TUGAS AKHIR – KI141502

KLASIFIKASI CITRA MENGGUNAKAN CONVOLUTIONAL NEURAL NETWORK (CNN) PADA CALTECH 101

I WAYAN SUARTIKA EKA PUTRA
NRP 5109100096

Dosen Pembimbing I
Arya Yudhi Wijaya, S.Kom., M.Kom.

Dosen Pembimbing II
Rully Soelaiman, S.Kom., M.Kom.

Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2016

UNDERGRADUATE THESES – KI141502

IMAGE CLASSFICATION USING CONVOLUTION NEURAL NETWORK (CNN) ON CALTECH 101

I WAYAN SUARTIKA EKA PUTRA
NRP 5109100096

First Advisor
Arya Yudhi Wijaya, S.Kom., M.Kom.

Second Advisor
Rully Soelaiman, S.Kom., M.Kom.

Department of Informatics
Faculty of Information Technology
Sepuluh Nopember Institute of Technology
Surabaya 2016

KATA PENGANTAR

Puji Tuhan, puji syukur bagi Hyang Widhi yang telah menyertai penulis, sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul:

Klasifikasi Citra Menggunakan Convolutional Neural Network (CNN) pada Caltech 101

Selesainya Tugas Akhir ini tidak lepas dari bantuan dan dukungan beberapa pihak. Sehingga pada kesempatan ini penulis mengucapkan terima kasih kepada :

1. Tuhan Yang Maha Esa atas penyertaan dari awal hingga akhir masa perkuliahan sehingga penulis dapat menyelesaikan Tugas Akhir dengan baik.
2. Bapak Rully Soelaiman, S.Kom., M.Kom., selaku pembimbing II yang telah membantu dan membimbing penulis dalam menyelesaikan Tugas Akhir ini dengan sabar.
3. Bapak Arya Yudhi Wijaya, S.Kom., M.Kom., selaku pembimbing I yang telah memberikan nasihat, arahan, dan bantuan sehingga penulis dapat menyelesaikan Tugas Akhir ini.
4. Segenap dosen dan karyawan Teknik Informatika ITS atas ilmu dan pengalaman yang telah diberikan selama penulis menjalani masa studi di Teknik Informatika ITS.

Penulis menyadari bahwa Tugas Akhir ini masih memiliki banyak kekurangan. Sehingga dengan kerendahan hati, penulis mengharapkan kritik dan saran dari pembaca untuk perbaikan ke depan.

Surabaya, Januari 2016
I Wayan Suartika Eka Putra

LEMBAR PENGESAHAN

KLASIFIKASI CITRA MENGGUNAKAN CONVOLUTIONAL NEURAL NETWORK (CNN) PADA CALTECH 101

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Komputasi Cerdas Visual
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh
I WAYAN SUARTIKA EKA PUTRA
NRP : 5109 100 096

Disetujui oleh Dosen Pembimbing Tugas Akhir:

Arya Yudhi Wijaya, S.Kom., M.Kom. .....
NIP: 197509142001122002 **(Pembimbing 1)**

Rully Soelaiman, S.Kom., M.Kom. .....
NIP: 197002131994021001 **(Pembimbing 2)**

SURABAYA
Januari, 2016

KLASIFIKASI CITRA MENGGUNAKAN CONVOLUTIONAL NEURAL NETWORK (CNN) PADA CALTECH 101

Nama Mahasiswa : I WAYAN SUARTIKA EKA PUTRA
NRP : 5109100096
Jurusan : Teknik Informatika FTIF-ITS
Dosen Pembimbing 1 : Arya Yudhi Wijaya, S.Kom., M.Kom.
Dosen Pembimbing 2 : Rully Soelaiman, S.Kom., M.Kom.

Abstrak

Deep Learning adalah sebuah bidang keilmuan baru dalam bidang Machine Learning yang akhir-akhir ini berkembang karena perkembangan teknologi GPU accelaration. Deep Learning memiliki kemampuan yang sangat baik dalam visi komputer. Salah satunya adalah pada kasus klasifikasi objek pada citra.

Tugas akhir ini mengimplementasikan salah satu metode machine learning yang dapat digunakan untuk klasifikasi citra objek yaitu CNN. Metode CNN terdiri dari dua tahap. Tahap pertama adalah klasifikasi citra menggunakan feedforward. Tahap kedua merupakan tahap pembelajaran dengan metode backpropagation. Sebelum dilakukan klasifikasi, terlebih dahulu dilakukan praproses dengan metode wrapping dan cropping untuk memfokuskan objek yang akan diklasifikasi. Selanjutnya dilakukan training menggunakan metode feedforward dan backpropagation. Terakhir adalah tahap klasifikasi menggunakan metode feedforward dengan bobot dan bias yang diperbarui.

Hasil uji coba dari klasifikasi citra objek dengan tingkat confusion yang berbeda pada basis data Caltech 101 menghasilkan rata-rata nilai akurasi mencapai. Sehingga dapat disimpulkan bahwa metode CNN yang digunakan pada Tugas Akhir ini mampu melakukan klasifikasi dengan baik.

Kata Kunci: Deep learning, Convolution Neural Network, Caltech 101.

IMAGE CLASSFICATION USING CONVOLUTION NEURAL NETWORK (CNN) ON CALTECH 101

Student's Name	**: I WAYAN SUARTIKA EKA PUTRA**
Student's ID	**: 5109100096**
Department	**: Teknik Informatika FTIF-ITS**
First Advisor	**: Arya Yudhi Wijaya, S.Kom., M.Kom.**
Second Advisor	**: Rully Soelaiman, S.Kom., M.Kom.**

Abstract

Deep Learning is a new branch of knowledge in the field of Machine Learning that in the past few years has developed due to the improvement in GPU Acceleration technologies. Deep Learning has great capabilities in the field of computer vision. One of its capabilities is in the case of object classification in images.

This final project implements one of the methods machine learning that can be used for image classification object that is CNN. CNN method consists of two stages. The first stage is to use feedforward for image classification. The second stage is about of learning to update weight and biases by back propagation method. Before the classification, first performed preprocessing with wrapping method and cropping to focusing the object in image to be classified. Furthermore, the training method used feedforward and backpropagation. The last stage is the classification using feedforward method with weights and biases are updated.

The results of image classification objects with different levels of confusion at Caltech 101 database generates an average value of accuracy reached 100%. So it can be concluded that the CNN method used on Final project able to perform classification going well.

Keywords: Deep learning, Convolution Neural Network, Caltech 101

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

Pada bagian ini akan dijelaskan beberapa hal dasar mengenai Tugas Akhir ini yang meliputi: latar belakang, tujuan, manfaat permasalahan, batasan permasalahan, metodologi serta sistematika penulisan Tugas Akhir. Penjelasan tentang hal-hal tersebut diharapkan dapat memberikan gambaran umum mengenai permasalahan sehingga penyelesaian masalah dapat dipahami dengan baik.

1.1 Latar Belakang Masalah

Salah satu *problem* dalam visi komputer yang telah lama dicari solusinya adalah klasifikasi objek pada citra secara umum. Bagaimana menduplikasi kemampuan manusia dalam memahami informasi citra, agar komputer dapat mengenali objek pada citra selayaknya manusia. Proses *feature engineering* yang digunakan pada ummumnya sangat terbatas dimana hanya dapat berlaku pada dataset tertentu saja tanpa kemampuan generalisasi pada jenis citra apapun. Hal tersebut dikarenakan berbagai perbedaan antar citra antara lain perbedaan sudut pandang, perbedaan skala, perbedaan kondisi pencahayaan, deformasi objek, dan sebagainya.

Kalangan akademisi yang telah lama bergelut pada *problem* ini. Salah satunya pendekatan yang berhasil digunakan adalah menggunakan Jaringan Syaraf Tiruan (JST) yang terinspirasi dari jaringan syaraf pada manusia. Konsep tersebut kemudian dikembangkan lebih lanjut dalam *Deep Learning*.

Pada tahun 1989, Yann LeCun dan teman-temannya berhasil melakukan klasifikasi citra kode *zip* menggunakan kasus khusus dari *Feed Forward Neural Network* dengan nama *Convolution Neural Network* (CNN)[1]. Karena keterbatasan perangkat keras, *Deep Learning* tidak dikembangkan lebih lanjut hingga pada tahun 2009 dimana Jurgen mengembangkan sebuah *Recurrent Neural Network* (RNN) yang mendapatkan hasil

signifikan pada pengenalan tulisan tangan [2]. Semenjak itu, dengan berkembangnya komputasi pada perangkat keras *Graphical Processing Unit* (GPU), pengembangan DNN berjalan dengan pesat. Pada tahun 2012, sebuah CNN dapat melakukan pengenalan citra dengan akurasi yang menyaingi manusia pada dataset tertentu [3]. Dewasa ini, *Deep Learning* telah menjadi salah satu topik hangat dalam dunia *Machine Learning* karena kapabilitasnya yang signifikan dalam memodelkan berbagai data kompleks seperti citra dan suara.

Metode *Deep Learning* yang saat ini memiliki hasil paling signifikan dalam pengenalan citra adalah *Convolutional Neural Network* (CNN) [4]. Hal tersebut dikarenakan CNN berusaha meniru sistem pengenalan citra pada visual cortex manusia [5] sehingga memiliki kemampuan mengolah informasi citra. Namun CNN, seperti metode *Deep Learning* lainnya, memiliki kelemahan yaitu proses pelatihan model yang lama. Dengan perkembangan perangkat keras, hal tersebut dapat diatasi menggunakan teknologi *General Purpose Graphical Processing Unit* (GPGPU).

Tugas Akhir ini berkaitan dengan klasifikasi citra pada obyek tertentu dengan menggunakan jaringan syaraf konvolusional yang pernah dibuat oleh Dan C. Ciresan, Ueli Meier, Jonathan Masci, Luca M. Gambardella, Jurgen Schmidhuber dalam "Flexible, High Performance Convolutional Neural Networks for Image Classfication".

1.2 Rumusan Permasalahan

Rumusan masalah yang diangkat pada Tugas Akhir ini adalah sebagai berikut:

1. Membangun model *Convolutional Neural Network* (CNN) untuk klasifikasi objek pada citra
2. Meminimalkan tingkat *error* yang didapat
3. Menyusun uji coba klasifikasi pada data citra Caltech 101 menggunakan metode *deep learning* (*Convolution Neural Network*).

1.3 Batasan Masalah

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

1. Data yang digunakan untuk klasifikasi citra objek adalah data citra objek dari basis data Caltech 101.
2. Data yang digunakan untuk data *training* dan data klasifikasi citra objek berupa citra dengan tingkat *confusion* yang berbeda dari basis data Caltech 101.
3. Implementasi menggunakan perangkat lunak MATLAB R2013a.

1.4 Tujuan

Tujuan dari Tugas Akhir ini adalah untuk membangun model *Convolutional Neural Network* yang dapat melakukan klasifikasi objek pada citra.

1.5 Manfaat

Mempermudah dan mempercepat proses klasifikasi citra objek dan mengurangi tingkat kesalahan dalam klasifikasi sehingga hasil yang diperoleh lebih akurat daripada proses manual.

1.6 Metodologi

Tahapan-tahapan yang dilakukan dalam pengerjaan Tugas Akhir ini adalah sebagai berikut:

1. Penyusunan propasal Tugas Akhir

Tahap awal untuk memulai pengerjaan Tugas Akhir adalah penyusunan proposal Tugas Akhir. Proposal Tugas Akhir yang diajukan memiliki gagasan yang sama dengan Tugas Akhir ini, yaitu implementasi metode Deep Neural Network untuk klasifikasi citra untuk dataset Caltech-101.

2. Studi literatur

Pada tahap ini dilakukan pencarian, pengumpulan, pembelajaran dan pemahaman informasi dan literatur yang berhubungan dengan algoritma yang digunakan dalam pengerjaan Tugas Akhir ini. Diantaranya mengenai *Neural Network, Deeplearning Neural Network* dan *Backward propagation*

3. Analisis dan desain perangkat lunak

Pada tahap ini berisi analisis dan desain perangkat lunak terhadap segmentasi citra. Dilakukan proses analisa terhadap permasalahan yang diangkat pada Tugas Akhir ini dan merancang perangkat lunak dengan menentukan data yang digunakan serta proses-proses yang dilakukan dalam Tugas Akhir ini.

4. Implementasi perangkat lunak

Implementasi merupakan tahap membangun rancangan program yang telah dibuat. Pada tahapan ini merealisasikan apa yang terdapat pada tahapan sebelumnya dengan menggunakan Matlab.

5. Pengujian dan evaluasi

Pada tahapan ini dilakukan uji coba pada data citra. Pengujian dan evaluasi akan dilakukan dengan membandingkan hasil klasifikasi dengan hasil *testing*.

6. Penyusunan buku Tugas Akhir

Pada tahapan ini disusun buku yang memuat dokumentasi mengenai pembuatan serta hasil dari implementasi perangkat lunak yang telah dibuat.

1.7 Sistematika Penulisan

Buku Tugas Akhir ini bertujuan untuk mendapatkan gambaran dari pengerjaan Tugas Akhir ini. Selain itu, diharapkan dapat berguna untuk pembaca yang tertarik untuk melakukan pengembangan lebih lanjut. Secara garis besar, buku Tugas Akhir terdiri atas beberapa bagian seperti berikut ini:

Bab I Pendahuluan

Bab ini berisi mengenai latar belakang, tujuan, dan manfaat dari pembuatan Tugas Akhir. Selain itu, permasalahan, batasan masalah, metodologi yang digunakan, dan sistematika penulisan juga merupakan bagian dari bab ini.

Bab II Tinjauan Pustaka

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

Bab III Analisis dan Perancangan

Bab ini membahas tahap analisis permasalahan dan perancangan dari sistem yang akan dibangun. Analisis permasalahan membahas permasalahan yang diangkat dalam pengerjaan Tugas Akhir.

Bab IV Implementasi

Bab ini membahas implementasi dari desain yang telah dibuat pada bab sebelumnya. Bab ini berisi proses implementasi dari setiap kelas pada semua modul.

Bab V Pengujian Dan Evaluasi

Bab ini membahas pengujian dengan metode pengujian blackbox untuk mengetahui penilaian aspek ketepatan dalam megimplementasikan model (correctness) yang telah dibuat pada Aplikasi.

Bab VI Kesimpulan Dan Saran

Bab ini merupakan bab terakhir yang menyampaikan kesimpulan dari hasil uji coba yang dilakukan dan saran untuk pengembangan perangkat lunak ke depannya.

BAB II
TINJAUAN PUSTAKA

Bab ini berisi penjelasan teori yang berkaitan dengan pengimplementasian perangkat lunak. Penjelasan ini bertujuan untuk memberikan gambaran secara umum terhadap sistem yang dibuat dan berguna sebagai penunjang dalam pengembangan perangkat lunak.

2.1 Feature Learning

Berbeda dengan *feature engineering* yang digunakan dalam *machine learning* pada umumnya, *feature learning* adalah metode dimana proses *feature extraction* dilakukan secara otomatis dan adaptif oleh model [2]. *Feature learning* muncul karena *feature engineering* sangatlah terbatas dalam artian tiap kasus data yang berbeda memerlukan *feature extraction* yang berbeda. Hal tersebut menjadikan metode *feature engineering* tidak memiliki kemampuan generalisasi pada keragaman jenis data seperti yang dibutuhkan dalam kasus klasifikasi objek pada citra.

Pada pengolahan data kompleks seperti citra dan suara, umumnya dilakukan ekstraksi fitur untuk mengubah data menjadi bentuk yang dapat dimengerti oleh metode *learning.* Namun proses tersebut sagat memakan waktu dan cenderung kurang dapat menggambarkan keseluruhan nilai informasi dari data. *Feature learning* mengatasi hal tersebut dengan membuat proses ekstraksi fitur adaptif yang dapat melakukan penyesuaian otomatis terhadap data yang digunakan.

Dalam perkembangannya, terdapat beragam metode *feature learning* yang dikembangkan dan secara garis besar dibagi menjadi *unsupervised feature learning* dan *supervised feature learning*. Metode *unsupervised* adalah metode *feature learning* yang berkembang terlebih dahulu dan memiliki tujuan untuk mentransformasi data menjadi representasi lain yang lebih mudah dipahami oleh komputer. Diantaranya terdapat *Autoencoder*,

Deep Belief Network, *Gaussian Mixture Models*, dan bahkan terdapat juga metode *K-Means* [3].

Pada awal perkembangannya, klasifikasi objek pada citra menggunakan metode tersebut, namun kemudian berkembang metode *feature learning* yang bersifat *supervised* sehingga metode *unsupervised* mulai ditinggalkan dikarenakan memiliki performa yang lebih buruk. Metode *supervised feature learning* yang banyak berkembang adalah *deep learning feature learning* dimana menggunakan *deep network* yang melakukan proses ekstraksi fitur dan klasifikasi menggunakan *learning*.

Model *deep learning* yang umum digunakan pada pengolahan citra adalah *Convolutional Neural Network* (CNN). Dalam kasus citra, *feature learning* dilakukan oleh CNN pada serangkaian *convolution layer* di dalamnya. Karena hal tersebut, CNN tidak memerlukan proses ekstraksi fitur khusus dan pada umumnnya hanya memerlukan praproses dasar untuk normalisasi data.

2.2 Convolutional Neural Network

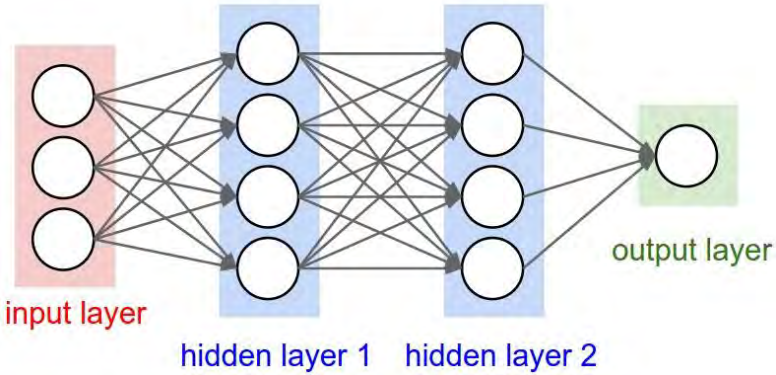
Convolutional Neural Network (CNN) adalah pengembangan dari *Multilayer Perceptron* (MLP) yang didesain untuk mengolah data dua dimensi. CNN termasuk dalam jenis *Deep Neural Network* karena kedalaman jaringan yang tinggi dan banyak diaplikasikan pada data citra. Pada kasus klasifikasi citra, MLP kurang sesuai untuk digunakan karena tidak menyimpan informasi spasial dari data citra dan menganggap setiap piksel adalah fitur yang independen sehingga menghasilkan hasil yang kurang baik.

CNN pertama kali dikembangkan dengan nama NeoCognitron oleh Kunihiko Fukushima, seorang peneliti dari *NHK Broadcasting Science Research Laboratories,* Kinuta, Setagaya, Tokyo, Jepang [4]. Konsep tersebut kemudian dimatangkan oleh Yann LeChun, seorang peneliti dari *AT&T Bell Laboratories* di Holmdel, New Jersey, USA. Model CNN dengan nama LeNet berhasil diterapkan oleh LeChun pada penelitiannya

mengenai pengenalan angka dan tulisan tangan [1]. Pada tahun 2012, Alex Krizhevsky dengan penerapan CNN miliknya berhasil menjuarai kompetisi *ImageNet Large Scale Visual Recognition Challenge* 2012. Prestasi tersebut menjadi momen pembuktian bahwa metode *Deep Learning*, khususnya CNN. Metode CNN terbukti berhasil mengungguli metode *Machine Learning* lainnya seperti SVM pada kasus klasifikasi objek pada citra.

2.2.1 Konsep CNN

Cara kerja CNN memiliki kesamaan pada MLP, namun dalam CNN setiap neuron dipresentasikan dalam bentuk dua dimensi, tidak seperti MLP yang setiap neuron hanya berukuran satu dimensi.



Gambar 2.1 Arsitektur MLP Sederhana [5]

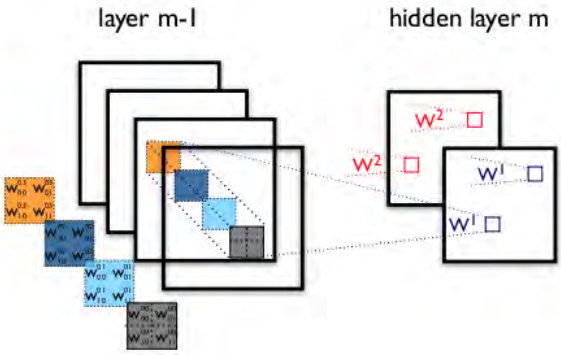
Sebuah MLP seperti pada Gambar 2.1 memiliki *i layer* (kotak merah dan biru) dengan masing-masing *layer* berisi j_i neuron (lingkaran putih). MLP menerima *input* data satu dimensi dan mempropagasikan data tersebut pada jaringan hingga menghasilkan *output*. Setiap hubungan antar neuron pada dua *layer* yang bersebelahan memiliki parameter bobot satu dimensi yang menentukan kualitas mode. Disetiap data *input* pada *layer* dilakukan operasi *linear* dengan nilai bobot yang ada, kemudian

hasil komputasi akan ditransformasi menggunakan operasi *non linear* yang disebut sebagai fungsi aktivasi.

Pada CNN, data yang dipropagasikan oleh jaringan adalah data dua dimensi, sehingga operasi linear dan parameter bobot pada CNN berbeda. Pada CNN operasi linear menggunakan operasi konvolusi, sedangkan bobot tidak lagi satu dimensi saja, namun berbentuk empat dimensi yang merupakan kumpulan kernel konvolusi seperti pada Gambar 2.2. Dimensi bobot pada CNN adalah:

neuron input x *neuron output* x *tinggi* x *lebar*

Karena sifat proses konvolusi, maka CNN hanya dapat digunakan pada data yang memiliki struktur dua dimensi seperti citra dan suara.



Gambar 2.2 Proses Konvolusi pada CNN [6]

2.2.2 Arsitektur Jaringan CNN

JST terdiri dari berbagai *layer* dan beberapa neuron pada masing-masing *layer*. Kedua hal tersebut tidak dapat ditentukan menggunakan aturan yang pasti dan berlaku berbeda-beda pada data yang berbeda [7].

Pada kasus MLP, sebuah jaringan tanpa *hidden layer* dapat memetakan persamaan *linear* apapun, sedangkan jaringan dengan

satu atau dua *hidden layer* dapat memetakan sebagian besar persamaan pada data sederhana. Namun pada data yang lebih kompleks, MLP memiliki keterbatasan. Pada permasalahan jumlah *hidden layer* dibawah tiga *layer*, terdapat pendekatan untuk menentukan jumlah neuron pada masing-masing *layer* untuk mendekati hasil optimal. Penggunaan layer diatas dua pada umumnya tidak direkomendasikan dikarenakan akan menyebabkan *overfitting* serta kekuatan *backpropagation* berkurang secara signifikan.

Dengan berkembangnya *deep learning*, ditemukan bahwa untuk mengatasi kekurangan MLP dalam menangani data kompleks, diperlukan fungsi untuk mentransformasi data input menjadi bentuk yang lebih mudah dimengerti oleh MLP. Hal tersebut memicu berkembangnya *deep learning* dimana dalam satu model diberi beberapa *layer* untuk melakukan transformasi data sebelum data diolah menggunakan metode klasifikasi. Hal tersebut memicu berkembangnya model neural network dengan jumlah *layer* diatas tiga. Namun dikarenakan fungsi *layer* awal sebagai metode ekstraksi fitur, maka jumlah *layer* dalam sebuah DNN tidak memiliki aturan universal dan berlaku berbeda-beda tergantung dataset yang digunakan.

Karena hal tersebut, jumlah *layer* pada jaringan serta jumlah neuron pada masing-masing *layer* dianggap sebagai *hyperparameter* dan dioptimasi menggunakan pendekatan *searching*.

Sebuah CNN terdiri dari beberapa *layer*. Berdasarkan aristektur LeNet5 [8], terdapat empat macam *layer* utama pada sebuah CNN namun yang diterapkan pada TA ini hanya tiga macam lapisan lantara lain:

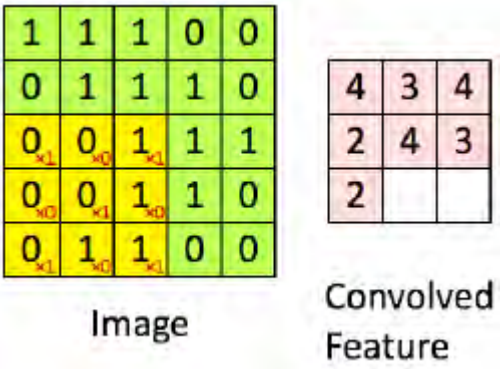
a. *Convolution Layer*

Convolution Layer melakukan operasi konvolusi pada *output* dari *layer* sebelumnya. *Layer* tersebut adalah proses utama yang mendasari sebuah CNN.

Konvolusi adalah suatu istilah matematis yang berati mengaplikasikan sebuah fungsi pada *output* fungsi lain

secara berulang. Dalam pengolahan citra, konvolusi berati mengaplikasikan sebuah *kernel*(kotak kuning) pada citra disemua *offset* yang memungkinkan seperti yang ditunjukkan pada Gambar 2.3. Kotak hijau secara keseluruhan adalah citra yang akan dikonvolusi. *Kernel* bergerak dari sudut kiri atas ke kanan bawah. Sehingga hasil konvolusi dari citra tersebut dapat dilihat pada gambar disebelah kanannya.

Tujuan dilakukannya konvolusi pada data citra adalah untuk mengekstraksi fitur dari citra *input*. Konvolusi akan menghasilkan transformasi *linear* dari data *input* sesuai informasi spasial pada data. Bobot pada *layer* tersebut menspesifikasikan *kernel* konvolusi yang digunakan, sehingga kernel konvolusi dapat dilatih berdasarkan *input* pada CNN.

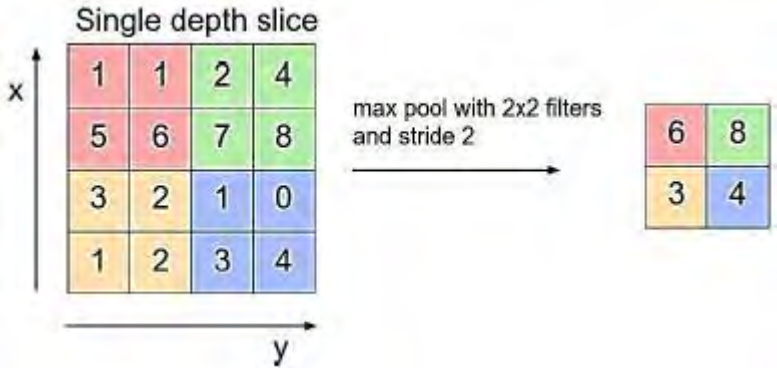


Gambar 2.3 Operasi Konvolusi [9]

b. *Subsampling Layer*

Subsampling adalah proses mereduksi ukuran sebuah data citra. Dalam pengolahan citra, *subsampling* juga bertujuan untuk meningkatkan invariansi posisi dari fitur. Dalam sebagian besar CNN, metode *subsampling* yang digunakan adalah *max pooling*. *Max pooling* membagi output dari *convolution layer* menjadi beberapa *grid* kecil

lalu mengambil nilai maksimal dari setiap *grid* untuk menyusun matriks citra yang telah direduksi seperti yang ditunjukkan pada Gambar 2.4. Grid yang berwarna merah, hijau, kuning dan biru merupakan kelompok grid yang akan dipilih nilai maksimumnya. Sehingga hasil dari proses tersebut dapat dilihat pada kumpulan grid disebelah kanannya. Proses tersebut memastikan fitur yang didapatkan akan sama meskipun objek citra mengalami translasi (pergeseran).



Gambar 2.4 Operasi Max Pooling [5]

Menurut Springenberg et al. [10], penggunaan *pooling layer* pada CNN hanya bertujuan untuk mereduksi ukuran citra sehingga dapat dengan mudah digantikan dengan sebuah *convolution layer* dengan *stride* yang sama dengan *pooling layer* yang bersangkutan.

c. *Fully Connected Layer*

Layer tersebut adalah *layer* yang biasanya digunakan dalam penerapan MLP dan bertujuan untuk melakukan transformasi pada dimensi data agar data dapat diklasifikasikan secara linear.

Setiap *neuron* pada *convolution layer* perlu ditransformasi menjadi data satu dimensi terlebih dahulu

sebelum dapat dimasukkan ke dalam sebuah *fully connected layer*. Karena hal tersebut menyebabkan data kehilangan informasi spasialnya dan tidak reversibel, *fully connected* layer hanya dapat diimplementasikan di akhir jaringan.

Dalam sebuah jurnal oleh Lin et al., dijelaskan bahwa *convolution layer* dengan ukuran kernel 1 x 1 melakukan fungsi yang sama dengan sebuah *fully connected layer* namun dengan tetap mempertahankan karakter spasial dari data. Hal tersebut membuat penggunaan *fully connectedlayer* pada CNN sekarang tidak banyak dipakai.

2.2.3 Fungsi Aktivasi

Fungsi aktivasi adalah fungsi *non linear* yang memungkinkan sebuah JST untuk dapat mentransformasi data *input* menjadi dimensi yang lebih tinggi sehingga dapat dilakukan pemotongan *hyperlane* sederhana yang memungkinkan dilakukan klasifikasi. Dalam CNN terdapat fungsi aktivasi digunakan yaitu fungsi *sigmoid*.

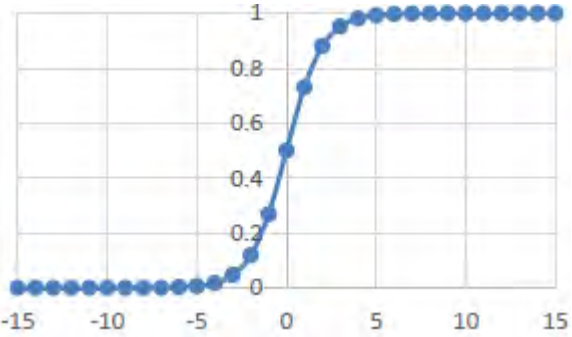
Fungsi *sigmoid* mentransformasi *range* nilai dari *input* x menjadi antara 0 dan 1 dengan bentuk distribusi fungsi seperti pada Gambar 2.5. Sehingga fungsi *sigmoid* memiliki bentuk sebagai berikut:

$$\sigma(x) = \frac{1}{(1 + e^{-x})} \tag{2.1}$$

Fungsi *sigmoid* sekarang sudah tidak banyak digunakan dalam praktek karena memiliki kelemahan utama yaitu *range* nilai *output* dari fungsi *sigmoid* tidak terpusat pada angka nol.

Hal tersebut menyebabkan terjadinya proses *backpropagation* yang tidak ideal, selain itu bobot pada JST tidak terdistribusi rata antara nilai positif dan negatif serta nilai bobot akan banyak mendekati ekstrim 0 atau 1. Dikarenakan komputasi nilai propagasi menggunakan perkalian, maka nilai ekstrim tersebut akan menyebabkan efek *saturating gradients* dimana jika nilai bobot cukup kecil, maka lama kelamaan nilai bobot akan

mendekati salah satu ekstrim sehingga memiliki gradien yang mendekati nol. Jika hal tersebut terjadi, maka neuron tersebut tidak akan dapat mengalami *update* yang signifikan dan akan nonaktif.



Gambar 2.5 Distribusi Fungsi Sigmoid

2.2.4 Parameter CNN

Seluruh JST adalah sebuah *parametric model* yang berati sebuah model yang melakukan *fiting* terhadap data dengan memodifikasi nilai dari parameter jaringannya (bobot dan bias). JST merupakan metode *machine learning* yang memiliki kelemahan utama yaitu banyaknya parameter pada model, baik parameter internal jaringan maupun parameter eksternal. Hal tersebut membuat model pelatihan serta model optimasi pada JST menjadi sebuah proses yang sangat memakan waktu.

Karena dalam *machine learning* terdapat banyak sekali parameter dalam pembuatan model, maka dikenal sebuah istilah yaitu hyperparameter. Hyperparameter didefinisikan sebagai parameter dari sebuah distribusi di luar distribusi pada model [19]. Dalam konteks modelklasifikasi yang sedang dipelajari.

Pada JST, yang disebut parameter adalah nilai bobot dan bias dari *layer*. Seluruh parameter lain disebut dengan istilah hyperparameter karena berada di luar domain distribusi dari model.

BAB III
PERANCANGAN

Pada Bab 3 dijelaskan secara singkat mengenai sistem dasar dalam pengerjaan Tugas Akhir ini, maka pada Bab 3 akan dijelaskan secara lebih detail dan menyeluruh mengenai penggunaan metode *Forward Propagation* dan *Backpropagation* untuk klasifikasi citra, serta peralatan yang digunakan dalam melakukan implementasi.

3.1 Bahan dan Peralatan yang Digunakan

Perangkat lunak yang digunakan dalam pengimplemetnasian klasifikasi citra pada dataset Caltech-101 adalah sebagai berikut:

Tabel 3.1 Bahan dan Peralatan

Perangkat Keras	Intel(R) Core(TM)2 Duo CPU T6600 @ 2.20 GHz 2.20 GHz RAM 2,00 GB
Perangkat Lunak	Matlab R2013a Windows 7 Ultimate-32 bit

3.2 Data Masukan

Data masukan pada sistem ini berupa citra dengan telah dirubah menjadi *gray scale* data citra yang digunakan adalah dataset Caltech-101. Caltech-101 adalah dataset dengan data citra yang terdiri dari 101 kelas objek dan 1 kelas *background*. Jumlah data disetiap kelas antara 20 data citra sampai 800 data citra. Jumlah data citra disetiap kelas rata-rata 50 data citra.

Data citra ini berhasil terkumpul pada tahun 2003 bulan September oleh Fei-Fei Li, Marco Andreetto, dan Marc' Aurelio Ranzato. Ukuran citra pada dataset ini tidak merata akan tetapi secara garis besar ukuran citra disetiap kelas sebesar 300 x 200 piksel.

Penerapan pada dataset Caltech-101 biasanya menggunakan data citra sebanyak 20 sampai 30 data citra disetiap kelasnya untuk pelatihan. Kategori-kategori yang akan digunakan pada uji coba dari CNN ini adalah *Emu, Flamingo, Ebis, Pigeon, Rooster, Face, Cougar,* dan *Crocodile*.

3.3 Data Keluaran

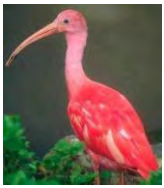
Keluaran akhir pada sistem ini adalah tingkat akurasi kebenaran dari klasifikasi data citra dari Dataset Caltech-101 dan data-data citra yang berhasil ataupun yang gagal diklasifikasi.

3.4 Pengolahan data *Input* dan perancangan Praproses

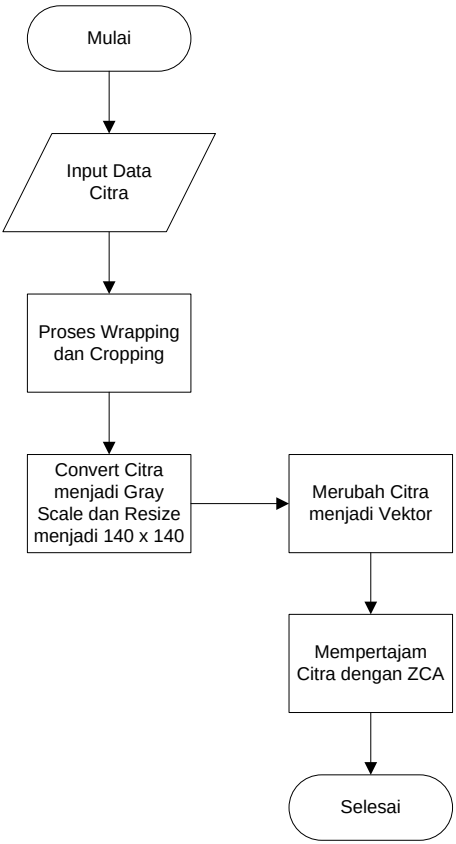
Salah satu proses utama yang ada pada aplikasi ini adalah praproses. Gambaran umum dari praproses dan tahap *training* digambarkan pada Gambar 3.3 dan Gambar 3.8. Citra masukan akan diolah ke dalam pra proses yaitu proses *wrapping* dan *cropping*. Pada *wrapping*, citra masukan dilakukan pengecekan terhadap *edge* dari objek utama pada citra tersebut. Dari *edge* pada citra tersebut ditentukan *edge* maksimalnya sehingga saat hasil *cropping* objek pada citra tersebut tetap utuh seperti pada Gambar 3.1 dan Gambar 3.2. Tahap *training* dimulai dengan merubah citra menjadi bentuk vektor yang telah dirubah dicerahkan dengan menggunakan *ZCA*.

(a) (b)

Gambar 3.1 (a) Data Citra awal (b) Data Citra mengalami pemusatan Objek



Gambar 3.2 Citra hasil *wrapping* dan *cropping* dengan *Annotation* dari Caltech 101



Gambar 3.3 Alur praproses dan pengolahan data *input*

Masukan	Citra Caltech 101
Keluaran	Citra Caltech 101 dalam bentuk vektor

```
1.  nClass ← 5
2.  nData ← 100(untuk training), 50(untuk
    testing)
3.  hasil ← direktori data citra disimpan
    annot ← direktori data annotation disimpan
4.  antC ← 1
5.  for ←x←3 to sejumlah data annotation yang
        ada
6.      namefolAn ← strcat('nama path',nama
        folder)
7.      tempnameAn ← dir(namefolAn)
8.      targetAn ← zeros(sejumlah nclass)
9.      targetAn(str2double(nama file)) ← 1
10.     for y←1 to sejumlah folder yang ada
            pada direktori tersebut
11.         fileAnnot ← strcat(nama
            folder,'\',tempnameAn(y).name)
12.         dTrain(antC).fileAnnot ← fileAnnot
13.         antC ← antC+1
14.     end for
15. end for
16. c ← 1
17. data training X/data testing X ←
            zeros(sejumlah data,ukuran citra)
18. data training Y/data testing Y ←
            zeros(sejumlah data,jumlah kelas)
19. for x←3 to sejumlah data citra yang ada
20.     namefol ← strcat('nama path',nama
        folder)
21.     tempnameAn ← dir(namefol)
22.     target ← zeros(sejumlah nclass)
23.     target(str2double(nama file)) ← 1
24.     for y←1 to sejumlah folder yang ada
            pada direktori tersebut
```

Gambar 3.4 *Pseudocode* Pengolahan *Input* (bagian pertama)

```
25.          file ← strcat(nama
             folder,'\',tempname(y).name)
26.          dTrain(c).file ← file
27.          c ← c+1
28.      end for
29.  end for
30.  vec ← 1
31.  for y←1 to nData
32.      X ← imread(file)
33.      file ← dTrain(y).file
34.      fileAnnot ← dTrainAn(y).fileAnnot
35.      [box1,box2,box3,box4] ←
         show_annotation(file,fileAnnot)
36.      imgAnot ←
         imcrop(X,[box1,box2,box3,box4])
37.      I ← rgb2gray(imgAnot)
38.      I ← imresize(I,[140 140])
39.      Ivec ← reshape(I',1,19600)
40.      dataTrainX(vec,:) ← Ivec
41.      dataTrainY(vec,:) ← target
42.      vec ← vec+1
43.  end for
44.  save('Caltech101X','dataTrainX')
45.  save('Caltech101Y','dataTrainY')
```

Gambar 3.5 *Pseudocode* Pengolahan *Input* (bagian kedua)

Proses pengolahan data citra dimulai dengan citra ukuran sembarang yang kemudian dirubah ukurannya menjadi 140 x 140. Citra tersebut dijadikan *grey scale* agar bisa diproses dengan mudah pada tahap *Training. Pseudocode* dari alur praproses dan pengolahan data *input* ditunjukkan pada Gambar 3.4 sampai Gambar 3.7.

3.4.1 Wrapping Citra Caltech 101

Program *wrapping* pada citra caltech 101 merupakan program pemusatan objek pada citra caltech 101 dengan memusatkan citra pada objek yang akan diklasifikasi akan

meningkatkan akurasi kebenaran dari klasifikasi yang dilakukan. Sebelum citra masukan menjadi ukuran 140 x 140. Citra dengan ukuran sembarang tersebut di-*crop* dengan menentukan batasan disetiap *edge* dari citra tersebut. Untuk mengetahui batasan dari citra tersebut dimasukkan data *anotation* dari Caltech 101. *Pseudocode* proses *wrapping* ditunjukkan pada Gambar 3.6 dan Gambar 3.7.

Masukan	Citra Caltech 101
Keluaran	Batasan-batasan objek pada citra Caltech

```
1.  IMTYPE ← 'jpg'
2.  GUIDELINE_MODE ← 1
3.  LARGEFONT ← 28
4.  MEDFONT ← 18
5.  BIG_WINDOW ← get(0,'ScreenSize')
6.  SMALL_WINDOW ← [100 100 512 480]
    load(annotation_file, 'box_coord',
    'obj_contour')
7.  ima ← imread(imgfile)
8.  ff ← figure(1). clf. imagsec(ima).
       axis image. axis ij. hold on.
9.  if length(ukuran ima) < 3
10.    colormap(gray)
11. end if
    set(ff,'Position', SMALL_WINDOW)
12. box_handle ← rectangle('position',
    [box_coord(3),box_coord(1),box_coord(4)-
    box_coord(3),box_coord(2)-box_coord(1)])
    set(box_handle,'edgecolor','y',
        'linewidth',5)
13. box1 ← box_coord(3)
14. box2 ← box_coord(1)
15. box3 ← box_coord(4)-box_coord(3)
16. box4 ← box_coord(2)-box_coord(1)
```

Gambar 3.6 *Pseudocode* Praproses (bagian pertama)

```
17.  for cc ← 1 to nilai obj_contour
18.       if cc < nilai obj_contour
             plot([obj_contour(1,cc),
19.          obj_contour(1,cc+1)]+box_coord(3),
             [obj_contour(2,cc),
20.          obj_contour(2,cc+1)]+box_coord(1),
             'r','linewidth',4)
21.       else
             plot(obj_contour(1,cc),
             obj_contour(1,1)]+box_coord(3),
             [obj_contour(2,cc),
             obj_contour(2,1)]+box_coord(1),'r',
             'linewidht',4)
22.       end if
23.  end for
```

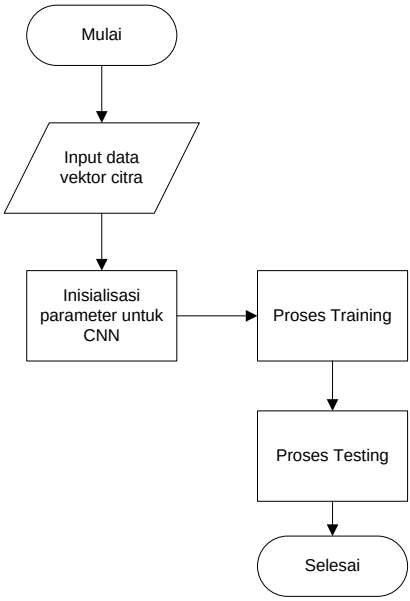
Gambar 3.7 *Pseudocode* Praproses (bagian kedua)

3.5 Program Utama

Program utama merupakan program yang memanggil fungsi-fungsi lain untuk menjalankan program secara keseluruhan. Proses-proses yang dilakukan dalam algoritma yang digunakan, dipisahkan dalam fungsi-fungsi yang berbeda untuk memudahkan pengecekan program untuk setiap prosesnya.

Fungsi yang pertama adalah *input* data *vektor*, fungsi ini mengambil parameter berupa jumlah data yang akan di *Training* dan jumlah kategori yang akan diklasifikasi.

Kedua fungsi *Training* dan fungsi *Testing*. Pada program utama parameter yang diperlukan adalah jumlah data *Training*, jumlah data *Testing*, jumlah *layer* yang akan dibentuk, ukuran *layer* yang akan dibentuk, ukuran *subsampling* dan jumlah iterasi yang diperlukan. *Pseudocode* program utama ditunjukkan pada Gambar 3.9.



Gambar 3.8 Alur proses program utama

Masukan	Citra Caltech 101 dalam bentuk vektor
Keluaran	Akurasi dan kecepatan klasifikasi yang dilakukan

1. dataTrainX ← citra vektor dengan ukuran 140x140 sebanyak 100 data dengan min piksel 0 dan max piksel 255
2. dataTestX ← citra vektor dengan ukuran 140x140 sebanyak 50 data dengan min piksel 0 dan max piksel 255
3. dataTrainY ← jumlah kelas pada data training
4. dataTestY ← jumlah kelas pada data testing
5. cnn.layers ← inisiasi parameter CNN
6. input ← dataTrainX, dataTrainY, dataTestX, dataTestY

Gambar 3.9 *Pseudocode* program utama (bagian Pertama)

```
7.  C1 ← jumlah map(16), kernelsize(16)
8.  M2 ← scale(5)
9.  C3 ← jumlah map(128), kernelsize(6)
10. M4 ← scale(5)
11. Seed generator ← 0
12. opts.alpha ← 1
13. opts.batchsize ← 50
15. opts.numepochs ← 500
16  cnn ← cnnsetup(parameter CNN, dataTrainX,
    dataTrainY)
17. starTime ← tic()
19. cnn ← cnntrain(parameter CNN, dataTrainX,
    dataTrainY, parameter opts)
    [er,bad] ← cnntest(parameter CNN,
20. dataTestX, dataTestY)
21. numRight ← jumlah keseluruhan data –
    jumlah data yang salah
```

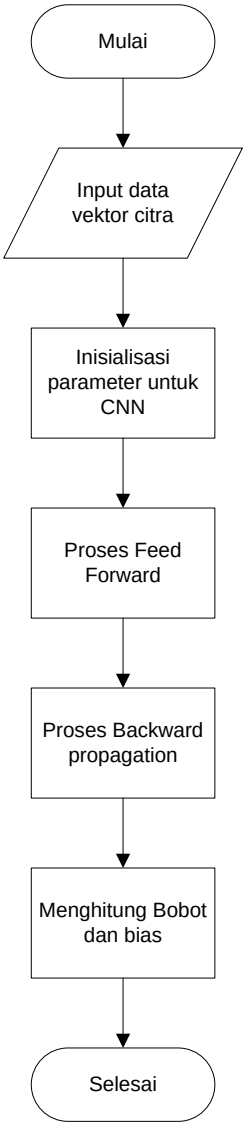
Gambar 3.10 *Pseudocode* program utama (bagian Kedua)

3.6 Perancangan proses *Training*

Proses *training* merupakan tahapan dimana CNN dilatih untuk memperoleh akurasi yang tinggi dari klasifikasi yang dilakukan. Tahapan ini terdiri dari proses *feed forward* dan proses *backpropagation*. Untuk memulai proses *feedforward* diperlukan jumlah dan ukuran *layer* yang akan dibentuk, ukuran *subsampling*, citra vektor yang diperoleh pada subbab 3.4.

Proses *feedforward* bekerja seperti pada subbab 2.2.2 dimana citra vektor akan melalui proses konvolusi dan *Max pooling* untuk mereduksi ukuran citranya dan memperbanyak neuronnya. Sehingga terbentuk banyak jaringan yang mana menambah variant data untuk dipelajari.

Hasil dari proses *feedforward* berupa bobot yang akan digunakan untuk mengevaluasi proses neural network tadi. Alur proses dan *pseudocode* proses *training* ditunjukkan pada Gambar 3.11 sampai Gambar 3.12.



Gambar 3.11 Alur proses *Training* secara umum

Masukan	Citra Caltech 101 dalam bentuk vektor, parameter CNN
Keluaran	Bobot dan bias yang telah diperbarui

```
1.  m ← ukuran citra
2.  numbatches ← m/opts.batchsize
3.  if rem(numbatches,1) ~= 0
4.     error('numbatches not integer')
5.  end if
6.  net.rL ← []
7.  for i←1 to opts.numepochs
8.      disp(['epoch' num2str(i) '/'
        num2str(opts.numepochs)])
9.      tic
10.     kk ← randperm(m)
11.     for l←1 to numbatches
12.         batch_x ← x(:, :, kk((l-1)*
            opts.batchsize + 1 : 1 *
            opts.batchsize))
13.         batch_y ← y(:,    kk((l-1)*
            opts.batchsize + 1 : 1 *
            opts.batchsize))
14.         net ← cnnff(net, batch_x)
15.         net ← cnnbp(net, batch_y)
16.         net ← cnnapplygrads(net, opts)
17.         if isempty(net.rL)
18.            net.rL(1) ← net.L
19.         end if
20.         net.rL(end + 1) ← 0.99 *
21.         net.rL(end) + 0.01 * net.L
22.     end for
23.     toc
24. end for
```

Gambar 3.12 *Pseudocode* Proses *Training*

3.6.1 Perancangan proses *feedforward*

Proses *feedforward* merupakan tahap pertama dalam proses *training*. Proses ini akan menghasilkan beberapa lapisan

untuk mengklasifikasi data citra yang mana menggunakan bobot dan bias yang telah diperbarui dari proses *backpropagation*. Pada awal tahap ini bobot yang digunakan merupakan nilai *random*. Setelah satu proses *Training* selesai bobot dan bias yang baru digunakan untuk proses *Training* yang baru. Tahap ini juga akan digunakan kembali saat proses *testing*. *Pseudocode feedforward* ditunjukkan pada Gambar 3.13.

Masukan	Citra Caltech 101 dalam bentuk vektor, parameter CNN
Keluaran	Struktur CNN dari data Citra yang akan diklasifikasi dan dilatih

```
1.  n ← numel(jumlah lapisan yang telah
      ditentukan)
2.  net.layers{1}.a{1} ← data citra
3.  inputmaps ← 1
4.  for l ← 2 to n
5.      if strcmp(net.layers adalah 'c')
6.         for j←1 to net.layers{l}.outputmaps
7.             z ← zeros((jumlah lapisan{l-1})
               -[kernelsize-1,kernelsize-1,0])
8.             for i←1 to inputmaps
9.                 z ← z + convn(net.layers{l-
                   1}.a{i}, net.layers{l}.
                   k{i}{j}, 'valid')
10.            end for
11.            net.layers{l}.a{j} ← sigm(z +
               net.layers{l}.b{j})
12.        end for
13.        inputmaps ←
14.        net.layers{l}.outputmaps;
15.     elseif strcmp(net.layers adalah 's')
16.            for j←1 to inputmaps
17.                z ← convn(net.layers{l -
                   1}.a{j}, ones
```

Gambar 3.13 *Pseudocode* Proses *feedforward* (bagian pertama)

```
      (net.layers{l}.scale)/
                    (net.layers{l}.scale ^ 2),
                    'valid')
18.   net.layers{l}.a{j} ← z(1 :
                    net.layers{l}.scale : end,
                    1:net.layers{l}.scale:end,:)
19              end for
20.       end if
21.   end for
22.   net.fv ← []
23.   for j ← 1 to numel(jumlah lapisan yang
          telah ditentukan)
24.       sa ← size(net.layers{n}.a{j})
25.       net.fv ← [net.fv;
26.       reshape(net.layers{n}.a{j}, sa(1)*
          sa(2), sa(3))]
27.   end for
28.   net.o ← sigm(net.ffW * net.fv +
29.   repmat(net.ffb, 1, size(net.fv, 2)))
```

Gambar 3.14 *Pseudocode* Proses *feedforward* (bagian kedua)

3.6.2 Perancangan proses *backpropagation*

Proses *backpropagation* merupakan tahap kedua dari proses *training*. Pada tahap ini seperti yang telah dijelaskan pada sub bab 2.3.6 hasil proses dari *feed forward* di-*trace* kesalahannya dari lapisan *output* sampai lapisan pertama. Untuk menandai bahwa data tersebut telah di-*trace* diperoleh bobot dan bias yang baru. Proses *Training* dilakukan sebanyak masukkan yang dimasukkan pada program utama seperti yang telah dijelaskan pada subbab 3.5. Keberhasilan klasifikasi berpengaruh pada jumlah iterasi yang dilalukan saat *Training*. *Pseudocode* proses *backpropagation* ditunjukkan pada Gambar 3.15.

Masukan	Citra bentuk vektor hasil dari proses feedforward (y), parameter CNN (net)

Gambar 3.15 *Pseudocode* Proses *backpropagation* (bagian pertama)

Keluaran	Struktur CNN dari data Citra yang akan diklasifikasi dan dilatih (net)

```
     n ← numel(jumlah lapisan yang telah
       ditentukan)
     net.e ← jaringan output – y
3.   net.L ← 1/2*sum(net.e(:).^2)/size(net.e,2)
4.   net.od ← net.e .*(net.o .*(1-net.o))
5.   net.fvd ← (net.ffW' * net.od)
     if strcmp(net.layers{n}.type,'c')
6.      net.fvd ← net.fvd .* (net.fv .* (1-
        net.fv))
21.  end if
22.
23.  sa ← size(net.layers{n}.a{1})
     fvnum ← sa(1) * sa(2)
     for j ← 1 to numel(net.layers{n}.a)
24.
25.      net.layers{n}.d{j} ← reshape
26.      (net.fvd(((j-1)*fvnum+1):j*fvnum,:)
         ,sa(1),sa(2),sa(3))
27.  end for
28.  for l←(n-1) to -1 to 1
29.      if strcmp(net.layers{l} adalah 'c')
30.         for j←1 to numel(net.layers{l}.a)
                net.layers{l}.d{j} ←
                net.layers{l}.a{j} .*
                (1-net.layers{l}.a{j}) .*
                (expand(net.layers{l + 1}.d{j},
                [net.layers{l + 1}.scale
                net.layers{l + 1}.scale 1]) /
31.             net.layers{l + 1}.scale ^ 2)
32.         end for
33.     elseif strcmp(net.layers{l} adalah 's')
                for i←1 to numel
34.                 (net.layers{l}.a)
                    z ← zeros(size
```

Gambar 3.16 *Pseudocode* Proses *back propagation* (bagian kedua)

```
35.                 (net.layers{l}.a{l}))
36.                 for j←1 to numel
37.                    (net.layers{l+1}.a)
38.                     z ← z + convn
39.                    (net.layers{l + 1}.d{j},
40.                    rot180(net.layers{l +
41.                    1}.k{i}{j}), 'full')
42.                      end for
43.                      net.layers{l}.d{i} ← z
44.          end for
45.      end if
46.  end for
47.  for l←2 to n
       if strcmp(net.layers{l} adalah 'c')
          for j←1 to numel(net.layers{l}.a)
              for i←1 to numel
                  (net.layers{l-1}.a)
48.               net.layers{l}.dk{i}{j} ←
49.               convn(flipall(net.layers{l -
                  1}.a{i}), net.layers{l}
                  .d{j}, 'valid') /
50.               size(net.layers{l}.d{j}, 3)
51.           end for
52.           net.layers{l}.db{j} ←
53.           sum(net.layers{l}.d{j}(:)) /
54.           size(net.layers{l}.d{j}, 3)
56.       end for
57.    end if
58.  end for
59.  net.dffW ← net.od*(net.fv)'/size(net.od,2)
     net.dffb ← mean(net.od, 2)
     function X ← rot180(X)
              X ← flipdim(flipdim(X,1),2)
     end function
```

Gambar 3.17 *Pseudocode* Proses *backpropagation* (bagian ketiga)

3.6.3 Perancangan *gradient* untuk CNN

Pada proses penerapan *gradient* untuk jaringan konvolusi merupakan proses untuk memperoleh nilai bobot dan bias yang baru. Hasil dari proses ini akan digunakan untuk proses latihan berikutnya. Proses perhitungan *gradient* termasuk bagian dari proses *backpropagation* karena perhitungan cukup panjang maka fungsi ini dipisah dan lebih mudah untuk mengakses dari proses *feedforward*. *Pseudocode* proses *gradient* ditunjukkan pada Gambar 3.18.

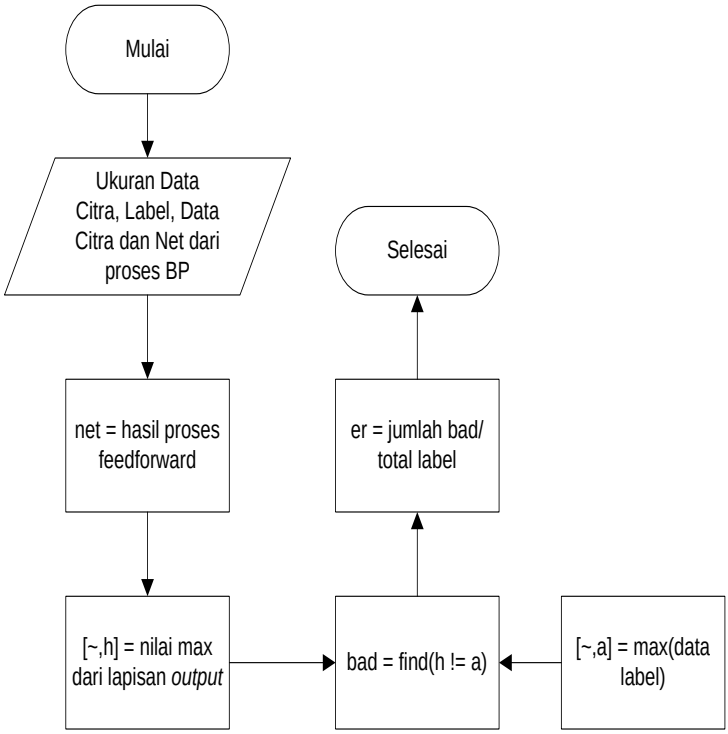
Masukan	Parameter CNN (net), *gradient options*(opt)
Keluaran	Bobot dan bias yang telah diperbarui (net.ffW dan net.ffb)

```
1.  for l←2 to numel(jumlah lapisan yang telah
                ditentukan)
        if strcmp(net.layers{l} adalah 'c')
           for j←1 to numel(net.layers{l}.a)
               for ii←1 to numel(net.layers{l-
                   1}.a)
                   net.layers{l}.k{ii}{j} ←
                   net.layers{l}.k{ii}{j} –
                   opts.alpha * net.layers{l}.
                   dk{ii}{j}
               end for
               net.layers{l}.b{j} ←
               net.layers{l}.b{j} - opts.alpha*
               net.layers{l}.db{j}
           end for
        end if
    end for
    net.ffW ← net.ffW - opts.alpha * net.dffW
    net.ffb ← net.ffb - opts.alpha * net.dffb
```

Gambar 3.18 *Pseudocode* Proses Perhitungan *Gradient*

3.7 Perancangan proses *Testing*

Proses *testing* merupakan proses klasifikasi menggunakan bobot dan bias dari hasil proses *training*. Proses ini tidak jauh berbeda dengan proses *training* yang membedakannya tidak terdapat proses *backpropagation* setelah proses *feedforward*. Sehingga hasil akhir dari proses ini menghasilkan akurasi dari klasifikasi yang dilakukan, data yang gagal diklasifikasi, nomor citra yang gagal diklasifikasi, dan bentuk network yang terbentuk dari proses *feedforward*. Alur proses dan *pseudocode* proses *testing* ditunjukkan pada Gambar 3.19 dan Gambar 3.20.



Gambar 3.19 Alur Proses *Testing*

Masukan	Parameter CNN (net), dimensi data citra (x), label data citra (y)
Keluaran	Nilai kesalahan (er) dan data citra yang gagal diklasifikasi (bad)

```
1.  net ← cnnff(net,x)
    [~, h] ← max(net.o)
    [~, a] ← max(y)
    bad ← find(h tidak sama dengan a)
    er ← numel(bad) / size(y,2)
```

Gambar 3.20 *Pseudocode* Proses *Testing*

BAB IV
IMPLEMENTASI PERANGKAT LUNAK

Pada bab ini akan dibahas mengenai implementasi yang dilakukan berdasarkan rancangan yang telah dijabarkan pada bab sebelumnya. Sebelum penjelasan implementasi akan ditunjukkan terlebih dahulu lingkungan untuk melakukan implementasi.

4.1 Lingkungan Implementasi

Lingkungan implementasi yang akan digunakan untuk melakukan implementasi adalah program MATLAB 8.1.0 (R2013a) yang dipasang pada sistem operasi Windows 7.

4.2 Implementasi

Pada bagian ini akan dijelaskan implementasi setiap subbab yang terdapat pada bab sebelumnya yaitu bab perancangan perangkat lunak. Bagian implementasi ini juga akan dibagi menjadi 3 bagian, yaitu bagian praproses dan pengolahan data *input*, proses *training* dan yang terakhir proses *testing*.

4.2.1 Implementasi Praproses dan Pengolahan Data *Input*

Pada bagian ini akan dijelaskan implementasi dari praposes dan pengolahan data *input*. Implementasi akan dijelaskan mulai dari praproses yang berupa *wrapping* dan *cropping*. Setelah itu dilanjutkan dengan merubah matrik citra menjadi vektor.

4.2.1.1 Implementasi Wrapping dan Cropping citra caltech

Input dari fungsi *wrapping* dan *cropping* berupa data citra caltech yang berukuran rata-rata 300 x 200 piksel dengan tipe data uint8. Citra tersebut akan diproses sehingga menghasilkan citra dengan menonjolkan objek utama dari citra tersebut. Implementasi ini merupakan implementasi *pseudocode* pada subbab 3.4.1. Implementasi dalam matlab ditunjukkan pada Kode Sumber 4.1.

```
1  IMTYPE = 'jpg';
2  GUIDELINE_MODE = 1;
3  LARGEFONT = 28;
4  MEDFONT = 18;
5  BIG_WINDOW = get(0,'ScreenSize');
6  SMALL_WINDOW = [100 100 512 480];
7  load(annotation_file,'box_coord','obj_contou
   r');
8  ima = imread(imgfile);
9  ff=figure(1); clf; imagesc(ima); axis image;
   axis ij; hold on;
10 if length(size(ima)) < 3
11    colormap(gray);
12 end
13 set(ff,'Position',SMALL_WINDOW);
14 box_handle=rectangle('position',[box_coord(3
   ),box_coord(1),box_coord(4)-box_coord(3),
   box_coord(2)-box_coord(1)]);
   set(box_handle,'edgecolor','y','linewidth',5
   );
15 box1 = box_coord(3);
16 box2 = box_coord(1);
17 box3 = box_coord(4)-box_coord(3);
18 box4 = box_coord(2)-box_coord(1);
19 for cc = 1:size(obj_contour,2)
20     if cc < size(obj_contour,2)
21         plot([obj_contour(1,cc),
           obj_contour(1,cc+1)]+box_coord(3),
           [obj_contour(2,cc),
           obj_contour(2,cc+1)]+box_coord(1),
           'r','linewidth',4);
22     else
23        plot([obj_contour(1,cc),
           obj_contour(1,1)]+box_coord(3),
           [obj_contour(2,cc),
          obj_contour(2,1)]+box_coord(1),
          'r','linewidth',4); end
24     end
```

Kode Sumber 4.1 Implementasi proses *wrapping*

4.2.1.2 Implementasi Pengolahan Data *Input*

Implementasi dari pengolahan data *input* merupakan proses untuk mengolah citra Caltech 101 agar dapat diklasifikasi secara optimal. *Input* dari program ini berupa citra Caltech dengan rata-rata ukurannya 300 x 200 piksel. Implementasi ini merupakan implementasi *pseudocode* pada subbab 3.4. Implementasi dalam matlab ditunjukkan pada Kode Sumber 4.2.

```
1  nclass = 5;
2  nData = 100;
3  hasil = dir('Training_100Data');
4  annot = dir('TrainingAnn_100Data');
5  antC=1;
6  for  x=3:size(annot,1)
7       namefolAn=strcat('TrainingAnn_100Data
       \',annot(x).name);
8       tempnameAn=dir(namefolAn);
9       targetAn=zeros(nclass,1);
10      targetAn(str2double(annot(x).name))=1;
11      for y=3:size(tempnameAn,1)
12           fileAnnot=strcat(namefolAn,
             '\',tempnameAn(y).name);
13           dTrainAn(antC).fileAnnot=fileAnnot;
14           antC=antC+1;
15      end
16 End
17 c=1;
18 dataTrainX=zeros(100,19600);
19 dataTrainY=zeros(100,5);
20 for x=3:size(hasil,1)
21     namefol=strcat('Training_100Data\
       ',hasil(x).name);
22     tempname = dir(namefol);
```

Kode Sumber 4.2 Implementasi Pengolahan Data *Input* (bagian pertama)

```
23      target=zeros(nclass,1);
24      target(str2double(hasil(x).name))=1;
25      for y=3:size(tempname,1)
26          file=strcat(namefol,'\',tempname(y)
            .name);
27          dTrain(c).file = file;
28          c=c+1;
        end
29  end
30  vec=1;
31  for y=1:nData
32      X = imread(file);
33      file = dTrain(y).file;
34      fileAnnot = dTrainAn(y).fileAnnot;
35      [box1,box2,box3,box4]=
        show_annotation(file,fileAnnot);
36      imgAnot=imcrop(X,[box1,box2,box3,box4]);
37      I = rgb2gray(imgAnot);
38      I = imresize(I,[140 140]);
39      IVec = reshape(I',1,19600);
40      dataTrainX(vec,:)=IVec;
41      dataTrainY(vec,:)=target';
42      vec=vec+1;
43  end
44  save('Caltech101X','dataTrainX');
45  save('Caltech101Y','dataTrainY');
```

Kode Sumber 4.3 Implementasi Pengolahan Data *Input* (bagian kedua)

4.2.2 Implementasi Proses *Training*

Pada bagian ini akan dijelaskan implementasi dari proses *training* jaringan konvolusi. Implementasi ini bertujuan untuk menghasilkan bobot dan bias yang baru. *Input* untuk program ini adalah berupa parameter untuk jaringan konvolusi, ukuran data citra dan labelnya serta masukan yang terkahir berupa *options gradient*. Pada proses *training* terdapat beberapa tahap proses untuk memproleh bobot dan bias yang baru, diantaranya proses

feedforward, proses *backpropagation* dan proses *gradient*. Implementasi ini merupakan implementasi *pseudocode* pada subbab 3.6. Implementasi dalam matlab untuk proses *training* ditunjukkan pada Kode Sumber 4.4

```
1  m = size(x, 3);
2  numbatches = m / opts.batchsize;
3  if rem(numbatches, 1) ~= 0
4     error('numbatches not integer');
5  end
6  net.rL = [];
7  for i = 1 : opts.numepochs
8      disp(['epoch ' num2str(i) '/' num2str
       (opts.numepochs)]);
9      Tic;
10     kk = randperm(m);
11     for l = 1 : numbatches
12         batch_x = x(:, :, kk((l - 1) * opts
           .batchsize + 1 : l * opts
           .batchsize));
13         batch_y = y(:,    kk((l - 1) * opts
           .batchsize + 1 : l * opts
           .batchsize));
14         net = cnnff(net, batch_x);
15         net = cnnbp(net, batch_y);
16         net = cnnapplygrads(net, opts);
17         if isempty(net.rL)
18            net.rL(1) = net.L;
19         end
20         net.rL(end + 1) = 0.99 * net.rL(end)
           + 0.01 * net.L;
21     end
22     toc;
23 end
```

Kode Sumber 4.4 Implementasi Proses *Training*

4.2.2.1 Proses *feedforward*

Implementasi fungsi *feedforward* bertujuan untuk melakukan klasifikasi pada data citra. *Input* untuk program ini adalah berupa parameter untuk Jaringan Konvolusi dan citra berupa vektor. Implementasi ini merupakan implementasi *pseudocode* pada subbab 3.6.1. Implementasi ditunjukkan pada Kode Sumber 4.5

```
1  n = numel(net.layers);
2  net.layers{1}.a{1} = x;
3  inputmaps = 1;
4  for l = 2 : n
5      if strcmp(net.layers{l}.type, 'c')
6          for j = 1 : net.layers{l}.outputmaps
7              z = zeros(size(net.layers{l-1}.a{1})-
               [net.layers{l}.kernelsize-1,net.
               layers{l}.kernelsize - 1, 0]);
8                  for i = 1 : inputmaps
9                      z = z + convn(net.layers{l-1}
                       .a{i}, net.layers{l}.k{i}{j},
                       'valid');
10                 end
11                 net.layers{l}.a{j} = sigm(z+net.
                   layers{l}.b{j});
12             end
13             inputmaps = net.layers{l}.outputmaps;
14             elseif strcmp(net.layers{l}.type,'s')
15                     for j = 1 : inputmaps
16                         z = convn(net.layers{l-1}
                             .a{j},ones(net.layers{l}
                             .scale)/(net.layers{l}
                             .scale^2), 'valid');
17                         net.layers{l}.a{j} = z(1:
                           net.layers{l}.scale:end,1:
                           net.layers{l}.scale : end,
                           :);
```

Kode Sumber 4.5 Implementasi proses *feedforward* (bagian pertama)

```
18                  end
19              end
20          end
21          net.fv = [];
22          for j = 1 : numel(net.layers{n}.a)
23              sa = size(net.layers{n}.a{j});
24              net.fv = [net.fv; reshape(net.layers{n}
                .a{j}, sa(1) * sa(2), sa(3))];
25          end
26          net.o = sigm(net.ffW * net.fv + repmat
                (net.ffb, 1, size(net.fv, 2)));
```

Kode Sumber 4.6 Implementasi proses *feedforward* (bagian kedua)

4.2.2.2 Proses *back propagation*

Implementasi fungsi *backpropagation* bertujuan untuk memperoleh men-*trace* kesalahan dari setiap lapisan pada proses *feedforward* mulai dari lapisan *output* sampai pada lapisan pertama. *Input* untuk program ini adalah nilai jaringan hasil proses dari *feedforward*. Implementasi ini merupakan implementasi *pseudocode* pada subbab 3.6.2. Implementasi dalam matlab untuk proses *backpropagation* ditunjukkan pada Kode Sumber 4.7.

```
1   n = numel(net.layers);
2   net.e = net.o - y;
3   net.L = 1/2*sum(net.e(:).^ 2)/size(net.e,2);
4   net.od = net.e .* (net.o .* (1 - net.o));
5   net.fvd = (net.ffW' * net.od);
6   if strcmp(net.layers{n}.type, 'c')
7      net.fvd = net.fvd .*(net.fv.*(1-net.fv));
8   end
9   sa = size(net.layers{n}.a{1});
10  fvnum = sa(1) * sa(2);
11  for j = 1 : numel(net.layers{n}.a)
```

Kode Sumber 4.7 Implementasi proses *backpropagation* (bagian pertama)

```
12      net.layers{n}.d{j} = reshape(net.fvd(((j
        - 1) * fvnum + 1) : j * fvnum, :),sa(1),
        sa(2), sa(3));
13  end
14  for l = (n - 1) : -1 : 1
15      if strcmp(net.layers{l}.type, 'c')
16          for j = 1 : numel(net.layers{l}.a)
17              net.layers{l}.d{j}=net.layers{l}
                .a{j} .* (1 - net.layers{l}.a{j})
                .*(expand(net.layers{l + 1}.d{j},
                [net.layers{l + 1}.scale
                net.layers{l + 1}.scale 1]) /
                net.layers{l + 1}.scale ^ 2);
18          end
19      elseif strcmp(net.layers{l}.type,'s')
20              for i = 1:numel(net.layers{l}.a)
21                  z = zeros(size(net.layers{l}
                    .a{1}));
22                  for j = 1:numel(net.layers{l
                        + 1}.a)
23                      z = z+convn(net.layers{l
                        + 1}.d{j}, rot180(net
                        .layers{l + 1}.k{i}{j}),
                        'full');
24                  end
25                  net.layers{l}.d{i} = z;
26              end
27      end
28  end
29  for l = 2 : n
30      if strcmp(net.layers{l}.type, 'c')
31          for j = 1 : numel(net.layers{l}.a)
32              for i = 1:numel(net.layers{l-
                    1}.a)
```

Kode Sumber 4.8 Implementasi proses *Backpropagation* (bagian kedua)

```
33                  net.layers{l}.dk{i}{j} =
                    convn(flipall(net.layers{l -
                    1}.a{i}), net.layers{l}.d{j},
                    'valid') / size(net.layers{l}
                    .d{j}, 3);
34              end
35              net.layers{l}.db{j} = sum(net
                .layers{l}.d{j}(:)) /
                size(net.layers{l}.d{j}, 3);
36          end
37      end
38  end
39  net.dffW = net.od*(net.fv)'/size(net.od,2);
40  net.dffb = mean(net.od, 2);
41  function X = rot180(X)
42          X = flipdim(flipdim(X, 1), 2);
43  end
```

Kode Sumber 4.9 Implementasi Proses *Backpropagation* (bagian ketiga)

4.2.2.3 Proses Perhitungan *Gradient* untuk CNN

Implementasi dari fungsi *gradient* bertujuan untuk memperoleh nilai bobot dan bias yang baru dari hasil proses *training*. *Input* untuk program ini adalah jaringan dari hasil proses *back propagation* beserta *options gradient* yang telah ditentukan pada program utama. Implementasi ini merupakan implementasi *pseudocode* pada subbab 3.6.3. Implementasi dalam matlab untuk fungsi *gradient* ditunjukkan pada Kode Sumber 4.10.

```
1   for l = 2 : numel(net.layers)
2       if strcmp(net.layers{l}.type, 'c')
3           for j = 1 : numel(net.layers{l}.a)
4               for ii = 1 : numel(net.layers{l -
                    1}.a)
```

Kode Sumber 4.10 Implementasi proses Perhitungan *Gradient* (bagian pertama)

```
                    net.layers{l}.k{ii}{j} =
                    net.layers{l}.k{ii}{j} -
                    opts.alpha * net.layers{l}.
                    dk{ii}{j};
                end
                net.layers{l}.b{j} =
                net.layers{l}.b{j} - opts.alpha *
                net.layers{l}.db{j};
            end
        end
    end
    net.ffW = net.ffW - opts.alpha * net.dffW;
    net.ffb = net.ffb - opts.alpha * net.dffb;
```

Kode Sumber 4.11 Implementasi proses Perhitungan *Gradient* (bagian kedua)

4.2.3 Implementasi Proses *Testing*

Pada bagian ini akan dijelaskan implementasi dari klasifikasi citra Caltech 101. Proses *testing* memiliki proses yang sama dengan proses *training* namun tidak disertai dengan tahap *backpropagation* dan perhitungan *gradient*. *Input* dari program ini berupa parameter untuk Jaringan Konvolusi, ukuran citra dan label dari citra tersebut. Hasil akhir dari implementasi ini berupa akurasi dan data citra yang tidak sesuai labelnya. Implementasi ini merupakan implementasi *pseudocode* pada subbab 3.7. Implementasi dalam matlab untuk proses *testing* ditunjukkan pada Kode Sumber 4.12.

```
net = cnnff(net, x);
[~, h] = max(net.o);
[~, a] = max(y);
bad = find(h ~= a);
er = numel(bad) / size(y, 2);
```

Kode Sumber 4.12 Implementasi proses Perhitungan *Testing*

BAB V
HASIL UJI COBA DAN EVALUASI

Pada bab ini akan dijelaskan uji coba yang dilakukan pada aplikasi yang telah dikerjakan serta analisa dari uji coba yang telah dilakukan. Pembahasan pengujian meliputi lingkungan uji coba, skenario uji coba serta analisis setiap pengujian.

5.1 Lingkungan Uji Coba

Lingkungan uji coba menjelaskan lingkungan yang digunakan untuk menguji implementasi pembuatan klasifikasi citra Caltech 101 pada Tugas Akhir ini. Lingkungan uji coba meliputi perangkat keras dan perangkat lunak yang dijelaskan sebagai berikut:

1. Perangkat keras
    a. Prosesor: Intel(R) Core(TM)2 Duo CPU T6600 @ 2.20GHz 2.20GHz
    b. Memory(RAM): 2 GB.
    c. Tipe sistem: 32-bit sistem operasi.
2. Perangkat lunak
    a. Sistem operasi: Windows 7 Ultimate.
    b. Perangkat pengembang: MATLAB 8.1.0 (R2013a).

5.2 Uji Coba Kebenaran Citra

Pada uji coba ini akan dilakukan pengujian terhadap kebenaran dan kesesuaian dari metode praproses dan *convolutional neural network* pada sistem yang telah dibuat. Uji coba kebenaran bertujuan untuk menunjukkan bahwa program telah dapat berjalan sebagaimana seharusnya.

5.2.1 Skenario Uji Coba Kebenaran Klasifikasi Citra Objek dengan Tingkat Confusion yang Berbeda

Data yang digunakan pada uji coba terdiri dari sebelas kategori, lima kategori pertama dilakukan untuk uji coba objek

yang memiliki golongan yang sama. Tingkat *confusion* dari lima kategori tersebut rata-rata 20%, 30% dan 60%. Kelima kategori tersebut adalah Emu, Flamingo, Ibis, Pigeon, Rooster. Sedangkan enam kategori sisanya digunakan untuk uji coba objek dengan jenis yang sama. Tingkat *confusion* dari keenam kategori tersebut rata-rata 10% sampai 90%. Keenam kategori tersebut adalah Faces, Faces_easy, Cougar_body, Cougar_face, Crocodile dan Crocodile_head.

Uji coba dimulai dengan mengubah citra dalam ruang RGB menjadi citra vektor dengan menggunakan proses pengolahan data *input*. Citra hasil klasifikasi dibandingkan dengan data *training* basis data Caltech 101 untuk diuji kebenarannya. Uji kebenaran citra hasil klasifikasi dilakukan dengan evaluasi nilai *error* dan akurasi menggunakan *confusion matrix* yang ditampilkan pada Tabel 5.1.

Tabel 5.1 Confusion Matrix untuk Klasifikasi Citra Objek

Prediksi	Kenyataan	
	Hasil Testing	Hasil Training
Hasil Testing	TP	FP
Hasil Training	FN	TN

Berdasarkan *confusion matrix* pada Tabel 5.1, TP diperoleh ketika suatu piksel pada citra hasil *testing* dan citra hasil klasifikasi merupakan kelas yang sama. FP diperoleh ketika suatu piksel pada citra hasil *testing* merupakan latar belakang objek, namun piksel hasil klasifikasi merupakan citra objek. FN diperoleh ketika suatu piksel pada citra hasil *testing* adalah citra objek, namun piksel hasil klasifikasi adalah latar belakang objek. Sedangkan TN diperoleh ketika suatu piksel pada citra hasil *testing* dan citra hasil klasifikasi merupakan latar belakang objek.

Error diperoleh dengan membandingkan jumlah piksel yang merupkan FP dan FN dengan jumlah semua piksel seperti yang ditunjukkan pada Persamaan 5.1.

$$error = \frac{FP + FN}{TP + FP + FN + TN} \quad (5.1)$$

Nilai *error* yang dihasilkan dari Persamaan 5.1 berbanding terbalik dengan nilai kebenaran. Semakin kecil *error* maka semakin tinggi nilai kebenaran, sebaliknya ketika nilai *error* semakin besar maka nilai kebenaran semakin rendah. Selain *error*, dapat pula digunakan akurasi yang ditunjukkan pada Persamaan 5.2. Akurasi berbanding lurus dengan nilai kebenaran, dimana semakin tinggi akurasi maka semakin tinggi nilai kebenaran.

$$akurasi = \frac{TP + TN}{TP + FP + FN + TN} \quad (5.2)$$

5.3 Uji Coba Klasifikasi Citra Objek Caltech 101 dengan Tingkat Confusion Berbeda

Pada uji coba ini akan dilakukan pengujian terhadap kebenaran dan kesesuaian metode klasifikasi citra objek pada sistem yang telah dibuat. Uji coba kebenaran bertujuan untuk menunjukkan bahwa program telah dapat berjalan sebagaimana seharusnya.

5.3.1 Data Uji Coba Kebenaran Klasifikasi Citra Objek Caltech 101 dengan Tingkat Confusion Berbeda

Data yang digunakan pada tahap uji coba kebenaran hasil klasifikasi berupa vektor citra objek beserta label kelas yang diperoleh dari 150 buah citra objek untuk golongan yang sama dan 80 buah citra objek untuk jenis objek yang sama pada basis data Caltech 101. Pemilihan data uji coba berdasarkan tingkat kesulitan dalam mengenali objek tersebut baik berdasarkan tingkat *confusion* maupun jenis kategori objek tersebut.

Tingkat *confusion* dari basis data Caltech 101 ditentukan menggunakan metode *benchmark* dari jurnal Grauman et. Al [12] dan algoritma *correspondence* dari Alexander C.Berg et. Al [14].

Berdasarkan penelitian dari H. Zhang et. Al [13] yang telah dilakukan diperoleh *confusion rate* dari setiap kategori citra pada basis data Caltech 101.

Untuk melakukan klasifikasi, terlebih dahulu dibuat suatu data *testing* dan data *training*. Data *testing* pada uji coba ini menggunakan data dari basis data Caltech 101 sejumlah 50 data untuk golongan yang sama dan 20 sampai 60 data untuk objek yang sama. Sedangkan data *training* akan diambil sejumlah 100 data untuk golongan yang sama dan 20 sampai 60 data untuk objek yang sama. Label kelas antara data *training* dan data *testing* memiliki label kelas yang sama dan berasal dari basis data Caltech 101.

Pembuatan data *training* dan data *testing* dilakukan secara acak dengan menggunakan *cross validation* pada data yang telah dipilih sebelumnya. Pengacakan data dilakukan sebanyak tiga kali untuk memperoleh hasil akurasi yang subjektif.

Uji coba kebenaran klasifikasi citra objek akan menggunakan tiga jenis data *training* yang telah ditentukan sebelumnya, yaitu 100 data *training* dengan *confusion* rendah, 100 data *training* dengan *confusion* sedang dan 100 data *training* dengan *confusion* tinggi. Data-data yang digunakan pada ketiga jenis data *training* tersebut akan dijelaskan pada Tabel 5.2 sampai Tabel 5.6.

Tabel 5.2 Daftar Citra Testing dan Training Jenis Unggas (bagian pertama)

No.	Nama Citra	Kelas	Recognition Rate
1.		Emu	26.087%

Tabel 5.3 Daftar Citra Testing dan Training Jenis Unggas (bagian kedua)

2.		Flamingo	21.622%
3.		Ibis	22.000%
4.		Pigeon	33.333%
5.		Rooster	68.421%

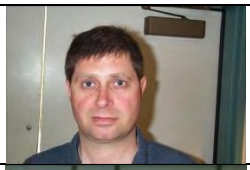
Tabel 5.4 Daftar Citra Testing Kategori Cougar

No.	Nama Citra	Kelas	Recognition Rate
1.		Cougar Body	11.765%
2.		Cougar Face	25.641%

Tabel 5.5 Daftar Citra Testing Kategori Crocodile

No.	Nama Citra	Kelas	Recognition Rate
1.		Crocodile	20.000%
2.		Crocodile Head	9.524%

Tabel 5.6 Daftar Citra Testing Kategori Faces

No.	Nama Citra	Kelas	Recognition Rate
1.		Faces	60.000%
2.		Faces Easy	94.321%

5.3.2 Uji Coba Kebenaran Klasifikasi Citra Objek dengan Golongan Unggas

Uji coba berikut ini menggunakan 100 data sebagai data *training*, sisanya sebanyak 50 data lainnya digunakan sebagai data *testing*. Pelatihan jaringan konvolusi dilakukan dalam 100 iterasi dengan *learning rate* sebesar 0,1. Uji coba dilakukan dengan data *training* dan *testing* yang diacak sebanyak tiga kali untuk memperoleh hasil klasifikasi yang subjektif.

Parameter yang digunakan pada jaringan konvolusi sebagai berikut: lapisan pertama terdapat 2 unit, lapisan kedua dan lapisan ketiga terdapat 16 *map*, lapisan keempat dan kelima terdapat 128 *map*, terakhir lapisan *output* sebanyak 5 unit.

Setelah pelatihan, dilakukan uji coba dengan 50 data. Hasil uji coba dapat dilihat pada Tabel 5.7. Dari sebanyak 50 data citra objek pada golongan unggas yang diuji cobakan, 50 data berhasil dikenali. Prosentase keberhasilan dari uji coba ini sebesar

20%. Lama proses *training* dan *testing* sebesar 9.770,45 detik. Hasil uji coba ini dapat dilihat pada Tabel 5.7

Tabel 5.7 Hasil Uji Coba Klasifikasi pada Golongan Unggas

Kategori	Berhasil		Gagal	
	Jumlah Data	Prosentase (%)	Jumlah Data	Prosentase (%)
Emu	0	0	10	100
Flamingo	0	0	10	100
Ibis	0	0	10	100
Pigeon	0	0	10	100
Rooster	10	100	0	0
Hasil Klasifikasi	10	20	40	80
Nilai Error Training				
Rata-Rata Min	0.1224			
Rata-Rata Max	0,9043			
Rata- Rata Waktu (detik)				
Training	9.754,17			
Testing	16,28			

5.3.3 Uji Coba Kebenaran Klasifikasi Citra Objek pada Kategori Cougar dengan 60 Data Training

Uji coba berikut ini menggunakan 60 data sebagai data *training*, sisanya sebanyak 20 data lainnya digunakan sebagai data *testing*. Pelatihan jaringan konvolusi dilakukan dalam 100 iterasi dengan *learning rate* sebesar 0,1. Uji coba dilakukan dengan data *training* dan *testing* yang diacak sebanyak tiga kali untuk memperoleh hasil klasifikasi yang subjektif.

Parameter yang digunakan pada jaringan konvolusi sebagai berikut: lapisan pertama terdapat 2 unit, lapisan kedua dan lapisan ketiga terdapat 16 *map*, lapisan keempat dan kelima terdapat 128 *map*, terakhir lapisan *output* sebanyak 5 unit.

Setelah pelatihan, dilakukan uji coba dengan 20 data. Hasil uji coba dapat dilihat pada Tabel 5.8. Dari sebanyak 20 data citra objek pada kategori Cougar yang diuji cobakan, 20 data berhasil dikenali. Prosentase keberhasilan dari uji coba ini sebesar 50%. Lama proses *training* dan *testing* sebesar 6.125,265 deitk. Hasil uji coba ini dapat dilihat pada Tabel 5.8

Tabel 5.8 Hasil Uji Coba Klasifikasi pada Kategori Cougar dengan Training 60 Data

Kategori	**Berhasil**		**Gagal**	
	Jumlah Data	Prosentase (%)	Jumlah Data	Prosentase (%)
Cougar Body	10	100	0	0
Cougar Face	0	0	10	100
Hasil Klasifikasi	10	50	10	50
Nilai Error Training				
Rata-Rata Min	0.0171			
Rata-Rata Max	0,3458			
Rata- Rata Waktu (detik)				
Training	6.115,075			
Testing	10,19			

5.3.4 Uji Coba Kebenaran Klasifikasi Citra Objek pada Kategori Cougar dengan 20 Data Training

Uji coba berikut ini menggunakan 20 data sebagai data *training*, sisanya sebanyak 60 data lainnya digunakan sebagai data *testing*. Pelatihan jaringan konvolusi dilakukan dalam 100 iterasi dengan *learning rate* sebesar 0,1. Uji coba dilakukan dengan data *training* dan *testing* yang diacak sebanyak tiga kali untuk memperoleh hasil klasifikasi yang subjektif.

Parameter yang digunakan pada jaringan konvolusi sebagai berikut: lapisan pertama terdapat 2 unit, lapisan kedua

dan lapisan ketiga terdapat 16 *map*, lapisan keempat dan kelima terdapat 128 *map*, terakhir lapisan *output* sebanyak 5 unit.

Setelah pelatihan, dilakukan uji coba dengan 60 data. Hasil uji coba dapat dilihat pada Tabel 5.9. Dari sebanyak 60 data citra objek pada kategori Cougar yang diuji cobakan, 60 data berhasil dikenali. Prosentase keberhasilan dari uji coba ini sebesar 50%. Lama proses *training* dan *testing* sebesar 2.217,28 detik. Hasil uji coba ini dapat dilihat pada Tabel 5.9.

Tabel 5.9 Hasil Uji Coba Klasifikasi pada Kategori Cougar dengan Training 20 Data

Kategori	**Berhasil**		**Gagal**	
	Jumlah Data	Prosentase (%)	Jumlah Data	Prosentase (%)
Cougar Body	30	100	0	0
Cougar Face	0	0	30	100
Hasil Klasifikasi	30	50	30	50
Nilai Error Training				
Min	0.1278			
Max	0.3457			
Rata- Rata Waktu (detik)				
Training	2.199,97			
Testing	17,31			

5.3.5 Uji Coba Kebenaran Klasifikasi Citra Objek pada Kategori Crocodile dengan 60 Data Training

Uji coba berikut ini menggunakan 60 data sebagai data *training*, sisanya sebanyak 20 data lainnya digunakan sebagai data *testing*. Pelatihan jaringan konvolusi dilakukan dalam 60 iterasi dengan *learning rate* sebesar 0,1. Uji coba dilakukan dengan data *training* dan *testing* yang diacak sebanyak tiga kali untuk memperoleh hasil klasifikasi yang subjektif.

Parameter yang digunakan pada jaringan konvolusi sebagai berikut: lapisan pertama terdapat 2 unit, lapisan kedua dan lapisan ketiga terdapat 16 *map*, lapisan keempat dan kelima terdapat 128 *map*, terakhir lapisan *output* sebanyak 5 unit.
Setelah pelatihan, dilakukan uji coba dengan 20 data. Hasil uji coba dapat dilihat pada Tabel 5.10. Dari sebanyak 20 data citra objek pada kategori Crocodile yang diuji cobakan, 20 data berhasil dikenali. Prosentase keberhasilan dari uji coba ini sebesar 50%. Lama proses *training* dan *testing* sebesar 2.289,085 detik. Hasil uji coba ini dapat dilihat pada Tabel 5.10

Tabel 5.10 Hasil Uji Coba Klasifikasi pada Kategori Crocodile dengan Training 60 Data

Kategori	**Berhasil**		**Gagal**	
	Jumlah Data	Prosentase (%)	Jumlah Data	Prosentase (%)
Crocodile	0	0	10	100
Crocodile Head	10	100	0	0
Hasil Klasifikasi	10	50	10	50
Nilai Error Training				
Min	0.1278			
Max	0.3458			
Rata- Rata Waktu (detik)				
Training	2.262,97			
Testing	26,115			

5.3.6 Uji Coba Kebenaran Klasifikasi Citra Objek pada Kategori Crocodile dengan 20 Data Training

Uji coba berikut ini menggunakan 20 data sebagai data *training*, sisanya sebanyak 60 data lainnya digunakan sebagai data *testing*. Pelatihan jaringan konvolusi dilakukan dalam 100 iterasi dengan *learning rate* sebesar 0,1. Uji coba dilakukan

dengan data *training* dan *testing* yang diacak sebanyak tiga kali untuk memperoleh hasil klasifikasi yang subjektif.

Parameter yang digunakan pada jaringan konvolusi sebagai berikut: lapisan pertama terdapat 2 unit, lapisan kedua dan lapisan ketiga terdapat 16 *map*, lapisan keempat dan kelima terdapat 128 *map*, terakhir lapisan *output* sebanyak 5 unit.
Setelah pelatihan, dilakukan uji coba dengan 60 data. Hasil uji coba dapat dilihat pada Tabel 5.11. Dari sebanyak 60 data citra objek pada kategori Cougar yang diuji cobakan, 60 data berhasil dikenali. Prosentase keberhasilan dari uji coba ini sebesar 50%. Lama proses *training* dan *testing* sebesar 2.997,99 detik. Hasil uji coba ini dapat dilihat pada Tabel 5.11.

Tabel 5.11 Hasil Uji Coba Coba Klasifikasi pada Kategori Crocodile dengan Training 20 Data

Kategori	**Berhasil**		**Gagal**	
	Jumlah Data	Prosentase (%)	Jumlah Data	Prosentase (%)
Crocodile	30	100	0	0
Crocodile Head	0	0	30	100
Hasil Klasifikasi	30	50	30	50
Nilai Error Training				
Min	0,12785			
Max	0,34585			
Rata- Rata Waktu (detik)				
Training	2.980,85			
Testing	17,14			

5.3.7 Uji Coba Kebenaran Klasifikasi Citra Objek pada Kategori Faces dengan 60 Data Training

Uji coba berikut ini menggunakan 60 data sebagai data *training*, sisanya sebanyak 20 data lainnya digunakan sebagai data *testing*. Pelatihan jaringan konvolusi dilakukan dalam 60

iterasi dengan *learning rate* sebesar 0,1. Uji coba dilakukan dengan data *training* dan *testing* yang diacak sebanyak tiga kali untuk memperoleh hasil klasifikasi yang subjektif.

Parameter yang digunakan pada jaringan konvolusi sebagai berikut: lapisan pertama terdapat 2 unit, lapisan kedua dan lapisan ketiga terdapat 16 *map*, lapisan keempat dan kelima terdapat 128 *map*, terakhir lapisan *output* sebanyak 5 unit.
Setelah pelatihan, dilakukan uji coba dengan 20 data. Hasil uji coba dapat dilihat pada Tabel 5.12. Dari sebanyak 20 data citra objek pada kategori Face yang diuji cobakan, 20 data berhasil dikenali. Prosentase keberhasilan dari uji coba ini sebesar 50%. Lama proses *training* dan *testing* sebesar 7.939,135 detik. Hasil uji coba ini dapat dilihat pada Tabel 5.12.

Tabel 5.12 Hasil Uji Coba Klasifikasi pada Kategori Face dengan Training 60 Data

Kategori	**Berhasil**		**Gagal**	
	Jumlah Data	Prosentase (%)	Jumlah Data	Prosentase (%)
Face	0	0	10	100
Face Easy	10	100	0	0
Hasil Klasifikasi	10	50	10	50
Nilai Error Training				
Min	0.0171			
Max	0.3460			
Rata- Rata Waktu (detik)				
Training	7.932,43			
Testing	6,705			

5.3.8 Uji Coba Kebenaran Klasifikasi Citra Objek pada Kategori Faces dengan 20 Data Training

Uji coba berikut ini menggunakan 20 data sebagai data *training*, sisanya sebanyak 60 data lainnya digunakan sebagai

data *testing*. Pelatihan jaringan konvolusi dilakukan dalam 100 iterasi dengan *learning rate* sebesar 0,1. Uji coba dilakukan dengan data *training* dan *testing* yang diacak sebanyak tiga kali untuk memperoleh hasil klasifikasi yang subjektif.

Parameter yang digunakan pada jaringan konvolusi sebagai berikut: lapisan pertama terdapat 2 unit, lapisan kedua dan lapisan ketiga terdapat 16 *map*, lapisan keempat dan kelima terdapat 128 *map*, terakhir lapisan *output* sebanyak 5 unit.
Setelah pelatihan, dilakukan uji coba dengan 60 data. Hasil uji coba dapat dilihat pada Tabel 5.13. Dari sebanyak 60 data citra objek pada kategori Face yang diuji cobakan, 60 data berhasil dikenali. Prosentase keberhasilan dari uji coba ini sebesar 50%. Lama proses *training* dan *testing* sebesar 2.033,24 detik. Hasil uji coba ini dapat dilihat pada Tabel 5.13

Tabel 5.13 Hasil Uji Coba Klasifikasi pada Kategori Face dengan Training 20 Data

Kategori	**Berhasil**		**Gagal**	
	Jumlah Data	Prosentase (%)	Jumlah Data	Prosentase (%)
Face	30	100	0	0
Face Easy	0	0	30	100
Hasil Klasifikasi	30	50	30	50
Nilai Error Training				
Min	0.1279			
Max	0,34595			
Rata- Rata Waktu (detik)				
Training	2.018,6			
Testing	14,64			

BAB VI
KESIMPULAN DAN SARAN

Bab ini membahas mengenai kesimpulan yang dapat diambil dari hasil uji coba yang telah dilakukan sebagai jawaban dari rumusan masalah yang dikemukan. Selain kesimpulan, juga terdapat saran yang ditunjukkan untuk pengembangan perangkat lunak lebih lanjut.

6.1 Kesimpulan

Dari hasil uji coba yang telah dilakukan terhadap pembuatan program praproses dan klasifikasi citra objek dapat diambil kesimpulan sebagai berikut:

1. Metode praproses dan metode klasifikasi yang digunakan telah memiliki alur proses yang benar, terbukti dengan nilai akurasi yang lebih baik dibandingkan dengan metode klasifikasi sebelumnya.
2. Metode praproses dan metode klasifikasi dengan menggunakan *Convolutional Neural Network* relatif handal untuk menentukan kebenaran dari klasifikasi citra objek. Hal ini terbukti dengan hasil akurasi sebesar. Untuk perbandingan citra hasil klasifikasi dari basis data Caltech 101.
3. Untuk data *training*, perubahan tingkat *confusion* tidak mempengaruhi hasil akurasi. Hal ini membuktikan bahwa klasifikasi menggunakan metode CNN relatif handal terhadap perubahan parameter yang dilakukan.
4. Dengan menggunakan data *training* yang baik dan optimal, maka subset dari data *training* tersebut juga akan menghasilkan klasifikasi yang baik.
5. Waktu eksekusi yang dibutuhkan untuk sebuah citra dapat dideteksi sebagai label kelas yang sesuai antara 3 jam sampai 5 jam.

6.2 Saran

Saran yang diberikan untuk pengembangan aplikasi klasifikasi citra objek pada Tugas Akhir ini antara lain:

1. Berdasarkan hasil uji coba, memerlukan waktu yang cukup lama untuk memperoleh hasil klasifikasi. Oleh karena itu dapat dilakukan pengoptimalan struktur yang mampu mempercepat proses *training* agar klasifikasi cepat diperoleh.

DAFTAR PUSTAKA

[1] Y. LeCun, "Handwritten Digit Recognition with a Back-Propagation Network," 1990.

[2] Wikipedia, "Feature Learning," [Online]. Available: http://en.wikipedia.org/wiki/Feature_learning.

[3] A.Coates, H.Lee and A.Y. Ng, "An Analysis of Single-Layer Networks in Unsupervised Feature Learning," 2011.

[4] K. Fukushima, "Neocognitron: A Self-Organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position," *Biological Cybernetics,*1980.

[5] A.Karpathy, "CS231n Convolutional Neural Network for Visual Recognition, "Stanford University, [Online].Available: http://cs231m.github.io/.

[6] LISA Lab, "Deep Learning Tutorial," [Online]. Available: http://deeplearning.net/tutorial/contents.html.

[7] D. Stathakis, "How Many Hidden Layers And Nodes?," *International Journal of Remote Sensing,* 2008.

[8] Stanford University, "An Introduction to Convolutional Neural Network," Vision Imaging Science and Technology Lab, Stanford University, [Online]. Available: http://white.stanford.edu/teach/index.php/An_Introduction_to_Convolutional_Neural_Networks.

[9] A. Ng, J. Ngiam, C. Yu Foo, Y. Mai and C. Suen, "Unsupervised Feature Learning and Deep Learning Tutorial," Stanford University, [Online]. Available: http://ufdl.stanford.edu/wiki/index.php/UFLDL_Tutorial.

[10] J. T. Springenberg, A. Dosovitskiy, T. Brox and M. Riedmiller, "Striving For Simplicity: The All Convolutional Net," *ICLR 2015*, 2015.

[11] C.Olah, "Neural Networks, Manifolds, and Topology," [Online]. Available: http://cola.github.io/posts/2014-03-NN-Manifolds-Topology/.

[12] Kristen Grauman, Trevor Darrell, "The Pyramid Match Kernel: Discriminative Classification with Sets of Image Features," Cambridge, MA, USA, 2005.

[13] Hao Zhang, Alexander C.Berg, Michael Maire, Jitendra Malik, "SVM-KNN: Discriminative Nearest Neighbor Classfication for Visual Category Recognition," University of California, Berkeley, *CA 94720*.

[14] Alexander C. Berg, Tamara L.Berg, Jitendra Malik, "Shape Matching and Object Recognition using Low Distorion Correspondences," U.C.Berkeley.

BIODATA PENULIS

I Wayan Suartika Eka Putra, lahir di Denpasar, pada tanggal 10 Juni 1991. Penulis menempuh pendidikan mulai dari SD Saraswati 6 Denpasar (1997-2003), SMP Negeri 3 Denpasar (2003-2006), SMA Negeri 3 Denpasar (2006-2009) dan S1 Teknik Informatika ITS (2009-2016). Selama masa kuliah, penulis aktif dalam organisasi Himpunan Mahasiswa Teknik Computer (HMTC) dan Tim Pembina Kerohanian Hindu ITS (TPKH-ITS). Diantaranya adalah menjadi ketua panitia PKTI, Staff RisTek pada HMTC dan Staff PSDM pada TPKH-ITS. Penulis juga aktif dalam kegiatan GEMASTIK 2012 pada cabang kompetisi Desain Web.

Selama kuliah di Teknik Informatika ITS, penulis mengambil bidang minat Komputasi Cerdas Visual (KCV) dengan fokus studi pada bidang *Machine Learning*. Penulis pernah menjadi asisten pada PIKTI-ITS. Komunikasi dengan penulis dapat dilakukan melalui email: **wayan.suartika777@gmail.com**.