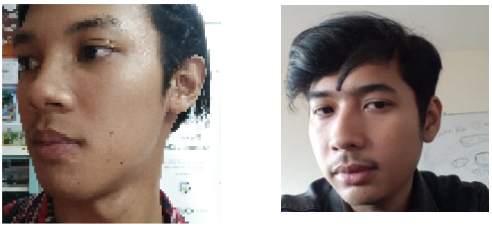
Tabel 5.7 Kemampuan prediksi model dataset 2

Optimizer	*Learning rate*	Accuracy (%)	Precision (%)	Recall (%)	F-Measure (%)
Adam	0,001	91,2	93,2	91,2	91,0
	0,0005	96,3	97,0	96,3	96,3
RMSprop	0,001	96,3	95,8	96,3	96,2
	0,0005	**97,5**	**98,0**	**97,5**	**97,5**

Hasil prediksi pada Gambar 5.52 adalah benar dengan prosentase keyakinan model memprediksi gambar sebesar 100%. Gambar 5.53 memprediksi gambar dengan label 05111540000170 di mana gambar sebenarnya memiliki label 05111740000177 dengan prosentase keyakinan model memprediksi gambar sebesar 53%.

```
Name=/home/mis-admin/Documents/pede/Tugas Akhir/DATA-ALL/2017-genaps/test/
05111640000178/05111640000178-0 .jpg Image ID: 49, Label: 05111640000178 Prosentase=1.0
```

Gambar 5.52 Contoh prediksi benar pada dataset 2



```
Name=/home/mis-admin/Documents/pede/Tugas Akhir/DATA-ALL/2017-genaps/test/
05111740000177/05111740000177-10.jpg Image ID: 23, Label: 05111540000170 Prosentase=0.5320029258728027
```

Gambar 5.53 Contoh prediksi salah pada dataset 2

Berdasarkan analisa, model salah memprediksi dikarenakan region mata pada Gambar 5.53 memiliki kemiripan.

5.5.3 Evaluasi Skenario Uji Coba Dataset 3

Tabel 5.8 merupakan hasil prediksi model pada dataset 3. Seperti yang sudah dijelaskan pada subbab 5.4. Berikut beberapa contoh keluaran dari hasil prediksi model yang salah dari uji coba pada dataset 3.

Tabel 5.8 Kemampuan prediksi model dataset 3

Optimizer	*Learning rate*	Accuracy (%)	Precision (%)	Recall (%)	F-Measure (%)
Adam	0,001	77,4	78,1	77,4	75,0
	0,0005	**85,3**	**87,8**	**85,3**	**84,5**
RMSprop	0,001	75,1	74,3	75,1	72,5
	0,0005	71,6	72,2	71,6	69,6

Gambar 5.54 adalah hasil prediksi model yang benar dengan gambar sebenarnya, yaitu gambar dengan label 05111540000150. Model berhasil mengenali gambar dengan prosentasi keyakinan sebesar 99,5%.

Gambar 5.55 memprediksi gambar dengan label 05111540000083 di mana gambar sebenarnya memiliki label 05111740000063 dengan prosentase keyakinan model memprediksi gambar sebesar 65,3%. Berdasarkan analisa, model salah memprediksi dikarenakan terdapat kemiripan di beberapa region seperti mata, tulang pipi dan rahang pada Gambar 5.55.

```
Name=/home/mis-admin/Documents/pede/Tugas Akhir/DATA-ALL/2018-gasal/test/
05111740000150/19_20181204110053.png Image ID: 62, Label: 05111740000150
Prosentase=0.9999990463256836
```

Gambar 5.54 Contoh prediksi benar pada dataset 3

```
Name=/home/mis-admin/Documents/pede/Tugas Akhir/DATA-ALL/2018-gasal/test/
05111740000063/19_20181204105445.png Image ID: 25, Label: 05111540000083
Prosentase=0.6539657711982727
```

Gambar 5.55 Contoh prediksi salah pada dataset 3

5.5.4 Hasil Analisa Keberhasilan CNN

Subbab ini akan menjelaskan bagaimana keberhasilan CNN terhadap uji coba yang telah dilakukan. Pada uji coba yang terdapat di subbab 5.5.1, model terbaik didapatkan pada *optimizer* Adam dengan *learning rate* 0,001. Sementara pada uji coba di subbab 5.5.2 model terbaik didapatkan pada *optimizer* RMSprop dengan *learning rate* 0,0005. Uji coba pada subbab 5.5.3 model terbaik didapatkan pada *optimizer* Adam dengan *learning rate* 0,0005. Hasil analisa keberhasilan CNN dapat dilihat pada Tabel 5.11.

Untuk mengetahui keberhasilan CNN pada setiap dataset yang dilakukan uji coba, perlu dilakukan analisa pada model SVM dan LDA terhadap masing-masing dataset yang digunakan sebagai pembanding. Hasil analisa pada LDA dan SVM dapat dilihat pada Tabel 5.9 dan Tabel 5.10

Berdasarkan Tabel 5.9, Tabel 5.10, dan Tabel 5.11 pada dataset 1, akurasi CNN paling tinggi dibanding dengan LDA dan SVM, sementara dalam waktu memprediksi model LDA lebih unggul dibandingkan SVM dan CNN. Begitu pun pada ukuran

ruang model. Untuk dataset 1 keberhasilan CNN tidak lebih baik dibanding dengan LDA.

Hasil analisa pada dataset 2 berdasarkan Tabel 5.9, Tabel 5.10, dan Tabel 5.11 akurasi model terbaik diperoleh dengan menggunakan SVM yaitu sebesar 98,0% sedangkan CNN memiliki akurasi 97,8% dan LDA sebesar 96,4%. Untuk waktu prediksi paling cepat diperoleh menggunakan LDA yaitu sebesar 0,04 detik kemudian disusul CNN dengan waktu prediksi selama 0,73 detik dan yang paling lama adalah SVM yaitu sebesar 3,84 detik. Untuk ukuran ruang model CNN memiliki ukuran paling kecil diantara SVM dan LDA. Hasil analisa pada dataset 2 menunjukan bahwa CNN lebih baik dibandingkan dengan LDA dan SVM.

Hasil analisa pada dataset 3 berdasarkan Tabel 5.9, Tabel 5.10, dan Tabel 5.11 menunjukan akurasi terbaik yaitu sebesar 85,2% dibanding LDA hanya sebesar 67,6% dan SVM sebesar 82,8%. Waktu prediksi model terbaik didapat dengan LDA sebesar 0,0208304 *seconds* kemudian CNN sebesar 0,69 detik dan SVM sebesar 2,29 detik. Sedangkan perbandingan ukuran ruang pada model, model menggunakan LDA memiliki ukuran yang lebih kecil yaitu 11,7 MB kemudian CNN sebesar 13,2 MB dan SVM sebesar 17,5 MB. Secara keseluruhan, analisa keberhasilan CNN pada dataset 3 lebih baik dibandingkan dengan LDA dan SVM.

Tabel 5.9 Hasil analisa LDA terhadap model terbaik

Dataset	Akurasi (%)	Waktu Prediksi (detik)	Ukuran Ruang Model (MB)
1	93,4	0,50	70,1
2	96,4	0,04	13,3
3	67,6	0,02	11,7

Tabel 5.10 Hasil analisa SVM terhadap model terbaik

Dataset	Akurasi (%)	Waktu Prediksi (detik)	Ukuran Ruang Model (MB)
1	88,5	15,32	73,2
2	98,0	3,84	20,5
3	82,8	2,29	17,5

Tabel 5.11 Hasil analisa CNN terhadap model terbaik

Dataset	Akurasi (%)	Waktu Prediksi (detik)	Ukuran Ruang Model (MB)
1	95,5	1,78	83
2	97,8	0,73	1,8
3	85,2	0,69	13,2

5.6 Analisa Uji Coba Dengan Aksesoris

Berikut ini akan dijelaskan beberapa hasil analisa pada gambar wajah yang terdapat aksesoris seperti kacamata dan topi.

Hasil analisa pada beberapa gambar wajah yang terdapat pada model tanpa menggunakan aksesoris model gagal mengenali wajah yang diprediksi. Berikut ini salah satu contoh data yang dilakukan prediksi menggunakan gambar dengan aksesoris kacamata, di mana pada saat proses *training* dilakukan gambar wajah tidak menggunakan atribut tersebut.

Gambar 5.56 Hasil prediksi model pada dataset 1 dengan aksesoris kacamata

menunjukan bahwa model gagal mengenali wajah yang diberikan kepada model pada skenario dataset 1. Model memprediksi label 05111440000044, sementara label sebenarnya adalah 05111540000135. Dalam hal ini, dataset pada saat proses *training*

dilakukan tidak menggunakan atribut kacamata seperti pada gambar wajah yang dilakukan sebagai data uji model.

Gambar 5.57 menunjukan bahwa model gagal mengenali wajah yang diberikan kepada model pada skenario dataset 3. Model memprediksi label 05111540000048, sementara label sebenarnya adalah 05111540000135. Dalam hal ini, dataset pada saat proses *training* dilakukan tidak menggunakan atribut kacamata seperti pada gambar wajah yang dilakukan sebagai data uji model.

Dari hasil analisa di atas, dapat disimpulkan bahwa aksesoris memengaruhi model dalam pengenalan wajah. Agar kemampuan model lebih baik dalam mengenali wajah dengan aksesoris, penulis memiliki beberapa saran untuk pengembangan model ke depannya yaitu pengambilan dataset wajah setiap mahasiswa dengan menggunakan dan tanpa menggunakan aksesoris (dalam hal ini bisa berupa kacamata dan topi) atau melakukan pra-prosesing pada dataset wajah untuk menghilangkan aksesoris pada wajah agar gambar wajah bebas dari aksesoris yang ada.

```
Name=/home/mis-admin/Documents/pede/Tugas Akhir/DATA-ALL/data test/testpd.jpg Image ID:
17, Label: 05111440000044 Prosentase=0.7283658981323242
Groundtruth:  [17]
Predicted:  ['05111440000044']
```

Gambar 5.56 Hasil prediksi model pada dataset 1 dengan aksesoris kacamata

```
Name=/home/mis-admin/Documents/pede/Tugas Akhir/DATA-ALL/data test/testpd.jpg Image ID:
14, Label: 05111540000048 Prosentase=0.9999991655349731
Groundtruth:  [14]
Predicted:  ['05111540000048']
```

Gambar 5.57 Hasil prediksi model pada dataset 3 dengan aksesoris kacamata

BAB VI
KESIMPULAN DAN SARAN

Pada bab ini akan dijelaskan kesimpulan dan saran pengembagan dari tugas akhir ini.

6.1 Kesimpulan

Dari berbagai uji coba yang telah dilakukan, terdapat 3 poin kesimpulan dari tugas akhir ini, yaitu:

1. Metode CNN berhasil diimplementasikan pada dataset Sistem Kehadiran Mahasiswa (SIKEMAS) dan meningkatkan akurasi dari metode LDA sebanyak 2,2% namun tidak untuk waktu prediksi dan ukuran ruang model, sementara akurasi dan waktu prediksi menggunakan CNN lebih baik dibandingkan dengan menggunakan SVM pada dataset SIKEMAS yaitu akurasi meningkat sebanyak 7% dan waktu prediksi model lebih cepat 13,54 detik, tetapi untuk ukuran ruang pada model tidak lebih baik dari SVM.
2. Hasil komputasi pada CNN berbeda pada setiap dataset yang digunakan.
   a. Untuk dataset 1 model terbaik didapatkan menggunakan *optimizer* Adam dengan nilai *learning rate* 0,0005 memiliki akurasi model sebesar 95,5% dan *loss* sebesar 0,239 dengan waktu prediksi model sebesar 1,78 detik dan ukuran ruang model sebesar 83 MB.
   b. Untuk dataset 2 model terbaik didapatkan menggunakan rmsprop dengan nilai *learning rate* 0,0005 memiliki akurasi model sebesar 97,8% dan *loss* sebesar 0,101 dengan waktu prediksi model sebesar 0,73 detik dan ukuran ruang model sebesar 1,8 MB.
   c. Untuk dataset 3 model terbaik didapatkan menggunakan *optimizer* Adam dengan nilai *learning rate* 0,0005 memiliki akurasi model sebesar 84,9% dan *loss* sebesar 1,18 dengan waktu prediksi model sebesar 0,69 detik dan ukuran ruang model sebesar 13,2 MB.

3. Dataset terbaik dalam pembuatan model dengan CNN adalah dataset 2 yang memiliki 4 variasi sudut pengambilan dengan akurasi model menggunakan *optimizer* rmsprop dengan *learning rate* 0,0005 sebesar 97,8% dan *loss* 0,101.

6.2 Saran

Berikut ini adalah saran untuk pengembangan penelitian selanjutnya.

1. Menggunakan dataset yang lebih banyak dan lebih variatif untuk data latih.
2. Mencoba menganalisa dengan parameter uji coba yang lebih banyak.
3. Melakukan *deployment* model pada aplikasi SIKEMAS.
4. Menambahkan pra-prosesing pada pelatihan dataset seperti menghilangkan aksesoris yang dipakai pada wajah.

DAFTAR PUSTAKA

[1] J. S. R. A. Dwi Sunaryono, "An Android Based Course Attendance System Using Face Recognition," *Journal of King Saud University - Computer Science and Information Sciences,* pp. 31-38 , 2018.

[2] A. Karpathy, "Convolutional Neural Networks for Visual Recognition," Stanford, 2018. [Online]. Available: http://cs231n.github.io/convolutional-networks/. [Diakses 4 December 2018].

[3] S. Chatterjee, "saama," 20 December 2017. [Online]. Available: https://www.saama.com/blog/different-kinds-convolutional-filters/. [Diakses 4 December 2018].

[4] R. Dharmadi, "Medium Nodeflux," Medium, 4 April 2018. [Online]. Available: https://medium.com/nodeflux/mengenal-convolutional-layer-dan-pooling-layer-3c6f5c393ab2. [Diakses 4 December 2018].

[5] H. P. J. E. J. M. Hugo Mayo, "AI in Radiology," 2018. [Online]. Available: https://www.doc.ic.ac.uk/~jce317/introduction-cnns.html. [Diakses 8 January 2019].

[6] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever dan R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research,* no. 15, pp. 1929-1958, 2014.

[7] S. Sena, “https://medium.com/@samuelsena,” Medium, 13 November 2017. [Online]. Available: https://medium.com/@samuelsena/pengenalan-deep-learning-part-7-convolutional-neural-network-cnn-b003b477dc94. [Diakses 6 May 2018].

[8] S. Ruder, “An overview of gradient descent optimization,” *Arxiv,* pp. 1-14, 2017.

[9] D. P. Kingma dan D. P. Kingma, “Adam : A Method for Stochastic Optimization,” *ICLR*, pp. 1-15, 2015.

[10] M. Abadi, P. Barham, J. Chen dan Z. Chen, “TensorFlow: A system for large-scale machine learning,” *USENIX,* no. 12, pp. 265-283, 2016.

[11] Google Brain Team, “Visualisation with TensorBoard,” Google AI, [Online]. Available: https://www.learningtensorflow.com/Visualisation/. [Diakses 27 12 2018].

[12] M. N. J. R dan B. K, “Performance Analysis of Neural Networks and Support Vector Machine using Confusion Matrix,” *IJARSET,* vol. III, no. 5, pp. 2106-2109, 2016.

[13] A. Ragan, “Taking the Confusion Out of Confusion Matrices,” 11 10 2018. [Online]. Available: https://towardsdatascience.com/taking-the-confusion-out-of-confusion-matrices-c1ce054b3d3e. [Diakses 27 12 2018].

[14] T. Guo, J. Dong, H. Li dan Y. Gao, "Simple Convolutional Neural Network on Image Classification," *IEEE,* pp. 721-724, 2017.

[15] "Definition Dataset," TechTarget, March 2016. [Online]. Available: https://whatis.techtarget.com/definition/data-set. [Diakses 7 May 2018].

[16] H. Gao, "A Walk-through of AlexNet," Medium, 8 8 2017. [Online]. Available: https://medium.com/@smallfishbigsea/a-walk-through-of-alexnet-6cbd137a5637. [Diakses 8 December 2018].

[17] A. Krizhevsky, I. Sutskever dan G. E. Hinton, "ImageNet Classification with Deep Convolutional," *Neural Information Processing Systems,* pp. 1-9, 2012.

[18] R. G. Radityatama, Rancang Bangun Aplikasi Mobile Android Sistem Kehadiran Mahasiswa Melalui Pencocokan Wajah dengan Menggunakan Library Android Face Recognition with Deep Learning Studi Kasus Jurusan Teknik Informatika ITS, Surabaya: ITS Press, 2017.

[19] A. Gozzoli, "Checkpointing Tutorial for TensorFlow, Keras, and PyTorch," floydhub, 21 11 2017. [Online]. Available: https://blog.floydhub.com/checkpointing-tutorial-for-tensorflow-keras-and-pytorch/. [Diakses 21 12 2018].

[20] "https://deeplearning4j.org/," [Online]. Available: https://deeplearning4j.org/convolutionalnetwork. [Diakses 6 May 2018].

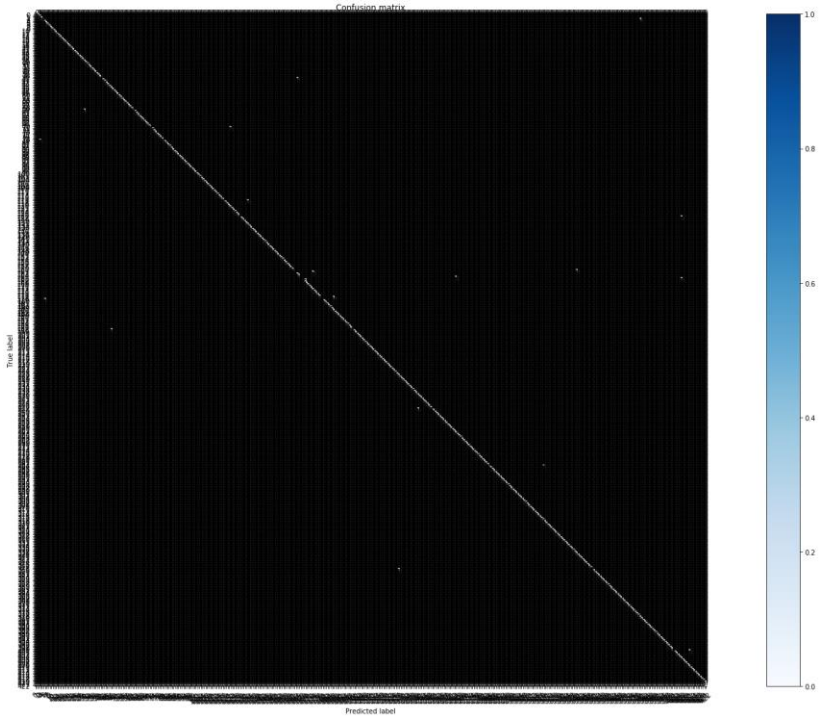
[Halaman ini sengaja dikosongkan]

LAMPIRAN

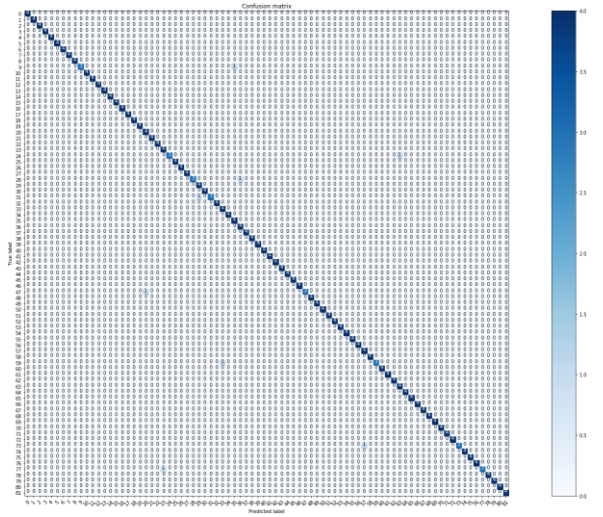
Lampiran 1

Confusion matrix pada setiap skenario terbaik dari setiap dataset.

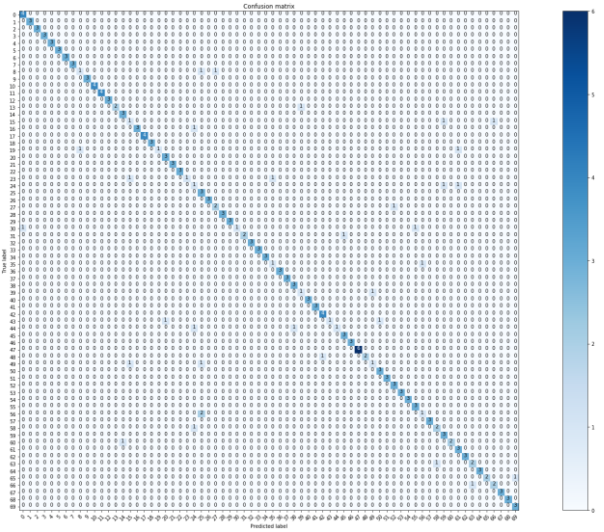
1. Hasil *confusion matrix* pada dataset 1 yang memiliki jumlah kelas sebanyak 43



2. Hasil *confusion matrix* pada dataset 2 yang memiliki jumlah kelas sebanyak 82 kelas.



3. Hasil *confusion matrix* pada dataset 3 yang memiliki jumlah kelas seabanyak 70 kelas.

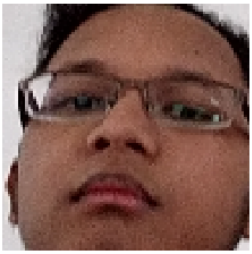


Lampiran 2

Beberapa hasil prediksi benar dan dari setiap parameter *optimizer* yang digunakan untuk setiap dataset.

1. Hasil prediksi benar dan salah pada dataset 1 menggunakan *optimizer* Adam (*learning rate* = 0,0005)

```
Name=/home/mis-admin/Documents/pede/Tugas Akhir/DATA-ALL/2017-gasal/test-images/
05111740000079/05111740000079 - NAJATUL MUSLIM DINATRA_9.png Image ID: 276, Label: 05111740000079
Prosentase=0.9999868869781494
```



```
Name=/home/mis-admin/Documents/pede/Tugas Akhir/DATA-ALL/2017-gasal/test-images/
05111540000168/05111540000168 - Rogo Jagad Alit_9.png Image ID: 226, Label: 05111740000026
Prosentase=0.4348115026950836
```

2. Hasil prediksi benar dan salah pada dataset 1 menggunakan *optimizer* RMSprop (*learning rate* = 0,001)

```
Name=/home/mis-admin/Documents/pede/Tugas Akhir/DATA-ALL/2017-gasal/test-images/
05111740000079/05111740000079 - NAJATUL MUSLIM DINATRA_9.png Image ID: 276, Label: 05111740000079
Prosentase=0.9950073957443237
```

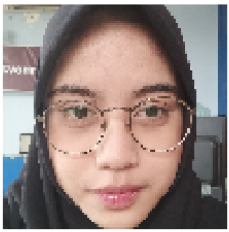
```
Name=/home/mis-admin/Documents/pede/Tugas Akhir/DATA-ALL/2017-gasal/test-images/
05111740000148/05111740000148 - BAGAS AGENG PRASOJO_9.png Image ID: 341, Label: 05111740000147
Prosentase=0.5711209774017334
```

3. Hasil prediksi benar dan salah pada dataset 1 menggunakan *optimizer* RMSprop (*learning rate* = 0,0005)

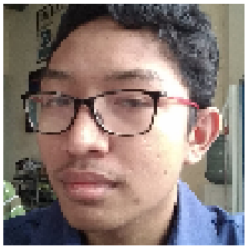
```
Name=/home/mis-admin/Documents/pede/Tugas Akhir/DATA-ALL/2017-gasal/test-images/
05111740000079/05111740000079 - NAJATUL MUSLIM DINATRA_9.png Image ID: 276, Label: 05111740000079
Prosentase=0.9818630814552307
```

Name=/home/mis-admin/Documents/pede/Tugas Akhir/DATA-ALL/2017-gasal/test-images/05111740000148/05111740000148 - BAGAS AGENG PRASOJO_9.png Image ID: 382, Label: 05111740000188 Prosentase=0.41888418793678284

4. Hasil prediksi benar dan salah pada dataset 2 menggunakan *optimizer* Adam (*learning rate* = 0,001)

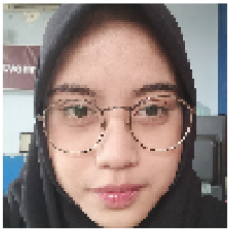


Name=/home/mis-admin/Documents/pede/Tugas Akhir/DATA-ALL/2017-genaps/test/05111540000001/05111540000001-0.jpg Image ID: 8, Label: 05111540000001 Prosentase=0.9930371642112732

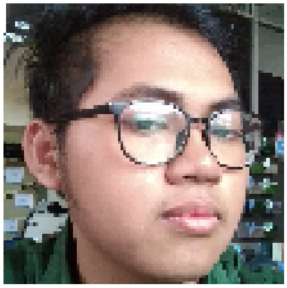


Name=/home/mis-admin/Documents/pede/Tugas Akhir/DATA-ALL/2017-genaps/test/05111640000056_/05111640000056_-10.jpg Image ID: 47, Label: 05111640000168 Prosentase=0.4653509557247162

5. Hasil prediksi benar dan salah pada dataset 2 menggunakan *optimizer* Adam (*learning rate* = 0,0005)

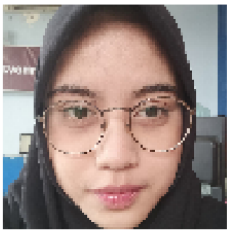


```
Name=/home/mis-admin/Documents/pede/Tugas Akhir/DATA-ALL/2017-genaps/test/
05111540000001/05111540000001-0.jpg Image ID: 8, Label: 05111540000001 Prosentase=0.9969998002052307
```

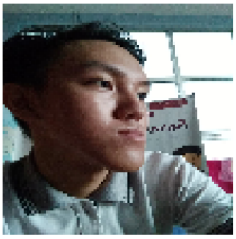


```
Name=/home/mis-admin/Documents/pede/Tugas Akhir/DATA-ALL/2017-genaps/test/
05111640000168/05111640000168-19.jpg Image ID: 27, Label: 05111640000056_
Prosentase=0.8086556792259216
```

6. Hasil prediksi benar dan salah pada dataset 2 menggunakan *optimizer* RMSprop (*learning rate* = 0,001)



```
Name=/home/mis-admin/Documents/pede/Tugas Akhir/DATA-ALL/2017-genaps/test/
05111540000001/05111540000001-0.jpg Image ID: 8, Label: 05111540000001 Prosentase=0.9997816681861877
```



```
Name=/home/mis-admin/Documents/pede/Tugas Akhir/DATA-ALL/2017-genaps/test/
05111740000062/05111740000062-3 Image ID: 48, Label: 05111640000176 Prosentase=0.9167464971542358
```

7. Hasil prediksi benar dan salah pada dataset 3 menggunakan *optimizer* Adam (*learning rate* = 0,001)

```
Name=/home/mis-admin/Documents/pede/Tugas Akhir/DATA-ALL/2018-gasal/test/
05111740000150/19_20181204110053.png Image ID: 62, Label: 05111740000150
Prosentase=0.9992244243621826
```

Name=/home/mis-admin/Documents/pede/Tugas Akhir/DATA-ALL/2018-gasal/test/05111740000184/20_20181204104732.png Image ID: 55, Label: 05111740000060 Prosentase=0.33162859082221985

8. Hasil prediksi benar dan salah pada dataset 3 menggunakan *optimizer* RMSprop (*learning rate* = 0,001)

Name=/home/mis-admin/Documents/pede/Tugas Akhir/DATA-ALL/2018-gasal/test/05111740000150/19_20181204110053.png Image ID: 62, Label: 05111740000150 Prosentase=0.9949847459793091

Name=/home/mis-admin/Documents/pede/Tugas Akhir/DATA-ALL/2018-gasal/test/05111740000184/20_20181204104732.png Image ID: 24, Label: 05111540000079 Prosentase=0.36500951647758484

9. Hasil prediksi benar dan salah pada dataset 3 menggunakan *optimizer* RMSprop (*learning rate* = 0,0005)

```
Name=/home/mis-admin/Documents/pede/Tugas Akhir/DATA-ALL/2018-gasal/test/
05111740000150/19_20181204110053.png Image ID: 62, Label: 05111740000150 Prosentase=0.9949847459793091
```

```
Name=/home/mis-admin/Documents/pede/Tugas Akhir/DATA-ALL/2018-gasal/test/
05111540000076/7_20181206144540.png Image ID: 56, Label: 05111740000063 Prosentase=0.493050754070282
```

[Halaman ini sengaja dikosongkan]

BIODATA PENULIS

Anisah Putri Diana
lahir di Tangerang, pada 8 Juni 1997. Penulis merupakan mahasiswa Departemen Informatika ITS angkatan 2015 dan merupakan administrator Laboratorium Mobile Innovation Studio (MIS) Informatika ITS. Berpengalaman menjadi asisten dosen pada mata kuliah Sistem Digital, Otomata, Komputasi Numerik dan Kecerdasan Komputasional. Penulis juga tercatat sebagai asisten dosen pada Pendidikan Informatika dan Komputer Terapan (PIKTI) ITS 2018. Selama menjalani masa perkuliahan, penulis aktif dalam berbagai kegiatan kemahasiswaan. Hal ini dibuktikan dengan pengalaman penulis sebagai panitia di beberapa acara kampus seperti Koordinator Dana dan Usaha dalam acara Schematics 2016 dan 2017. Tidak hanya aktif dalam kepanitiaan, pada tahun kedua perkuliahan penulis juga aktif dalam organisasi kemahasiswaan yang ada di ITS. Penulis pernah menjadi staf Media Informasi di Himpunan Mahasiswa Teknik Computer-Informatika ITS dan staf *External Affairs* BEM Fakultas Teknologi dan Informasi (FTIf) ITS.

Dalam menyelesaikan pendidikan sarjana penulis mengambil bidang minat Komputasi Cerdas dan Visi (KCV) yang merupakan ketertarikan penulis dalam Visi Komputer khususnya dan Kecerdasan Buatan umumnya.