

**IMPLEMENTASI DEEP LEARNING MENGGUNAKAN
CONVOLUTIONAL NEURAL NETWORK UNTUK KLASIFIKASI ALAT
TULIS**

(Studi Kasus : Gambar alat tulis (*Ballpoint*, Penghapus dan Penggaris))

TUGAS AKHIR

Diajukan Sebagai Salah Satu Syarat Untuk Memperoleh Gelar Sarjana
Program Studi Statistika



Jimmy Pujoseno

14 611 160

**JURUSAN STATISTIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS ISLAM INDONESIA
YOGYAKARTA
2018**

HALAMAN PERSETUJUAN PEMBIMBING

TUGAS AKHIR

Judul : Implementasi *Deep Learning* Menggunakan
Convolutional Neural Network untuk Klasifikasi Alat

Tulis

Nama Mahasiswa : Jimmy Pujoseno

Nomor Mahasiswa : 14 611 160

**TUGAS AKHIR INI TELAH DIPERIKSA DAN DISETUJUI UNTUK
DIUJIKAN**

Yogyakarta, 14 Februari 2018

الجامعة الإسلامية
الاندونيسية

Pembimbing



(Dr. RB. Fajriya Hakim, S.Si., M.Si.)

HALAMAN PENGESAHAN

TUGAS AKHIR

IMPLEMENTASI DEEP LEARNING MENGGUNAKAN CONVOLUTIONAL NEURAL NETWORK UNTUK KLASIFIKASI ALAT

TULIS ISLAM

Nama Mahasiswa : Jimmy Pujoseno

Nomor Mahasiswa : 14 611 160

TUGAS AKHIR INI TELAH DIUJIKAN

PADA TANGGAL 21 MARET 2018

Nama Penguji

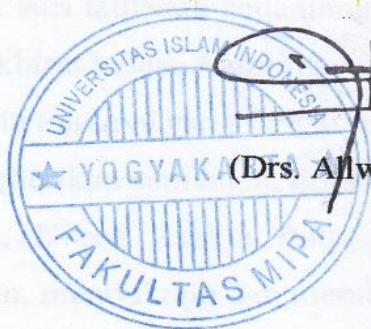
1. Dr. RB. Fajriya Hakim, S.Si., M.Si.
2. Tuti Purwaningsih, S.Stat., M.Si.
3. Annisa Uswatun Khasanah, S.T., M.Sc

Tanda tangan



Mengetahui,

Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam



(Drs. Aliwar, M.Sc., Ph.D.)

KATA PENGANTAR



Assalamu'alaikum Warahmatullaahi Wabarakaatuh

Alhamdulillah *alhamdulillah* *alhamdulillah*, Puji Syukur senantiasa saya panjatkan kehadirat Allah SWT yang telah melimpahkan rahmat, hidayah, dan nikmatnya yang tak terhingga, sehingga penulis dapat menyelesaikan tugas akhir yang berjudul **“Implementasi Deep Learning Menggunakan Convolutional Neural Network Untuk Klasifikasi Alat Tulis (Studi Kasus : Gambar Alat Tulis (Ballpoint, Penghapus, dan Penggaris))”** sebagai salah satu persyaratan yang harus dipenuhi dalam menyelesaikan jenjang strata satu di Jurusan Statistika, Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Islam Indonesia. Shalawat serta salam semoga selalu tercurah kepada Nabi Muhammad SAW serta para sahabat dan pengikutnya yang senantiasa menjaga keimanan dan keislamannya hingga akhir hayatnya.

Penyelesaian tugas akhir ini tidak terlepas dari dukungan, bantuan, arahan, dan bimbingan dari berbagai pihak. Untuk itu pada kesempatan ini penulis ingin mengucapkan terima kasih yang sebesar-besarnya kepada:

1. Allah SWT yang telah memberikan rahmat, nikmat sehat dan kesempatan sehingga penyusun dapat melaksanakan penelitian hingga menyusun Tugas Akhir dengan baik.
2. Rasulullah Muhammad SAW, atas segala cintanya kepada umat manusia dan merupakan suri tauladan sepanjang zaman, semoga kita mendapatkan syafa'at beliau di akhirat kelak. Aamiin.
3. Kedua orang tua saya yang sangat saya cintai, Sujiman dan Nety Herawaty yang selalu memberikan semangat, do'a dan dukungan disetiap langkah saya.
4. Adik saya, Niken Zazkia Putri, serta Keluarga Besar saya yang selalu mendo'akan, mendukung dan memberikan semangat kepada saya.

5. Drs. Allwar. M.Sc, Ph.D. selaku Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam.
6. Bapak Dr. RB. Fajriya Hakim, M.Si. selaku Ketua Program Studi Statistika sekaligus sebagai pembimbing saya, yang telah banyak memberikan dukungan dan masukan yang membangun serta selalu bersedia meluangkan waktunya untuk berkonsultasi dan memberikan arahan yang sangat inspiratif.
7. Seluruh staf pengajar Program Studi Statistika Universitas Islam Indonesia yang telah memberikan bekal ilmu kepada penulis, sehingga penulis dapat menyelesaikan tugas akhir ini.
8. Keluarga besar IKS(Ikatan Keluarga Statistika), sebagai Organisasi yang membawahi mahasiswa statistika FMIPA UII, terimakasih atas kebersamaan, kekeluargaan, kekompakan, keceriaan dan pelajaran berharga lainnya.
9. Sahabat seperjuangan yaitu Riza Indriani Rakhmalia, Erene Fajrila, Nur Hidayah, dan Inayatus S, yang telah berjuang bersama selama di bangku perkuliahan dan senantiasa menjadi pengingat dalam hal kebaikan. Semoga ukhuwah kita tetap terjaga dan selalu diridhoi Allah SWT.
10. Sahabat seperjuangan "Project 2 Minggu Kelar" yaitu Tiara Shafira, dan Sendyka Cakra Pradana yang telah berjuang bersama selama di bangku perkuliahan dan senantiasa menjadi pengingat dalam hal kebaikan. Semoga ukhuwah kita tetap terjaga dan selalu diridhoi Allah SWT.
11. Teman-teman bimbingan TA yang sudah sama-sama berjuang, saling mengingatkan dan memberi motivasi serta dorongan untuk menyelesaikan Tugas Akhir ini.
12. Teman-teman Statistika UII Angkatan 2014 yang bersama-sama menjadi pejuang gelar S.Stat dan Toga UII, terimakasih semangatnya.
13. Pihak-pihak lain yang mungkin penulis belum sebutkan, yang telah membantu dalam penyusunan tugas akhir ini.

Demikian Tugas Akhir ini, penulis mengucapkan terima kasih kepada semua pihak yang telah memberikan bantuan baik moril maupun materil sehingga tugas akhir ini dapat diselesaikan. Penulis menyadari bahwa tugas akhir ini masih jauh dari kata sempurna dan masih banyak kekurangan. Hal tersebut dikarenakan

keterbatasan ilmu dan pengetahuan yang dimiliki penulis semata. Oleh karena itu penulis mengharapkan kritik dan saran dari pembaca untuk menyempurnakan penulisan laporan ini. Semoga Tugas Akhir ini dapat memberikan manfaat bagi penulis khususnya dan umumnya bagi semua pihak yang membutuhkan. Akhir kata, semoga Allah SWT senantiasa melimpahkan rahmat serta hidayah-Nya kepada kita semua, Amin amin ya robbal ‘alamiin.

Wassalamu’alaikum Warahmatullaahi Wabarakaatuh

Yogyakarta, 14 Februari 2018

Penulis

DAFTAR ISI

HALAMAN JUDUL	i
HALAMAN PERSETUJUAN PEMBIMBING	ii
HALAMAN PENGESAHAN	iii
KATA PENGANTAR	iv
DAFTAR ISI	vii
DAFTAR TABEL	x
DAFTAR GAMBAR	xi
DAFTAR LAMPIRAN	xiii
PERNYATAAN	xiv
INTISARI	xv
ABSTRACT	xvi
BAB I PENDAHULUAN	1
1.1. Latar Belakang Masalah	1
1.2. Rumusan Masalah	3
1.3. Batasan Masalah	3
1.4. Tujuan Penelitian	4
1.5. Manfaat Penelitian	4
1.6. Sistematika Penulisan	4
BAB II TINJAUAN PUSTAKA	6
BAB III LANDASAN TEORI	13
3.1. Citra	13
3.2. Alat Tulis	14
3.3. <i>Web Crawler</i>	15
3.4. <i>Jquery</i>	15
3.5. Kecerdasan Buatan	16
3.6. <i>Machine Learning</i>	18
3.7. <i>Deep Learning</i>	19
3.8. <i>Artificial Neural Network</i>	20
3.9. Konsep <i>Neural Network</i>	21

3.9.1.	Jaringan Syaraf Pada Otak Manusia	21
3.9.2.	Struktur <i>Neural Network</i>	22
3.9.3.	Arsitektur Jaringan Syaraf Tiruan	23
3.9.4.	<i>Activation Function</i>	24
3.9.5.	Pembelajaran <i>Backpropagation</i>	25
3.9.6.	Bias	27
3.9.7.	<i>Learning Rate</i>	28
3.9.8.	Fungsi <i>Error</i>	28
3.10.	<i>Convolution Neural Network</i>	28
3.10.1.	Operasi Konvolusi	29
3.10.2.	Arsitektur Jaringan CNN	30
BAB IV	METODOLOGI PENELITIAN	34
4.1.	Populasi dan Sampel.....	34
4.2.	Variabel dan Definisi Operasional Variabel.....	34
4.3.	Jenis dan Sumber Data.....	34
4.4.	Metode Analisis Data	34
4.5.	Tahapan Penelitian.....	35
BAB V	ANALISIS DAN PEMBAHASAN	37
5.1.	Pembuatan <i>Dataset</i>	37
5.1.1.	Program <i>Javascript</i>	38
5.1.2.	Program <i>Python</i>	40
5.2.	<i>Dataset</i> Untuk Pelatihan CNN.....	42
5.3.	Preprocessing CNN	44
5.4.	Pembuatan Model CNN.....	50
5.5.	Evaluasi dan Prediksi Hasil Train.....	55
5.6.	Perbandingan Parameter CNN.....	56
5.6.1.	Penentuan ukuran filter.....	57
5.6.2.	Pengaruh Jumlah Data Terhadap Akuras	57
5.6.3.	Pengaruh Skenario Pelatihan Data	58
BAB VI	PENUTUP	60
6.1.	Kesimpulan.....	60

6.2. Saran	60
DAFTAR PUSTAKA	62
LAMPIRAN	64

DAFTAR TABEL

Tabel 2.1	Perbandingan dengan penelitian terdahulu	11
Tabel 4.1	Definisi operasional variabel	34
Tabel 5.1	<i>Import Packages</i>	40
Tabel 5.2	Membuat <i>Argument</i>	40
Tabel 5.3	Perulangan <i>Download URL</i>	41
Tabel 5.4	Perulangan <i>Load Image</i>	42
Tabel 5.5	Dataset Untuk Pelatihan.....	43
Tabel 5.6	Mempersiapkan Gambar.....	44
Tabel 5.7	<i>Resize dan Combine</i>	46
Tabel 5.8	<i>Reorder Dimension</i>	49
Tabel 5.9	<i>Create Model CNN</i>	50
Tabel 5.10	<i>Training Model</i>	53
Tabel 5.11	<i>Evaluate and Prediction</i>	55
Tabel 5.12	Penentuan Ukuran Filter	57
Tabel 5.13	Pengaruh Jumlah Data Terhadap Akurasi.....	57
Tabel 5.14	Pengaruh Skenario Pelatihan Data.....	58

DAFTAR GAMBAR

Gambar 3.1	Ilustrasi <i>Neuron</i> Dengan Model Matematisnya.....	20
Gambar 3.2	Skema <i>Neuron</i>	21
Gambar 3.3	Struktur <i>Artificial Neural Network</i>	22
Gambar 3.4	Arsitektur Jaringan Syaraf Tiruan	23
Gambar 3.5	<i>Linear Function</i>	24
Gambar 3.6	<i>Sigmoid and Tanh Function (Non-Linear)</i>	25
Gambar 3.7	<i>ReLU (Non-Linear)</i>	25
Gambar 3.8	Jaringan <i>Backpropagation</i>	26
Gambar 3.9	Arsitektur CNN	30
Gambar 3.10	<i>Image RGB</i>	30
Gambar 3.11	<i>Feature Map</i>	31
Gambar 3.12	<i>Max Pooling</i>	33
Gambar 4.1	Diafram Alir Penelitian	36
Gambar 5.1	<i>Google Image Search</i> Penghapus	38
Gambar 5.2	<i>Download URL</i>	39
Gambar 5.3	<i>Output hiddenElement</i>	39
Gambar 5.4	<i>Output Dataset</i> untuk <i>Training</i>	44
Gambar 5.5	<i>Properties</i> Gambar	45
Gambar 5.6	<i>Plot</i> Gambar <i>Test</i>	46
Gambar 5.7	Struktur <i>Train</i>	47
Gambar 5.8	Struktur <i>Train</i>	47
Gambar 5.9	<i>Train</i> Hasil <i>Combine</i>	48
Gambar 5.10	<i>Test</i> Hasil <i>Combine</i>	48
Gambar 5.11	<i>Output</i> Hasil <i>Reorder Dimension</i>	49
Gambar 5.12	<i>Summary Model</i>	51
Gambar 5.13	Proses Pembuatan Model.....	52
Gambar 5.14	<i>Plot Model</i>	54
Gambar 5.15	<i>Output Tensorboard</i> Akurasi <i>Train</i>	54
Gambar 5.16	<i>Output Tensorboard</i> Akurasi <i>Test</i>	55

Gambar 5.17 <i>Output Evaluasi Model</i>	56
Gambar 5.18 <i>Output Evaluasi Model</i>	56
Gambar 5.19 <i>Output Prediksi Data Baru</i>	58

DAFTAR LAMPIRAN

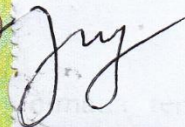
Lampiran 1	<i>Script Javascript Create Dataset.....</i>	64
Lampiran 2	<i>Script Python Crawling Data</i>	65
Lampiran 3	<i>Script R Persiapan Gambar.....</i>	67
Lampiran 4	<i>Script R Pembuatan dan Training Model</i>	69
Lampiran 5	<i>Script R Evaluate and Prediction</i>	71

PERNYATAAN

Dengan ini saya menyatakan bahwa dalam Tugas Akhir ini tidak terdapat karya yang sebelumnya pernah diajukan untuk memperoleh gelar keserjanaan di suatu Perguruan Tinggi dan sepanjang pengetahuan saya juga tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang diacu dalam naskah ini dan disebutkan dalam daftar pustaka.

Yogyakarta, 14 februari 2018




Jimmy Pujoseno

IMPLEMENTASI DEEP LEARNING MENGGUNAKAN CONVOLUTIONAL NEURAL NETWORK UNTUK KLASIFIKASI ALAT TULIS

(Studi Kasus : Gambar alat tulis (*Ballpoint*, Penghapus dan Penggaris)

Jimmy Pujoseno

Program Studi Statistika Fakultas Matematika dan Ilmu Pengetahuan Alam
Universitas Islam Indonesia

INTISARI

Alat tulis merupakan alat yang digunakan untuk menulis dimana terdapat beberapa jenis alat tulis diantaranya penghapus, ballpoint penggaris pensil dan lain - lain. Alat tulis tidak terlepas dari kehidupan sehari – hari, karena setiap hari pasti digunakan baik disekolah, kantor dan tempat lainnya. Kebutuhan alat tulis yang banyak tentu membuat banyak toko yang menjual alat tulis. Dengan banyaknya toko-toko yang menjual alat tulis tentu dibutuhkan alat untuk membedakan atau mengklasifikasikannya supaya mempermudah dalam penjualan alat tulis tersebut. Seiring berkembangnya teknologi terutama pada bidang kecerdasan buatan (AI) memiliki manfaat yang besar. Salah satu cabang dari AI yaitu Machine Learning(ML) dimana mengajari mesin layaknya dapat berfikir seperti manusia. Salah satu pendekatan ML yaitu deep learning, dimana mampu melakukan pembelajaran lebih dalam. Deep learning yang digunakan untuk pengenalan objek adalah *Convolution Neural Network* (CNN). Dengan melakukan klasifikasi objek diharapkan mampu mempermudah toko dalam menjual alat tulis. Pada penelitian ini dilakukan pengenalan objek *ballpoint*, penghapus, dan penggaris menggunakan *software Rstudio* dan *package keras* dengan *back-end Tensorflow*. Data *sample* yang digunakan sebanyak 300 gambar merupakan hasil *crawling* dari *google image*. Untuk melakukan klasifikasi perlu dilakukan training data yang akan membentuk sebuah model. Model tersebut digunakan untuk klasifikasi gambar train dan test untuk 3 kategori. Akurasi yang dihasilkan model untuk data train adalah 100% dan untuk data test sebesar 95%. Kemudian dilakukan percobaan untuk 3 data baru , didapatkan 1 gambar masuk kategori yang salah.

Kata kunci : *Artificial Intelegence, Convolution Neural Network, Keras, Alat Tulis*

IMPLEMENTASI DEEP LEARNING MENGGUNAKAN CONVOLUTIONAL NEURAL NETWORK UNTUK KLASIFIKASI ALAT TULIS

(Studi Kasus : Gambar alat tulis (*Ballpoint*, Penghapus dan Penggaris))

Jimmy Pujoseno

Department of Statisticks, Faculty of Mathematics and Natural Sciences

Islamic University of Indonesia

ABSTRACT

Stationery is a tool used to write where there are several types of stationery such as erasers, ruler, ballpoint etc. Stationery can not be separated from everyday life, because every day must be used both in school, office and other places. Needs a lot of stationery certainly makes a lot of stores that sell stationery. With the many shops that sell stationery certainly needed a classify tool to facilitate in the sale of the stationery. Along with the development of technology, especially in the field of intelligence for having great benefits. One of the AI is Machine Learning (ML) which teaches machine can think like human. One approach of ML is deep learning, which is able to do deeper learning. One of the deep learning methods that used for object recognition is Convolution Neural Network (CNN). By doing the object classification is expected to facilitate the store in selling stationery. In this research, we do object recognition of ballpoint, eraser, and ruler using Rstudio software and keras package with Tensorflow back-end. The sample of data that used are 300 images from crawling of google image. The classification needs to train the data to form a model. The model is used for training and testing 3 categories of data. The accuracy of data train is 100% and for the 95% data test. Then experiment for 3 new data, gave a result 1 picture in wrong category.

Keywords : *Artificial Intelegence, Convolution Neural Network, Keras, Stationery*

BAB I

PENDAHULUAN

1.1. Latar Belakang Masalah

Alat tulis merupakan alat yang digunakan untuk menulis dimana terdapat beberapa jenis alat tulis diantaranya *ballpoint* , pensil, penghapus, penggaris dan lain-lain. Alat tulis telah digunakan sejak jaman dahulu dimana dimulai dari menggunakan batu sebagai alat tulis kemudian terus berkembang menggunakan bulu dan sekarang lebih canggih lagi. Alat tulis tidak terlepas dari kehidupan sehari – hari, karena setiap hari pasti digunakan baik disekolah, kantor dan tempat lainnya.

Kebutuhan alat tulis yang banyak tentu membuat banyak toko yang menjual alat tulis. Dengan banyaknya toko-toko yang menjual alat tulis tentu dibutuhkan alat untuk membedakan atau mengklasifikasikannya supaya mempermudah dalam penjualan alat tulis tersebut. Sehingga nantinya pada tingkat yang paling tinggi pembeli dipermudah dengan tanpa harus mengantri untuk membeli alat tulis tersebut. Salah satu cara untuk menyelesaikan masalah ini yaitu memanfaatkan teknologi yang sangat berkembang saat ini.

Teknologi adalah keseluruhan sarana untuk menyediakan barang-barang yang diperlukan bagi kelangsungan dan kenyamanan hidup manusia. Pada era ini menggunakan teknologi dalam membantu menyelesaikan pekerjaan menjadi keharusan dalam kehidupan sehari – hari. Salah satu kemajuan dari teknologi yaitu komputer saat ini sangat dibutuhkan, misalnya di supermarket, rumah makan, dan perusahaan.

Komputer berasal dari bahasa Inggris yaitu, *computer* yang berasal dari kata *compute* yang artinya menghitung. Jadi secara bahasa, *computer* adalah alat hitung. Komputer diciptakan sebagai alat bantu hitung yang memudahkan manusia dalam melakukan penghitungan. Namun dalam perkembangan teknologi, komputer dapat digunakan dalam membantu berbagai macam pekerjaan manusia, seperti dalam bidang akuntansi, dan administrasi (Wikipedia.org, 2018). Komputer saat ini telah jauh berbeda dengan komputer beberapa tahun lalu dimana saat ini peran komputer sangat besar dalam membantu kehidupan manusia. Dengan

perkembangan ilmu pengetahuan salah satunya kecerdasan buatan membuat komputer memiliki manfaat jauh lebih besar dari sebelumnya. Sejak awal, komputer telah berhasil melakukan berbagai tugas yang sulit bagi manusia. Berikan permasalahan gravitasi, perancangan roket, simulasi reaksi kimia, dan berbagai hal lain yang terlihat mustahil, komputer dapat menyelesaikannya dengan baik dan sangat cepat. Namun anehnya, ketika berusaha menyelesaikan masalah yang sangat mudah bagi manusia, seperti mengenali objek dalam foto dan memahami percakapan manusia, komputer seakan tak berdaya. Setidaknya sampai era *Artificial Intelligence* dimulai.

Menurut H. A. Simon (1987) Kecerdasan buatan (*artificial intelligence*) merupakan kawasan penelitian, aplikasi dan instruksi yang terkait dengan pemrograman komputer untuk melakukan sesuatu hal yang dalam pandangan manusia adalah cerdas. Dengan demikian diharapkan komputer bisa menirukan beberapa fungsi otak manusia, seperti berpikir dan belajar. Sistem kecerdasan komputer ini bisa dilatih atau dilakukan pembelajaran terhadap komputer itu sendiri atau biasa disebut dengan *machine learning*.

Machine learning (ML), bagian dari kecerdasan buatan (*artificial intelligence*), merupakan metode untuk mengoptimalkan performa dari sistem dengan mempelajari data sampel atau data histori (Alpaydin, 2009) *Machine learning* (ML) adalah cabang dari AI yang meliputi mesin yang dapat menguraikan data dan mampu belajar sendiri melalui proses *training*. Dari proses training didapatkan model yang dapat digunakan untuk menjawab dari inputan data sesuai dengan hasil *training model*, dari data dapat dibuat dua model yang terkenal yaitu regresi dan klasifikasi. ML banyak digunakan di berbagai bisnis dewasa ini karena sangat efisien bila digunakan di berbagai bidang seperti pengenalan suara, objek, dan wajah, penerjemahan, dan tugas-tugas lainnya. Contohnya adalah *DeepMind Google*. Pendekatan ML yang banyak digunakan pada saat ini adalah *deep learning*, yang memiliki konsep utama akan mempelajari fitur yang terdapat pada data baru ketika menemukan kemiripan fitur pada data yang lama, yaitu data yang telah dipejarinya. Sekarang sudah ditemukan beberapa model *deep learning*, di antaranya adalah *Deep Convolutional Neural Networks* (DCNN) untuk *image classification*,

Deep Belief Network - Deep Neural Network (DBN - DNN) untuk *speech recognition*, *Recurrent Neural Network (RNN)* untuk penerjemahan bahasa, *Query-Oriented Deep Extraction (QODE)* yang berbasiskan *Restricted Boltzmann Machine (RBM)* untuk peringkasan multi dokumen, *Conditional Restricted Boltzmann Machine (RBM)* untuk *Drug-Target Interaction (DTI) prediction* dan *Deep Belief Network (DBN)* untuk *time-series data prediction*.

Klasifikasi menurut Han (2006) adalah teknik data mining yang digunakan untuk memprediksi kategori dari objek yang belum memiliki kategori. CNN merupakan salah satu metode yang digunakan untuk klasifikasi gambar dimana pada metode CNN, terinspirasi oleh korteks mamalia visual sel sederhana dan kompleks. Model ini dapat mengurangi sejumlah parameter bebas dan dapat menangani deformasi gambar input seperti translasi, rotasi dan skala (Zufar dan Budi, 2016). Berdasarkan penjelasan kelebihan CNN tersebut, dapat diambil kesimpulan bahwa CNN memiliki kemampuan klasifikasi yang diperuntukkan untuk data gambar. Dari permasalahan diatas peneliti melakukan penelitian tentang implementasi deep learning menggunakan CNN pada klasifikasi alat tulis yang bertujuan agar dengan kemajuan teknologi manusia juga tidak lupa pada alat-alat tulis yang juga mempunyai banyak manfaat.

1.2. Rumusan Masalah

Berdasarkan permasalahan diatas, adapun permasalahan yang akan dikaji dalam penelitian ini adalah sebagai berikut:

1. Bagaimana penerapan metode *Convolutional Neural Network* pada klasifikasi alat tulis ?
2. Bagaimana tingkat akurasi model *Convolutional Neural Network* yang digunakan ?
3. Bagaimana hasil klasifikasi dari data baru menggunakan model *Convolutional Neural Network* ?

1.3. Batasan Masalah

Adapun batasan masalah yang digunakan peneliti agar pembahasan dalam penelitian ini tidak menyimpang dari pokok pembahasan. Maka peneliti memiliki batasan masalah sebagai berikut:

1. Model *deep learning* yang diimplementasikan yaitu *Convolutional Neural Network* (CNN).
2. Data yang digunakan pada penelitian ini yaitu data hasil *crawling* dari *google image*
3. *Software* yang digunakan pada penelitian ini yaitu R dan *python*

1.4. Tujuan Penelitian

Tujuan dari penelitian ini adalah sebagai berikut:

1. Mengimplementasikan *deep learning* dengan metode CNN pada klasifikasi alat tulis
2. Seberapa besar tingkat akurasi model CNN untuk klasifikasi gambar pena, penghapus dan penggaris.
3. Mengetahui hasil klasifikasi dari data baru menggunakan model *Convolutional Neural Network*.

1.5. Manfaat Penelitian

Adapun manfaat dari penelitian ini yakni sebagai berikut:

1. Mengetahui penerapan *deep learning* dengan metode CNN pada klasifikasi gambar.
2. Komputer mampu untuk mengklasifikasi *ballpoint*, penghapus dan penggaris.
3. Mengingat kembali bentuk dari alat tulis

1.6. Sistematika Penulisan

Sistematika penulisan yang dipergunakan dalam penulisan tugas akhir ini dapat diuraikan sebagai berikut:

BAB I PENDAHULUAN

Pada bab ini akan dibahas tentang latar belakang masalah, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian dan sistematika penulisan.

BAB II TINJAUAN PUSTAKA

Bab ini memaparkan penelitian-penelitian terdahulu yang berhubungan dengan permasalahan yang diteliti dan menjadi acuan konseptual.

BAB III LANDASAN TEORI

Pada bab ini akan dibahas tentang teori-teori dan konsep yang berhubungan dengan penelitian yang dilakukan dan mendukung dalam pemecahan masalahnya. Selain itu, bab ini juga memuat teori-teori dalam pelaksanaan pengumpulan dan pengolahan data serta saat melakukan penganalisaan.

BAB IV METODOLOGI PENELITIAN

Bab ini memaparkan populasi dan sampel, variabel penelitian, jenis dan sumber data, metode analisis data, dan tahapan penelitian.

BAB V ANALISIS DAN PEMBAHASAN

Pada bab ini akan dibahas mengenai analisa yang dilakukan terhadap hasil pengumpulan, pengolahan dan analisa data yang diperoleh dari hasil penelitian.

BAB VI PENUTUP

Pada bab ini akan dibahas mengenai kesimpulan yang diperoleh dari hasil penelitian dan analisa data yang telah dilakukan serta saran-saran yang dapat diterapkan dari hasil pengolahan data yang dapat menjadi masukan yang berguna kedepannya.

BAB II

TINJAUAN PUSTAKA

Penelitian terdahulu sebagai kajian bagi penulis sangat penting untuk mengetahui hubungan antara penelitian yang dilakukan sebelumnya dengan penelitian yang penulis lakukan saat ini serta dapat menghindari adanya duplikasi. Hal ini bermanfaat untuk menunjukkan bahwa penelitian yang dilakukan mempunyai arti penting sehingga dapat diketahui kontribusi penelitian terhadap ilmu pengetahuan.

Penelitian mengenai klasifikasi gambar sebelumnya telah dilakukan oleh Farsiah dkk (2013). Penelitian ini membahas tentang algoritma penentuan kategori dari gambar berwarna menggunakan *K-Nearest Neighbour* (KNN) dan *Support Vector Machine* (SVM). Pengujian dilakukan *color dataset* dan fitur terbaik dipilih menggunakan metode pemilihan fitur (*feature selection*). Hasil uji coba menunjukkan bahwa akurasi dari metode KNN sama baiknya dengan metode SVM.

Penelitian mengenai CNN sebelumnya telah dilakukan oleh Gummesson (2016). Tujuan utama dari penelitian ini adalah untuk mengotomatisasi klasifikasi dengan menggunakan *Convolutional Neural Networks* (CNN). Dengan diperkenalkannya Jaringan *Neural Convolutional* bidang pengenalan pola terputus. Cara klasik merancang fitur buatan tangan untuk klasifikasi sangat berbeda untuk membiarkan komputer menentukan sendiri fitur mana yang penting, pendekatan baru diaktifkan oleh CNN. Ini bersama dengan hasil terobosan pada patokan gambar telah membuat CNN metode yang digunakan dengan baik dalam pengenalan pola. Dalam penelitian ini, sebuah CNN dengan filter konvolusi kecil telah dilatih dari nol dengan menggunakan arus *gradien stochastic* dengan momentum. Tingkat kesalahan untuk CNN adalah 7,3%, yang secara signifikan lebih baik daripada pekerjaan sebelumnya yang menggunakan kumpulan data yang sama. Karena hasil yang baik diperoleh walaupun kumpulan data agak kecil, kesimpulannya adalah bahwa CNN adalah metode yang menjanjikan untuk masalah ini.

Penelitian mengenai CNN sebelumnya juga telah dilakukan oleh Gibert (2016). Penelitian ini menyajikan dua pendekatan baru dan terukur dengan menggunakan *Convolutional Neural Networks* (CNN) untuk menetapkan perangkat lunak jahat ke keluarga yang sesuai. Di satu sisi, pendekatan pertama memanfaatkan CNN untuk mempelajari hierarki fitur untuk membedakan sampel *malware* yang digambarkan sebagai gambar skala abu-abu. Di sisi lain, pendekatan kedua menggunakan arsitektur CNN yang diperkenalkan oleh YoonKim untuk mengklasifikasikan sampel *malware* sesuai instruksi x86 mereka. Metode yang diusulkan mencapai perbaikan sebesar 93,86% dan 98,56% berkenaan dengan tolok ukur probabilitas yang sama.

Penelitian mengenai CNN sebelumnya juga telah dilakukan oleh Serra (2017). Tujuan dari penelitian ini adalah untuk mengembangkan sistem *Face Recognition* yang lengkap untuk *GoldenSpear LLC*, sebuah perusahaan berbasis AI. Sistem yang dikembangkan menggunakan *Convolutional Neural Networks* untuk mengekstrak fitur wajah yang relevan. Fitur-fitur ini memungkinkan untuk membandingkan wajah di antara mereka dengan cara yang efisien. Sistem ini dapat dilatih untuk mengenali sekumpulan orang, dan belajar secara *on-line*, dengan mengintegrasikan orang-orang baru yang ia proses dan perbaiki keterpaparan orang-orang di dalamnya. Keakuratan yang ada di sekitar 100 orang berhasil mencapai 95%, dan karena terlalu banyak dengan jumlah orang dalam sistem.

Penelitian mengenai CNN sebelumnya juga telah dilakukan oleh Erlangung (2007). Penelitian ini menyajikan metode berbasis CNN untuk deteksi fitur wajah otomatis. Sistem yang diusulkan menggunakan prosedur hirarkis yang pertama kali secara kasar melokalisasi mata, hidung dan mulut dan kemudian menemukan hasilnya dengan mendeteksi 10 titik fitur wajah yang berbeda. Tingkat deteksi metode ini adalah 96% untuk database AR dan 87% untuk database BioID yang menoleransi kesalahan 10% dari jarak antar-okular. Akhirnya, diusulkan pendekatan pengenalan wajah baru berdasarkan arsitektur CNN spesifik yang mempelajari pemetaan non-linear ruang gambar ke dalam ruang sub-dimensi yang lebih rendah dimana kelas yang berbeda lebih mudah dipisahkan. Metode ini diterapkan ke beberapa database wajah publik dan memperoleh tingkat pengenalan

yang lebih baik dibandingkan dengan pendekatan pengenalan wajah klasik berdasarkan PCA atau LDA. Selain itu, sistem yang diusulkan sangat kuat terhadap kebisingan dan oklusi parsial. Penelitian ini juga menyajikan metode berbasis CNN untuk masalah klasifikasi klarifikasi biner pengenalan gender dengan gambar wajah dan mencapai akurasi mutakhir. Hasil yang disajikan dalam penelitian ini menunjukkan bahwa CNN bekerja dengan sangat baik pada berbagai tugas pemrosesan gambar wajah, seperti pelurusan wajah, deteksi fitur wajah dan pengenalan wajah dan dengan jelas menunjukkan bahwa teknik CNN adalah pendekatan serbaguna, efisien dan kuat untuk analisis citra wajah.

Penelitian mengenai *crawling images* telah dilakukan oleh Shrinivasacharya dan Sudhamani (2013). Penelitian ini menyajikan alat berbasis web yang mengumpulkan dan mengindeks sekelompok gambar web yang tersedia internet. Alat ini mengumpulkan kata kunci atau frase dari pengguna untuk mengambil gambar dari web. Kemudian kata kunci yang dikumpulkan ini diterapkan ke berbagai alat penelusuran umum seperti *Google*, *Yahoo*, *Bind* dll. Informasi halaman web yang dikumpulkan disimpan dalam file sementara sampai ukuran file 200KB dari server. Kemudian konten file ini akan dipindai dan ekstrak URL gambar dan itu dibandingkan dengan URL yang ada dalam database untuk menghindari duplikat unduhan. Gambar URL yang diekstrak diunduh dan akhirnya menyimpan gambar unik dan metadata yang sesuai seperti nama file, url, ukuran, dll di database. Akhirnya gambar yang dihasilkan ini digunakan dalam sistem Pengambilan Citra Berbasis Konten (CBIR) untuk mengekstrak gambar yang relevan yang dibutuhkan oleh klien dengan menggunakan konten gambar daripada informasi berbasis teks.

Penelitian mengenai klasifikasi telah dilakukan oleh Apriliya dkk (2015). Penelitian ini menyajikan klasifikasi dalam studi kasus penilaian status gizi balita menggunakan metode *Naïve Bayes Classifier*, asumsi independensi antar parameter perlu diperhitungkan. Independensi antar parameter dilihat dari korelasi antar parameter yang digunakan. Artikel ini membahas mengenai uji korelasi antar parameter dalam studi kasus penilaian status gizi menggunakan metode *Cosine Similarity*. Kemudian hasil uji korelasi tersebut dijadikan prosedur penentuan

model dalam metode *Naive Bayes Classifier*. Sehingga dapat diketahui model yang paling baik dalam penilaian status gizi menggunakan metode *Naive Bayes Classifier*. Penentuan model terbaik dilihat dari akurasi, kesederhanaan, waktu, dan akuisisi data pada model. Pada skenario data 60%:40%, model terbaik ditunjukkan oleh model yang terdiri dari parameter berat, bmi, dan umur, dengan akurasi sebesar 94.4%. Sedangkan pada skenario data 80%:20% model terbaik ditunjukkan pada model yang terdiri dari parameter berat, bmi, tinggi, umur, dan jenis kelamin, dengan akurasi 94,8%. Penelitian ini menunjukkan bahwa kolerasi parameter mempengaruhi hasil klasifikasi. Penggunaan parameter independen belum tentu menghasilkan akurasi yang maksimal. Bahkan, model terbaik yang dipilih terdiri dari parameter dependen.

Penelitian mengenai klasifikasi menggunakan CNN telah dilakukan juga oleh Rismiyati (2016). Proses yang dilakukan pada penelitian ini adalah pengumpulan data, proses *preprocessing*, klasifikasi dan pengujian. *Preprocessing* dilakukan dengan memotong *region of interest* (ROI) yang berisi salak saja. Klasifikasi dilakukan dengan CNN, dimana untuk mendapatkan akurasi yang terbaik parameterparameter yang ada harus diujikan. Terdapat dua model yang digunakan yaitu model dua kelas dan empat kelas. Untuk masing-masing model, nilai parameter optimal perlu dicari untuk mendapat akurasi terbaik. Selain itu percobaan juga dilakukan dengan menggunakan ukuran data latih yang beragam untuk melihat pengaruh jumlah data latih terhadap akurasi. Pengujian dilakukan dengan metode *stratified cross validation* untuk mengukur akurasi berdasar confusion matrix. Hasil penelitian menunjukkan bahwa akurasi terbaik untuk model dua kelas didapatkan dengan metode CNN dengan menggunakan learning rate 0.0001, satu lapisan konvolusi dengan jumlah filter lima belas dengan ukuran 3x3x3, dan jumlah neuron pada lapisan tersembunyi 100. Akurasi yang didapatkan adalah 81,5%. Model empat kelas mendapat akurasi 70,7% dengan dua lapisan konvolusi.

Beberapa tinjauan pustaka diatas merupakan referensi untuk mengembangkan penelitian ini. Dari hasil analisa berbagai penelitian yang sudah dilakukan terdahulu akan dilakukan riset tentang penerapan pengolahan citra untuk

klasifikasi gambar menggunakan metode *Deep Learning* yaitu *Convolution Neural Network*. Metode *deep learning* ini digunakan untuk memecahkan masalah klasifikasi gambar karena penggunaannya yang memiliki proses komputasi yang relatif cepat, sehingga penelitian ini dapat menghasilkan klasifikasi gambar yang akurat.

Tabel 2.1 merupakan tabel rangkuman perbandingan dengan penelitian sebelumnya yang berkaitan klasifikasi gambar khususnya metode *convolution neural network*.

Tabel 2.1 Perbandingan dengan penelitian terdahulu

No.	Penulis	Judul	Metode	Persamaan	Perbedaan
1.	Gummeson (2016)	<i>Prostate Cancer Classification using Convolutional Neural Networks</i>	<i>Convolution Neural Network</i>	Sama-sama menggunakan <i>Convolution Neural Network</i>	Penelitian terdahulu dilakukan untuk klasifikasi kanker prostat
2.	Gibert (2016)	<i>Convolutional Neural Networks for Malware Classification</i>	<i>Convolution Neural Network</i>	Sama-sama menggunakan <i>Convolution Neural Network</i>	Penelitian terdahulu dilakukan untuk klasifikasi <i>malware</i>
3.	Farsiah dkk. (2013)	Klasifikasi gambar berwarna menggunakan <i>K-Nearest Neighbor</i> dan <i>Support Vector Machine</i>	<i>K-Nearest Neighbor</i> dan <i>Support Vector Machine</i>	Sama-sama melakukan klasifikasi gambar	Penelitian terdahulu menggunakan metode <i>K-Nearest Neighbor</i> dan <i>Support Vector Machine</i>
4.	Serra (2017)	<i>Face Recognition using Deep Learning</i>	<i>Deep learning Convolution Neural Network</i>	Sama-sama menggunakan <i>Deep Learning Convolution Neural Network</i>	Penelitian terdahulu tentang <i>Face Recognition</i>
5.	Erlangung (2007)	<i>Face Image Analysis With Convolutional Neural Networks</i>	<i>Convolution Neural Network</i>	Sama-sama menggunakan <i>Convolution Neural Network</i>	Objek penelitian sebelumnya menggunakan gambar wajah.
6.	<i>Shrinivasacharya t dan Sudhamani</i> (2013)	<i>An Image Crawler For Content Based Image Retrieval System</i>	<i>Crawling Image</i>	Sama – sama <i>Crawling images</i>	Objek penelitian sebelumnya menggunakan banyak World Wide Web

7.	Apriliya dkk (2015)	<i>Penentuan Model Terbaik pada Metode Naive Bayes Classifier dalam Menentukan Status Gizi Balita dengan Mempertimbangkan Independensi Parameter</i>	<i>Naïve Bayes</i>	Sama – sama klasifikasi	Pada penelitian ini menggunakan Alat tulis sebagai objek
8.	Rismiyati (2016)	<i>Implementasi Convolutional Neural Network Untuk Sortasi Mutu Salak Ekspor Berbasis Citra Digital</i>	<i>Convolutional Neural Network</i>	Sama-sama menggunakan <i>Convolution Neural Network</i>	Pada penelitian sebelumnya tentang sortasi salak

BAB III

LANDASAN TEORI

3.1 Citra

Citra merupakan media informasi yang tersusun atas kombinasi warna. Suatu informasi citra dapat dianalisis dengan cara pengelompokan berdasarkan persamaan pada bentuk dimensi citra maupun warna. Citra yang mengandung banyak unsur warna dapat dikelompokkan ke dalam beberapa warna dasar tertentu. Bagian warna-warna pada citra yang memiliki kemiripan terhadap warna dasar yang ditentukan, dapat diarahkan untuk menjadi bagian pada kelompok warna dasar tersebut. Citra digital direpresentasikan sebagai fungsi dua dimensi $f(x,y)$, dengan nilai x dan y masing-masing merupakan koordinat spasial dan nilai fungsi pada setiap titik (x,y) merupakan intensitas citra atau tingkat keabuan pada titik koordinat tersebut (Isyi, 2017). Proses pengolahan citra pada penelitian ini terdiri dari beberapa tahapan, yang masing-masing prosesnya saling terkait.

Ciri merupakan suatu tanda yang khas, yang membedakan antara satu dengan yang lain. Citra juga memiliki ciri yang dapat membedakannya dengan citra yang lain. Masing-masing ciri citra didapatkan dari proses ekstraksi ciri. Ciri – ciri dasar dari citra sebagai berikut:

1. Warna

Ciri warna suatu citra dapat dinyatakan dalam bentuk histogram dari citra tersebut yang dituliskan dengan: $H(r,g,b)$, dengan (r,g,b) adalah jumlah munculnya pasangan warna r (*red*), g (*green*) dan b (*blue*) tertentu.

2. Bentuk

Ciri bentuk suatu citra dapat ditentukan oleh tepi (sketsa), atau besaran moment dari suatu citra. Pemakaian besaran moment pada ciri bentuk ini banyak digunakan orang dengan memanfaatkan nilai-nilai transformasi *fourier* dari citra. Proses yang dapat digunakan untuk menentukan ciri bentuk adalah deteksi tepi, *threshold*, segmentasi, dan perhitungan moment seperti (mean, median dan standard deviasi dari setiap lokal citra).

3. Tekstur

Tekstur merupakan karakteristik intrinsik dari suatu citra yang terkait dengan tingkat kekasaran (*roughness*), granularitas (*granulation*), dan keteraturan (*regularity*) susunan struktural piksel. Aspek tekstural dari sebuah citra dapat dimanfaatkan sebagai dasar dari segmentasi, klasifikasi, maupun interpretasi citra.

3.2 Alat Tulis

Alat tulis merupakan peralatan yang digunakan menulis selain itu juga ada alat tambahan yang digunakan untuk melengkapi kegiatan menulis berikut menurut (Wikipedia.org, 2018) alat – alat tulis :

1. Pensil

Pensil biasanya sebutan alat tulis yang mengandung grafit sebagai pigmen. Grafit dibungkus oleh kayu, namun pada saat ini pembungkus grafit bisa berupa kertas keras maupun sejenis karet lentur. Selain itu saat ini pensil mekanik dipergunakan dengan luas.

2. Pena/ ballpoint

Awalnya dibentuk dari bulu angsa yang di potong meruncing bagian ujungnya dan sebelum menulis harus dicelupkan kedalam tinta terlebih dahulu. Pena modern saat ini tidak lagi terbuat dari bulu angsa, namun dapat terbuat dari plastik, *stainless steel*, kuningan dengan ujung lancip serta memiliki selongsong tempat penyimpanan tinta sehingga tidak diperlukan lagi pencelupan ke dalam tinta.

3. Spidol

Alat tulis yang memiliki ujung lunak untuk menulis. Spidol lebih sering digunakan untuk keperluan khusus atau menggambar. Memiliki tinta khusus yang mengandung alkohol disimpan dalam sejenis busa yang dapat menguap jika terpapar udara terus menerus.

4. Alat Tambahan

Selain alat tulis yang telah disebut di atas, terdapat pula penghapus yang gunanya untuk melakukan perbaikan terhadap kesalahan yang dilakukan dengan menggunakan pensil. Penyerut pensil digunakan untuk menajamkan ujung pensil, sehingga menghasilkan tulisan yang lebih rapi. Penggaris, selain digunakan sebagai alat ukur juga digunakan untuk membantu dalam pembuatan garis pada tulisan.

3.3 **Web Crawler**

Web Crawler adalah meng-*crawl* (merayapi) seluruh informasi suatu website yang biasanya digunakan untuk meng-index suatu *website*, pemeliharaan *website*, atau digunakan untuk memperoleh data khusus contohnya email. Dan hal ini juga dapat digunakan untuk memvalidasi *hyperlink* dan kode HTML. *Web Crawler* dimulai dengan me-list daftar URL yang akan dikunjungi, yang disebut dengan seed. *Web crawler* akan mengunjungi URL yang ada di daftar dan mengidentifikasi semua *hyperlink* di halaman tersebut serta menambahkannya kedalam daftar URL yang akan dikunjungi yang disebut *crawl frontier*. URL yang telah ada dikunjungi dan diambil informasi yang ada sesuai yang dibutuhkan. Dengan banyaknya jumlah URL yang mungkin di-*crawl* oleh *crawler server* yang membuatnya sulit untuk menghindari pengambilan konten yang sama. Misalkan protokol HTTP GET membuat kombinasi URL yang sangat banyak dan sedikit dari URL tersebut menghasilkan konten yang berbeda dan selebihnya menghasilkan konten yang sama untuk URL yang berbeda, inilah yang menimbulkan masalah bagi *crawler* agar bisa mengambil konten yang berbeda dari URL-URL tersebut. (Wikipedia.org, 2018)

3.4 **JQuery**

JQuery adalah salah satu *library javascript*. Dengan *JQuery*, kita dapat melakukan banyak hal yang tidak bisa dilakukan oleh HTML maupun CSS. Misalnya menampilkan artikel tanpa me-*reload* halaman, memunculkan *pop-up* di tengah – tengah halaman, menyembunyikan artikel jika di klik dan sebagainya. (Kun, 2010)

jQuery adalah *library* atau kumpulan kode *JavaScript* siap pakai. Keunggulan menggunakan *jQuery* dibandingkan dengan *javascript* standard, yaitu menyederhanakan kode *javascript* dengan cara memanggil fungsi – fungsi yang disediakan oleh *jquery*. *Javascript* sendiri merupakan bahasa *scripting* yang bekerja di sisi *client/browser* sehingga *website* jadi lebih interaktif. *jQuery* pertama kali di rilis tahun 2006 oleh *John Resig*. *jQuery* menjadi sangat populer hingga telah digunakan pada banyak *website* termasuk *website* – *website* kelas dunia seperti *Google*, *Amazon*, *Twitter*, *ESPN*, dan lain – lain. (Aloysius, 2011)

Jika disimpulkan secara umum berdasarkan teori – teori yang ada *jQuery* adalah *library javascript* yang berisi kode – kode *javascript* yang di sederhanakan yang tidak bisa dilakukan oleh HTML maupun CSS, penggunaan *jQuery* sendiri sangat mudah karena sudah dituliskan dan dapat dipakai langsung di halaman yang dapat memanggil kode *jQuery*, secara umum *jQuery* melakukan kerja keras koding *javascript* namun dalam versi yang lebih mudah dan sederhana.

3.5 Kecerdasan Buatan

Kecerdasan Buatan (bahasa Inggris: *Artificial Intelligence* atau AI) didefinisikan sebagai kecerdasan entitas ilmiah. Sistem seperti ini umumnya dianggap komputer. Kecerdasan diciptakan dan dimasukkan ke dalam suatu mesin (komputer) agar dapat melakukan pekerjaan seperti yang dapat dilakukan manusia. Beberapa macam bidang yang menggunakan kecerdasan buatan antara lain sistem pakar, permainan komputer (*games*), logika *fuzzy*, jaringan syaraf tiruan dan robotika.

Banyak hal yang kelihatannya sulit untuk kecerdasan manusia, tetapi untuk Informatika relatif tidak bermasalah. Seperti contoh: mentransformasikan persamaan, menyelesaikan persamaan integral, membuat permainan catur atau *Backgammon*. Di sisi lain, hal yang bagi manusia kelihatannya menuntut sedikit kecerdasan, sampai sekarang masih sulit untuk direalisasikan dalam Informatika. Seperti contoh: Pengenalan Obyek/Muka, bermain sepak bola.

Walaupun AI memiliki konotasi fiksi ilmiah yang kuat, AI membentuk cabang yang sangat penting pada ilmu komputer, berhubungan dengan perilaku,

pembelajaran dan adaptasi yang cerdas dalam sebuah mesin. Penelitian dalam AI menyangkut pembuatan mesin untuk mengotomatisasikan tugas-tugas yang membutuhkan perilaku cerdas. Termasuk contohnya adalah pengendalian, perencanaan dan penjadwalan, kemampuan untuk menjawab diagnosa dan pertanyaan pelanggan, serta pengenalan tulisan tangan, suara dan wajah. Hal-hal seperti itu telah menjadi disiplin ilmu tersendiri, yang memusatkan perhatian pada penyediaan solusi masalah kehidupan yang nyata. Sistem AI sekarang ini sering digunakan dalam bidang ekonomi, obat-obatan, teknik dan militer, seperti yang telah dibangun dalam beberapa aplikasi perangkat lunak komputer rumah dan video game. 'Kecerdasan buatan' ini bukan hanya ingin mengerti apa itu sistem kecerdasan, tapi juga mengkonstruksinya. (Wikipedia.org, 2018)

Kecerdasan Buatan berhubungan dengan pemanfaatan mesin untuk memecahkan persoalan yang rumit dengan cara yang lebih manusiawi. Hal ini biasanya dilakukan dengan mengikuti atau mencontoh karakteristik dan analogi berpikir dari kecerdasan atau Intelligensia manusia, dan menerapkannya sebagai algoritma yang dikenal oleh komputer. Dengan suatu pendekatan yang kurang lebih fleksibel dan efisien dapat diambil tergantung dari keperluan, yang mempengaruhi bagaimana wujud dari perilaku kecerdasan buatan. AI biasanya dihubungkan dengan Ilmu Komputer, akan tetapi juga terkait erat dengan bidang lain seperti Matematika, Psikologi, Pengamatan, Biologi, Filosofi, dan yang lainnya. Kemampuan untuk mengkombinasikan pengetahuan dari semua bidang ini pada akhirnya akan bermanfaat bagi kemajuan dalam upaya menciptakan suatu kecerdasan buatan.

Pengertian lain dari kecerdasan buatan menurut Rich dan Knight (1991) ialah sebuah studi tentang bagaimana membuat komputer melakukan hal-hal yang pada saat ini dapat dilakukan lebih baik oleh manusia. Pada awal diciptakannya, komputer hanya difungsikan sebagai alat hitung saja. Namun seiring dengan perkembangan zaman, maka peran komputer semakin mendominasi kehidupan manusia. Komputer tidak lagi hanya digunakan sebagai alat hitung, lebih dari itu, komputer diharapkan untuk dapat diberdayakan untuk mengerjakan segala sesuatu yang bisa dikerjakan oleh manusia.

Manusia bisa menjadi pandai dalam menyelesaikan segala permasalahan di dunia ini karena manusia mempunyai pengetahuan dan pengalaman. Pengetahuan diperoleh dari belajar. Semakin banyak bekal pengetahuan yang dimiliki oleh seseorang tentu saja diharapkan akan lebih mampu dalam menyelesaikan permasalahan. Namun bekal pengetahuan saja tidak cukup, manusia juga diberi akal untuk melakukan penalaran, mengambil kesimpulan berdasarkan pengetahuan dan pengalaman yang mereka miliki. Tanpa memiliki kemampuan untuk menalar dengan baik, manusia dengan segudang pengalaman dan pengetahuan tidak akan dapat menyelesaikan masalah dengan baik. Demikian pula dengan kemampuan menalar yang sangat baik, namun tanpa bekal pengetahuan dan pengalaman yang memadai, manusia juga tidak akan bisa menyelesaikan masalah dengan baik.

Agar komputer bisa bertindak seperti dan sebaik manusia, maka komputer juga harus diberi bekal pengetahuan dan mempunyai kemampuan untuk menalar. Untuk itu AI akan mencoba untuk memberikan beberapa metoda untuk membekali komputer dengan kedua komponen tersebut agar komputer bisa menjadi mesin pintar.

3.6 *Machine Learning*

Istilah *machine learning* pertama kali didefinisikan oleh Arthur Samuel ditahun 1959. Menurut Arthur Samuel, *machine learning* adalah salah satu bidang ilmu komputer yang memberikan kemampuan pembelajaran kepada komputer untuk mengetahui sesuatu tanpa pemrogram yang jelas. Menurut Mohri dkk (2012) *machine learning* dapat didefinisikan sebagai metode komputasi berdasarkan pengalaman untuk meningkatkan performa atau membuat prediksi yang akurat. Definisi pengalaman disini ialah informasi sebelumnya yang telah tersedia dan bisa dijadikan data pembelajar.

Dalam pembelajaran *machine learning*, terdapat beberapa skenario-skenario. Seperti:

1. *Supervised Learning*

Penggunaan skenario *supervised learning*, pembelajaran menggunakan masukan data pembelajaran yang telah diberi label. Setelah itu membuat prediksi dari data yang telah diberi label.

2. *Unsupervised Learning*

Penggunaan skenario *Unsupervised Learning*, pembelajaran menggunakan masukan data pembelajaran yang tidak diberi label. Setelah itu mencoba untuk mengelompokan data berdasarkan karakteristik-karakteristik yang ditemui.

3. *Reinforcement Learning*

Pada skenario *reinforcement learning* fase pembelajaran dan tes saling dicampur. Untuk mengumpulkan informasi pembelajar secara aktif dengan berinteraksi ke lingkungan sehingga untuk mendapatkan balasan untuk setiap aksi dari pembelajar.

Saat ini telah banyak pendekatan *machine learning* yang digunakan untuk deteksi *spam*, *Optical character recognition* (OCR), pengenalan wajah, deteksi penipuan *online*, NER (*Named Entity Recognition*), *Part-of-Speech Tagger*.

3.7 *Deep Learning*

Deep Learning merupakan salah satu bidang dari *Machine Learning* yang memanfaatkan jaringan syaraf tiruan untuk implementasi permasalahan dengan dataset yang besar. Teknik *Deep Learning* memberikan arsitektur yang sangat kuat untuk *Supervised Learning*. Dengan menambahkan lebih banyak lapisan maka model pembelajaran tersebut bisa mewakili data citra berlabel dengan lebih baik. Pada *Machine Learning* terdapat teknik untuk menggunakan ekstraksi fitur dari data pelatihan dan algoritma pembelajaran khusus untuk mengklasifikasi citra maupun untuk mengenali suara. Namun, metode ini masih memiliki beberapa kekurangan baik dalam hal kecepatan dan akurasi.

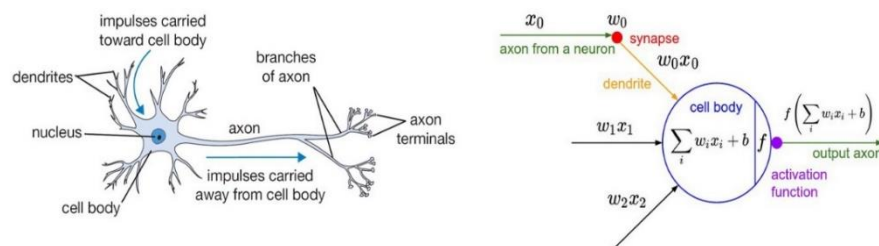
Aplikasi konsep jaringan syaraf tiruan yang dalam (banyak lapisan) dapat ditanggihkan pada algoritma *Machine Learning* yang sudah ada sehingga komputer sekarang bisa belajar dengan kecepatan, akurasi, dan skala yang besar. Prinsip ini terus berkembang hingga *Deep Learning* semakin sering digunakan pada komunitas riset dan industri untuk membantu memecahkan banyak masalah data besar seperti *Computer vision*, *Speech recognition*, dan *Natural Language Processing*. *Feature Engineering* adalah salah satu fitur utama dari *Deep Learning* untuk mengekstrak pola yang berguna dari data yang akan memudahkan model untuk membedakan kelas. *Feature Engineering* juga merupakan teknik yang paling penting untuk

mencapai hasil yang baik pada tugas prediksi. Namun, sulit untuk dipelajari dan dikuasai karena kumpulan data dan jenis data yang berbeda memerlukan pendekatan teknik yang berbeda juga.

Algoritma yang digunakan pada *Feature Engineering* dapat menemukan pola umum yang penting untuk membedakan antara kelas. Dalam *Deep Learning*, metode CNN atau *Convolutional Neural Network* sangatlah bagus dalam menemukan fitur yang baik pada citra ke lapisan berikutnya untuk membentuk hipotesis nonlinier yang dapat meningkatkan kekompleksitasan sebuah model. Model yang kompleks tentunya akan membutuhkan waktu pelatihan yang lama sehingga di dunia *Deep Learning* penggunaan GPU sudah sangatlah umum (Danukusumo, 2017)

3.8 Artificial Neural Network

Neural network adalah model yang terinspirasi oleh bagaimana neuron dalam otak manusia bekerja. Tiap neuron pada otak manusia saling berhubungan dan informasi mengalir dari setiap neuron tersebut. Gambar di bawah adalah ilustrasi neuron dengan model matematisnya.



Gambar 3.1 Ilustrasi neuron dengan model matematisnya

Tiap neuron menerima input dan melakukan operasi dot dengan sebuah weight, menjumlahkannya (*weighted sum*) dan menambahkan bias. Hasil dari operasi ini akan dijadikan parameter dari *activation function* yang akan dijadikan *output* dari neuron tersebut.

Neural Networks (Jaringan Saraf Tiruan) menurut Haykin (2009) didefinisikan sebagai Sebuah *neural network* (JST: Jaringan Saraf Tiruan) adalah prosesor yang terdistribusi paralel, terbuat dari unit-unit yang sederhana, dan memiliki

kemampuan untuk menyimpan pengetahuan yang diperoleh secara eksperimental dan siap pakai untuk berbagai tujuan. *Neural network* ini meniru otak manusia dari sudut :

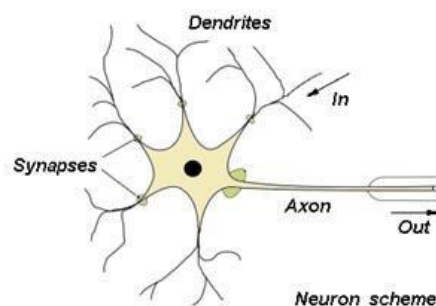
1. Pengetahuan diperoleh oleh *network* dari lingkungan, melalui suatu proses pembelajaran.
2. Kekuatan koneksi antar unit yang disebut *synaptic weights*, berfungsi untuk menyimpan pengetahuan yang telah diperoleh oleh jaringan tersebut.

Secara sederhana, Jaringan Syaraf Tiruan adalah sebuah alat pemodelan data statistik *non-linier*. Jaringan Syaraf Tiruan dapat digunakan untuk memodelkan hubungan yang kompleks antara input dan output untuk menemukan pola-pola pada data. Menurut suatu teorema yang disebut “teorema penaksiran universal”, Jaringan Syaraf Tiruan dengan minimal sebuah lapis tersembunyi dengan fungsi aktivasi *non-linear* dapat memodelkan seluruh fungsi terukur *Borel* apapun dari suatu dimensi ke dimensi lainnya.

3.9 Konsep Neural Network

3.9.1 Jaringan Syaraf Pada Otak Manusia

Ide dasar *Neural Network* dimulai dari otak manusia, dimana otak memuat sekitar 1011 neuron. Neuron ini berfungsi memproses setiap informasi yang masuk. Satu neuron memiliki 1 akson, dan minimal 1 dendrit. Setiap sel syaraf terhubung dengan syaraf lain, jumlahnya mencapai sekitar 104 sinapsis. Masing-masing sel itu saling berinteraksi satu sama lain yang menghasilkan kemampuan tertentu pada kerja otak manusia.



Gambar 3.2 Skema Neuron

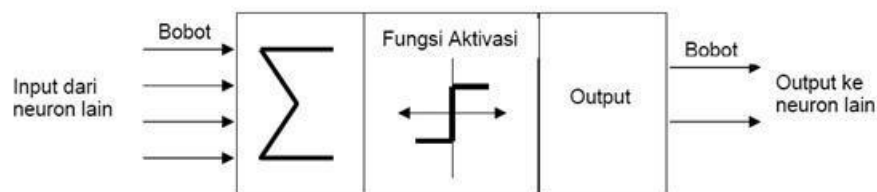
Dari gambar di atas, bisa dilihat ada beberapa bagian dari otak manusia, yaitu:

1. Dendrit (*Dendrites*) berfungsi untuk mengirimkan impuls yang diterima ke badan sel syaraf.
2. Akson (*Axon*) berfungsi untuk mengirimkan impuls dari badan sel ke jaringan lain
3. Sinapsis berfungsi sebagai unit fungsional di antara dua sel syaraf.

Proses yang terjadi pada otak manusia adalah sebuah neuron menerima impuls dari neuron lain melalui dendrit dan mengirimkan sinyal yang dihasilkan oleh badan sel melalui akson. Akson dari sel syaraf ini bercabang-cabang dan berhubungan dengan dendrit dari sel syaraf lain dengan cara mengirimkan impuls melalui sinapsis. Sinapsis adalah unit fungsional antara 2 buah sel syaraf, misal A dan B, dimana yang satu adalah serabut akson dari neuron A dan satunya lagi adalah dendrit dari neuron B. Kekuatan sinapsis bisa menurun/meningkat tergantung seberapa besar tingkat propagasi (penyiaran) sinyal yang diterimanya. Impuls-impuls sinyal (informasi) akan diterima oleh neuron lain jika memenuhi batasan tertentu, yang sering disebut dengan nilai ambang (*threshold*). (socs.binus.ac.id)

3.9.2 Struktur Neural Network

Dari struktur neuron pada otak manusia, dan proses kerja yang dijelaskan di atas, maka konsep dasar pembangunan neural network buatan (*Artificial Neural Network*) terbentuk. Ide mendasar dari *Artificial Neural Network* (ANN) adalah mengadopsi mekanisme berpikir sebuah sistem atau aplikasi yang menyerupai otak manusia, baik untuk pemrosesan berbagai sinyal elemen yang diterima, toleransi terhadap kesalahan/*error*, dan juga *parallel processing*.



Gambar 3.3 Struktur Artificial Neural Network

Karakteristik dari ANN dilihat dari pola hubungan antar neuron, metode penentuan bobot dari tiap koneksi, dan fungsi aktivasinya. Gambar di atas menjelaskan struktur ANN secara mendasar, yang dalam kenyataannya tidak hanya sederhana seperti itu.

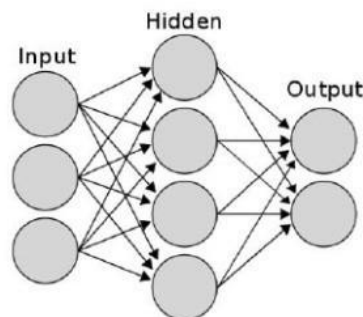
1. *Input*, berfungsi seperti dendrite
2. *Output*, berfungsi seperti akson
3. Fungsi aktivasi, berfungsi seperti sinapsis

Neural network dibangun dari banyak node/unit yang dihubungkan oleh *link* secara langsung. *Link* dari unit yang satu ke unit yang lainnya digunakan untuk melakukan propagasi aktivasi dari unit pertama ke unit selanjutnya. Setiap *link* memiliki bobot numerik. Bobot ini menentukan kekuatan serta penanda dari sebuah konektivitas.

Proses pada ANN dimulai dari *input* yang diterima oleh neuron beserta dengan nilai bobot dari tiap-tiap *input* yang ada. Setelah masuk ke dalam neuron, nilai *input* yang ada akan dijumlahkan oleh suatu fungsi perambatan (*summing function*), yang bisa dilihat seperti pada di gambar dengan lambang sigma (Σ). Hasil penjumlahan akan diproses oleh fungsi aktivasi setiap neuron, disini akan dibandingkan hasil penjumlahan dengan *threshold* (nilai ambang) tertentu. Jika nilai melebihi *threshold*, maka aktivasi neuron akan dibatalkan, sebaliknya, jika masih dibawah nilai *threshold*, neuron akan diaktifkan. Setelah aktif, neuron akan mengirimkan nilai *output* melalui bobot-bobot outputnya ke semua neuron yang berhubungan dengannya. Proses ini akan terus berulang pada input-input selanjutnya. (socs.binus.ac.id)

3.9.3 Arsitektur Jaringan Syaraf Tiruan

Secara umum, Arsitektur JST terdiri atas beberapa lapisan, yaitu lapisan masukan (*input layer*), lapisan tersembunyi (*hidden layer*), dan lapisan keluaran (*output layer*). Masing-masing lapisan mempunyai jumlah node atau neuron yang berbeda-beda. Arsitektur JST tersebut dapat diilustrasikan sebagai gambar berikut ini :



Gambar 3.4 Arsitektur Jaringan Syaraf Tiruan

a. Lapisan Masukan (*input layer*)

Lapisan masukan merupakan lapisan yang terdiri dari beberapa neuron yang akan menerima sinyal dari luar dan kemudian meneruskan ke neuron-neuron lain dalam jaringan. Lapisan ini diilhami berdasarkan ciri-ciri dan cara kerja sel-sel saraf sensori pada jaringan saraf biologi.

b. Lapisan tersembunyi (*hidden layer*)

Lapisan tersembunyi merupakan tiruan dari sel-sel syaraf konektor pada jaringan saraf biologis. Lapisan tersembunyi berfungsi meningkatkan kemampuan jaringan dalam memecahkan masalah. Konsekuensi dari adanya lapisan ini adalah pelatihan menjadi makin sulit atau lama.

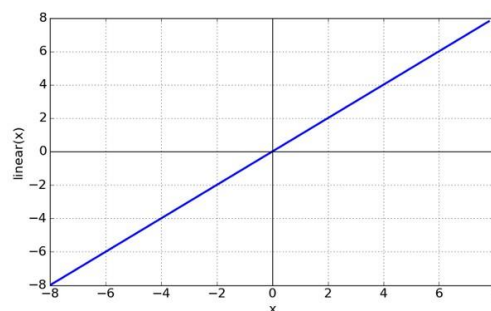
c. Lapisan keluaran (*output layer*)

Lapisan keluaran berfungsi menyalurkan sinyal-sinyal keluaran hasil pemrosesan jaringan. Lapisan ini juga terdiri dari sejumlah neuron. Lapisan keluaran merupakan tiruan dari sel saraf motor pada jaringan saraf biologis.

3.9.4 Activation Function

Sesuai dengan namanya, *activation function* berfungsi untuk menentukan apakah neuron tersebut harus “aktif” atau tidak berdasarkan dari *weighted sum* dari *input*. Secara umum terdapat 2 jenis *activation function*, *Linear* dan *Non-Linear Activation function*.

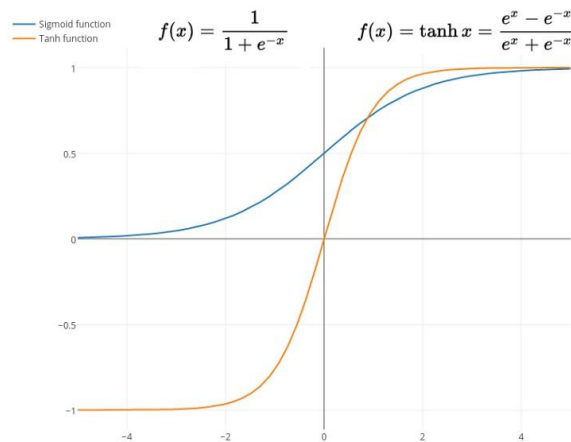
a. Linear Function



Gambar 3.5 Linear Function

Bisa dikatakan secara “default” *activation function* dari sebuah neuron adalah *Linear*. Jika sebuah neuron menggunakan *linear function*, maka keluaran dari neuron tersebut adalah *weighted sum* dari *input* + bias.

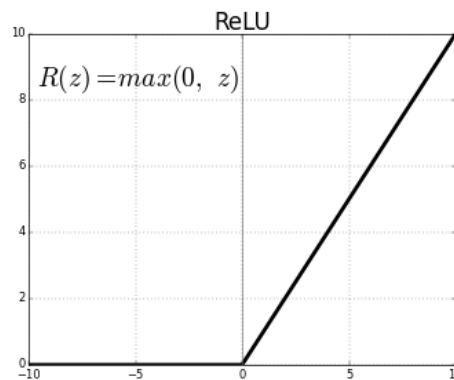
b. Sigmoid and Tanh Function (Non-Linear)



Gambar 3.6 Sigmoid and Tanh Function (Non-Linear)

Sigmoid function mempunyai rentang antara 0 hingga 1 sedangkan rentang dari *Tanh* adalah -1 hingga 1. Kedua fungsi ini biasanya digunakan untuk klasifikasi 2 class atau kelompok data.

c. ReLU (Non-Linear)



Gambar 3.7 ReLU (Non-Linear)

Pada dasarnya *ReLU* melakukan “threshold” dari 0 hingga infinity. *ReLU* juga dapat menutupi kelemahan yang dimiliki oleh *Sigmoid* dan *Tanh*. Sebenarnya masih banyak *activation function* yang lain, namun beberapa fungsi yang disebutkan diatas merupakan fungsi yang sering digunakan. (www.medium.com)

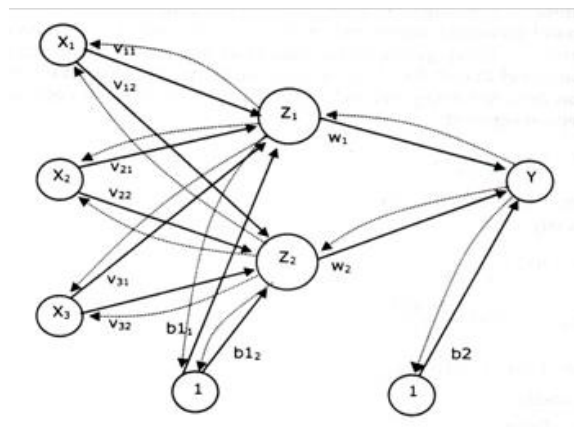
3.9.5 Pembelajaran Backpropagation

Salah satu sifat neural network yang menyerupai dengan otak manusia adalah bahwa neural network membutuhkan proses pembelajaran. Pembelajaran dilakukan untuk menentukan nilai bobot yang tepat untuk masing-masing input. Bobot

bertambah jika informasi yang diberikan oleh neuron yang bersangkutan tersampaikan. Sebaliknya jika informasi tidak disampaikan maka nilai bobot berubah secara dinamis sehingga dicapai suatu nilai yang seimbang. Apabila nilai ini telah mampu mengindikasikan hubungan yang diharapkan antara input dan output, proses pembelajaran bisa dihentikan.

Backpropagation merupakan algoritma pembelajaran yang terawasi dan biasanya digunakan oleh *perceptron* dengan banyak lapisan untuk mengubah bobot-bobot yang terhubung dengan neuron-neuron yang ada pada lapisan tersembunyi. Algoritma ini menggunakan *error output* untuk mengubah nilai bobot-bobotnya dalam arah mundur (*backward*). Untuk mendapatkan error ini, tahap perambatan maju (*forward propagation*) harus dikerjakan terlebih dahulu. Pada saat perambatan maju, neuron-neuron diaktifkan dengan menggunakan fungsi aktivasi yang dapat diturunkan, seperti fungsi *sigmoid* (Kusumadewi, 2004).

Arsitektur jaringan *backpropagation* seperti ditunjukkan pada Gambar 3.8. Gambar tersebut menunjukkan neural network yang terdiri dari tiga unit neuron pada lapisan input (x_1 , x_2 , dan x_3), dua neuron pada lapisan tersembunyi (Z_1 dan Z_2), dan 1 unit neuron pada lapisan output (Y). Bobot yang menghubungkan x_1 , x_2 , dan x_3 dengan neuron pertama pada lapisan tersembunyi adalah V_{11} , V_{21} , dan V_{31} . b_{11} dan b_{12} adalah bobot bias yang menuju neuron pertama dan kedua pada lapisan tersembunyi. Bobot yang menghubungkan Z_1 dan Z_2 dengan neuron pada lapisan output adalah w_1 dan w_2 . Bobot bias b_2 menghubungkan lapisan tersembunyi dengan lapisan output.



Gambar 3.8 Jaringan Backpropagation

Algoritma *backpropagation* untuk jaringan dengan satu lapisan tersembunyi sebagaimana pada Gambar 3.8 bekerja sebagai berikut (Kusumadewi,2004):

- A) Inisialisasi bobot (menetapkan nilai bobot awal untuk semua parameter).
- B) Tetapkan kondisi berhenti yang berupa maksimum *epoch*, iterasi, atau target *error*. Satu *epoch* adalah satu putaran training untuk semua data latih yang ada. Gambar 3.11 Jaringan Backpropagation
- C) Tetapkan *learning rate*(α). D) Inisialisasi *epoch*=0. E) Kerjakan selama kondisi berhenti belum terpenuhi (*epoch* < maksimum *epoch* dan *error* < target *error*):
 1. *epoch*=*epoch*+1.
 2. Untuk tiap-tiap pasangan elemen yang dilakukan pembelajaran, lakukan feedforward.
 3. Hitung kesalahan (*error*) antara hasil klasifikasi dan label kelas, dan gunakan informasi ini untuk mencari gradien kesalahan terhadap parameter-parameter yang ada.
 4. Lakukan *update* bobot dengan informasi gradien yang didapatkan sebelumnya dengan Persamaan (1).

$$w(t+1)=w(t)-\eta \nabla E(w(t)) \quad (1)$$

Pada persamaan tersebut, $w(t+1)$ adalah bobot yang baru, $w(t)$ adalah bobot lama, η adalah *learning rate* dan $E(w(t))$ adalah gradien dari ∇ *error*.

3.9.6 Bias

Bias dapat ditambahkan sebagai salah satu komponen dengan nilai bobot yang selalu bernilai 1. Jika melibatkan bias, maka fungsi aktivasi menjadi:

$$f(x) = \begin{cases} 1 & \text{jika } net \geq 0 \\ -1 & \text{jika } net < 0 \end{cases} \quad (2)$$

Dimana:

$$x = b + \sum_i x_i w_i \quad (3)$$

3.9.7 Learning Rate

Penggunaan parameter *learning rate* memiliki pengaruh penting terhadap waktu yang dibutuhkan untuk tercapainya target yang diinginkan. Secara perlahan akan mengoptimalkan nilai perubahan bobot dan menghasilkan *error* yang lebih kecil (Fajri, 2011). Variabel *learning rate* menyatakan suatu konstanta yang bernilai antara 0.1-0.9. Nilai tersebut menunjukkan kecepatan belajar dari jaringannya. Jika nilai *learning rate* yang digunakan terlalu kecil maka terlalu banyak *epoch* yang dibutuhkan untuk mencapai nilai target yang diinginkan, sehingga menyebabkan proses *training* membutuhkan waktu yang lama. Semakin besar nilai *learning rate* yang digunakan maka proses pelatihan jaringan akan semakin cepat, namun jika terlalu besar justru akan mengakibatkan jaringan menjadi tidak stabil dan menyebabkan nilai *error* berulang bolak-balik diantara nilai tertentu, sehingga mencegah *error* mencapai target yang diharapkan. Oleh karena itu pemilihan nilai variable *learning rate* harus seoptimal mungkin agar didapatkan proses *training* yang cepat (Hermawan, 2006).

3.9.8 Fungsi Error

Dalam masalah klasifikasi setiap *input* $x(n)$ memiliki label, *output* yang diinginkan $d(n)$ kemudian dapat dibuat fungsi kesalahan sebagai berikut

$$E(w) = - \sum_{n=1}^N \ln \left(\frac{e^{y_{d(n)}(n)}}{\sum_{i=1}^k e^{y_i(n)}} \right). \quad (4)$$

Fungsi ini disebut *cross-entropy error* dan biasanya digunakan untuk masalah klasifikasi karena memiliki hukuman yang tinggi untuk klasifikasi yang salah. n adalah pola input yang diuji dan i adalah indeks untuk *node output*. Tujuan dari algoritma pelatihan adalah untuk meminimalkan fungsi ini. (Gummesson, 2016)

3.10 Convolution 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. 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. 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. (Suartika, I Wayan, Arya Yudhi Wijaya, dan Rully Soelaiman).

3.10.1 Operasi Konvolusi

Operasi Konvolusi Operasi konvolusi adalah operasi pada dua fungsi argumen bernilai nyata (Goodfellow dkk, 2016). Operasi ini menerapkan fungsi *output* sebagai *Feature Map* dari input citra. Input dan *output* ini dapat dilihat sebagai dua argumen bernilai riil. Secara formal operasi konvolusi dapat ditulis dengan rumus berikut.

$$s(t) = (x * w)(t) \quad (5)$$

Fungsi $s(t)$ memberikan *output* tunggal berupa *Feature Map*, argumen pertama adalah *input* yang merupakan x dan argumen kedua w sebagai kernel atau filter. Jika kita melihat *input* sebagai citra dua dimensi, maka kita bisa mengasumsikan t sebagai *pixel* dan menggantinya dengan i dan j . Oleh karena itu, operasi untuk konvolusi ke *input* dengan lebih dari satu dimensi dapat ditulis sebagai berikut

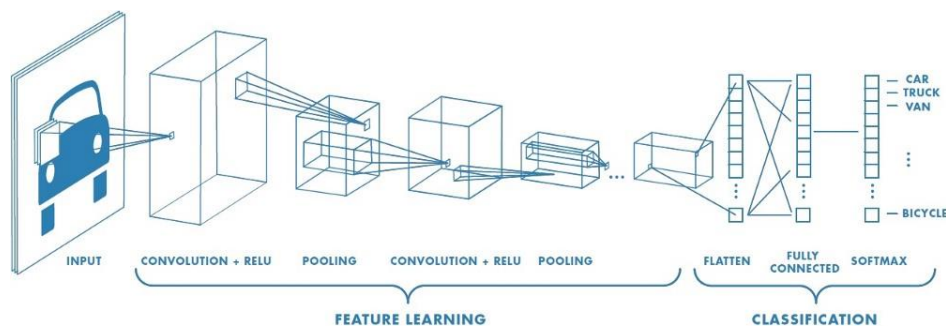
$$S_{(i,j)} = (K * I)_{(i,j)} = \sum \sum I_{(i-m,j-n)} K_{(m,n)} \quad (6)$$

Persamaan di atas adalah perhitungan dasar dalam operasi konvolusi dimana i dan j adalah piksel dari citra. Perhitungannya bersifat komutatif dan muncul saat K sebagai kernel, I sebagai input dan kernel yang dapat dibalik relatif terhadap input. Sebagai alternatif, operasi konvolusi dapat dilihat sebagai

perkalian matriks antara citra masukan dan kernel dimana keluarannya dapat dihitung dengan *dot product*. (Rismiyati, 2016)

3.10.2 Arsitektur Jaringan CNN

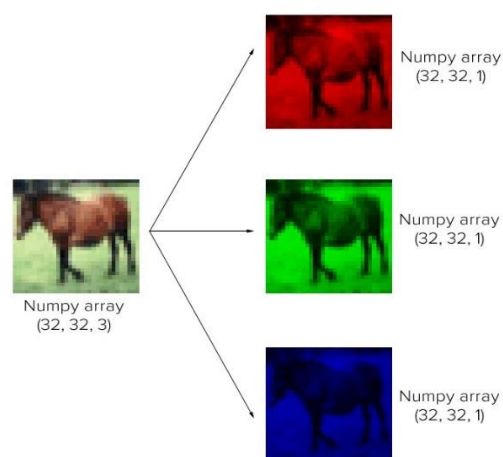
Arsitektur dari CNN dibagi menjadi 2 bagian besar, **Feature Extraction Layer** dan **Fully-Connected Layer (MLP)**.



Gambar 3.9 Arsitektur CNN

a. Feature Extraction Layer

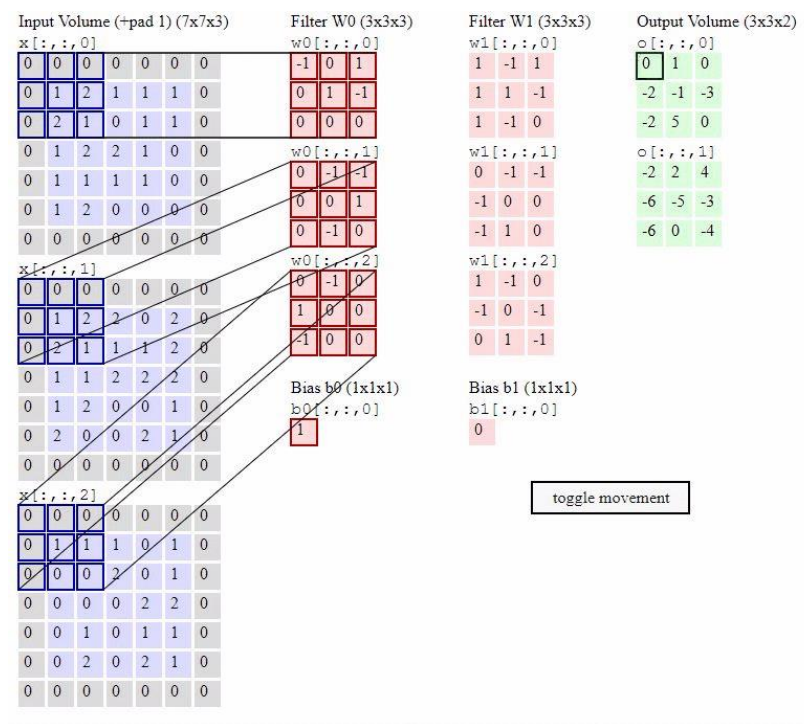
Proses yang terjadi pada bagian ini adalah melakukan “encoding” dari sebuah *image* menjadi *features* yang berupa angka-angka yang merepresentasikan image tersebut (*Feature Extraction*). *Feature extraction layer* terdiri dari dua bagian. *Convolutional Layer* dan *Pooling Layer*. Namun kadang ada beberapa riset/paper yang tidak menggunakan *pooling*.



Gambar 3.10 Image RGB

Gambar diatas adalah RGB (*Red, Green, Blue*) *image* berukuran 32x32 pixels yang sebenarnya adalah multidimensional array dengan ukuran 32x32x3 (3 adalah jumlah *channel*). *Convolutional layer* terdiri dari neuron yang tersusun sedemikian rupa sehingga membentuk sebuah filter dengan panjang dan tinggi (*pixels*). Sebagai contoh, layer pertama pada *feature extraction layer* biasanya adalah *convolution layer* dengan ukuran 5x5x3. Panjang 5 *pixels*, tinggi 5 *pixels* dan tebal atau jumlah 3 buah sesuai dengan channel dari *image* tersebut.

Ketiga *filter* ini akan digeser keseluruh bagian dari gambar. Setiap pergeseran akan dilakukan operasi “dot” antara *input* dan nilai dari *filter* tersebut sehingga menghasilkan sebuah *output* atau biasa disebut sebagai *activation map* atau *feature map*.



Gambar 3.11 *Feature Map*

Stride adalah parameter yang menentukan berapa jumlah pergeseran *filter*. Jika nilai *stride* adalah 1, maka *conv. filter* akan bergeser sebanyak 1 *pixels* secara horizontal lalu vertical. Pada ilustrasi diatas, *stride* yang digunakan adalah 2. Semakin kecil *stride* maka akan semakin detail informasi yang kita dapatkan dari sebuah *input*, namun membutuhkan komputasi yang lebih jika dibandingkan

dengan *stride* yang besar. Namun perlu diperhatikan bahwa dengan menggunakan *stride* yang kecil kita tidak selalu akan mendapatkan performa yang bagus.

Padding atau *Zero Padding* adalah parameter yang menentukan jumlah *pixels* (berisi nilai 0) yang akan ditambahkan di setiap sisi dari *input*. Hal ini digunakan dengan tujuan untuk memanipulasi dimensi *output* dari *conv. layer* (*Feature Map*).

Tujuan dari penggunaan *padding* adalah dimensi *output* dari *conv. layer* selalu lebih kecil dari *inputnya* (kecuali penggunaan 1x1 *filter* dengan *stride* 1). *Output* ini akan digunakan kembali sebagai *input* dari *conv. layer* selanjutnya, sehingga makin banyak informasi yang terbuang. Dengan menggunakan *padding*, kita dapat mengatur dimensi *output* agar tetap sama seperti dimensi *input* atau setidaknya tidak berkurang secara drastis. Sehingga kita bisa menggunakan *conv. layer* yang lebih dalam sehingga lebih banyak *features* yang berhasil di-extract. *Padding* juga mampu meningkatkan performa dari model, karena *conv. filter* akan fokus pada informasi yang sebenarnya yaitu yang berada diantara *zero padding* tersebut. Pada ilustrasi diatas, dimensi dari *input* sebenarnya adalah 5x5, jika dilakukan *convolution* dengan *filter* 3x3 dan *stride* sebesar 2, maka akan didapatkan *feature map* dengan ukuran 2x2. Namun jika kita tambahkan *zero padding* sebanyak 1, maka *feature map* yang dihasilkan berukuran 3x3 (lebih banyak informasi yang dihasilkan). Untuk menghitung dimensi dari *feature map* kita bisa gunakan rumus seperti dibawah ini:

$$\text{output} = \frac{W-N+2P}{s} + 1 \quad (7)$$

W = Panjang/Tinggi *Input*

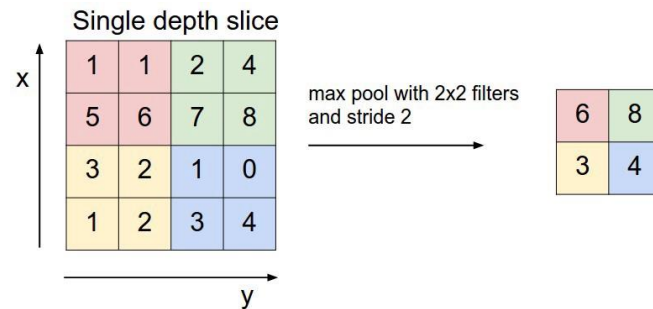
N = Panjang/Tinggi *Filter*

P = *Zero Padding*

S = *Stride*

Pooling layer biasanya berada setelah *conv. layer*. 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*. Sebagai contoh jika kita menggunakan *Max Pooling* 2x2

dengan *stride* 2, maka pada setiap pergeseran *filter*, nilai *maximum* pada area 2×2 *pixel* tersebut yang akan dipilih, sedangkan *Average Pooling* akan memilih nilai rata-ratanya.



Gambar 3.12 Max Pooling

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

b. Fully-Connected Layer (FC Layer)

Feature map yang dihasilkan dari *feature extraction layer* masih berbentuk *multidimensional array*, sehingga harus melakukan “flatten” atau *reshape feature map* menjadi sebuah *vector* agar bisa digunakan sebagai *input* dari *fully-connected layer*.

Lapisan *Fully-Connected* adalah lapisan di mana semua neuron aktivasi dari lapisan sebelumnya terhubung semua dengan neuron di lapisan selanjutnya seperti halnya jaringan saraf tiruan biasa. Setiap aktivasi dari lapisan sebelumnya perlu diubah menjadi data satu dimensi sebelum dapat dihubungkan ke semua neuron di lapisan

Lapisan *Fully-Connected* biasanya digunakan pada metode *Multi* lapisan *Perceptron* dan bertujuan untuk mengolah data sehingga bisa diklasifikasikan. Perbedaan antara lapisan *Fully-Connected* dan lapisan konvolusi biasa adalah neuron di lapisan konvolusi terhubung hanya ke daerah tertentu pada *input*, sementara lapisan *Fully-Connected* memiliki neuron yang secara keseluruhan terhubung. Namun, kedua lapisan tersebut masih mengoperasikan produk dot, sehingga fungsinya tidak begitu berbeda. (Danukusumo, 2017)

BAB IV

METODOLOGI PENELITIAN

4.1 Populasi dan Sampel

Populasi dalam penelitian ini adalah gambar *ballpoint*, penghapus dan penggaris yang diambil dari situs pencarian *google* sedangkan sampel yang digunakan dalam penelitian ini adalah gambar *ballpoint*, penghapus dan penggaris yang hanya memuat satu *object* dalam satu gambar dengan total 300 gambar.

4.2 Variabel dan Definisi Operasional Variabel

Variable yang digunakan dalam penelitian ini dalam Tabel 4.1 tentang penjelasan dan definisi operasional penelitian.

Tabel 4.1 *Definisi Operasional Penelitian*

Variabel	Devinisi Operasional Variabel
Ballpoint	Gambar yang hanya berisi <i>ballpoint</i> dengan <i>background</i> putih
Penghapus	Gambar yang hanya berisi penghapus dengan <i>background</i> putih
Penggaris	Gambar yang hanya berisi penggaris dengan <i>background</i> putih

Data gambar dibagi kedalam 2 kelompok yaitu data *train* dan data *test* yang dilakukan beberapa perbandingan proporsi, diantaranya 80% dengan 20%, 70% dengan 30% dan 60% dengan 40%. Kemudian untuk percobaan model terbaik digunakan data *test* sebanyak 3 gambar dimana berisi 1 gambar masing – masing *variable*.

4.3 Jenis dan Sumber Data

Jenis data yang digunakan dalam penelitian ini adalah data primer. Data tersebut diperoleh dengan teknik *crawling* gambar dari situs *google*.

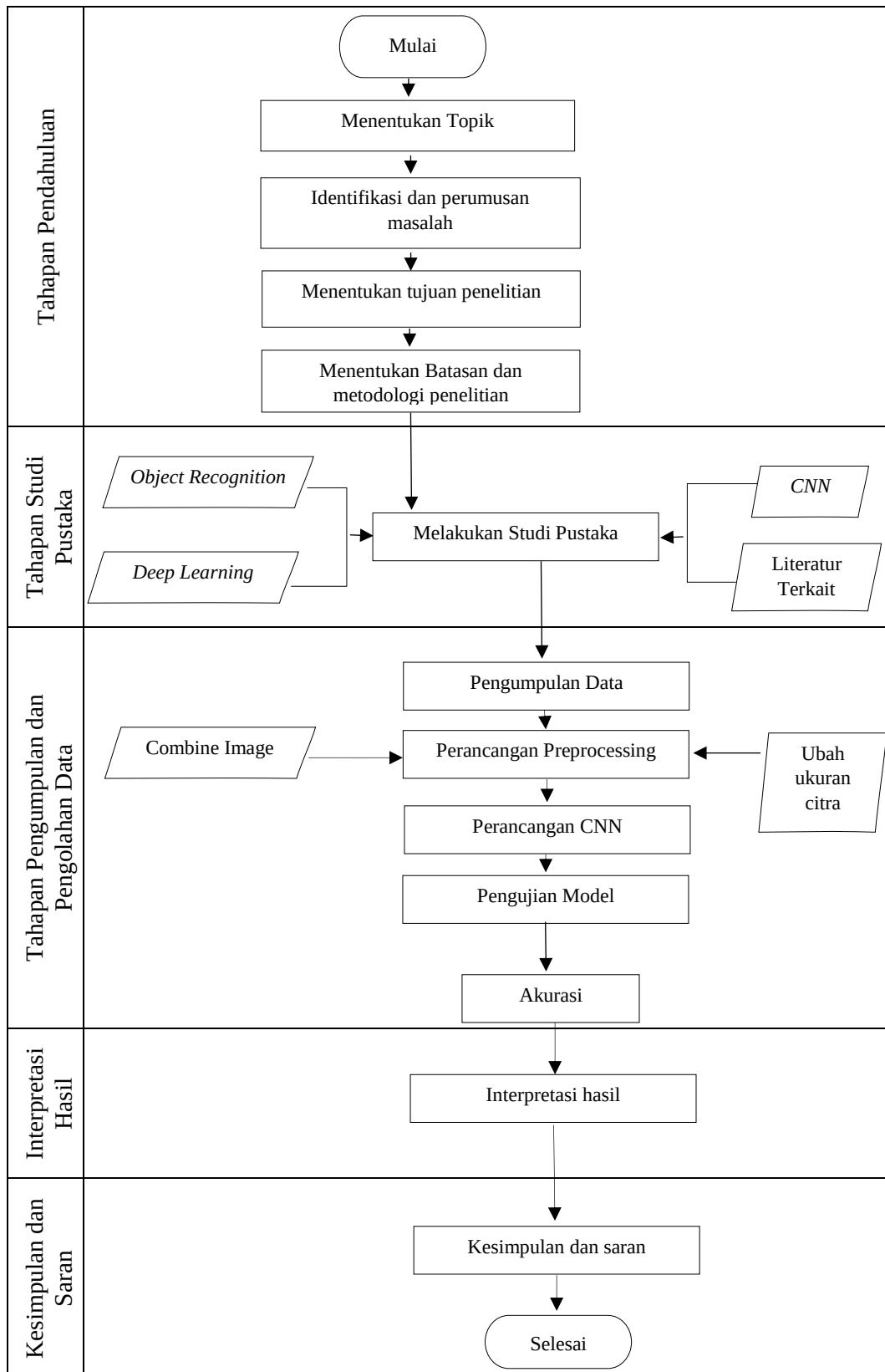
4.4 Metode Analisis Data

Software yang digunakan dalam penelitian ini adalah *google chrome*, *python 3.5.4*, *javascript* dan *R Studio*. Metode analisis data yang digunakan dalam

penelitian ini adalah metode *Convolution Neural Network*, digunakan untuk mengklasifikasikan gambar *ballpoint*, penghapus dan penggaris.

4.5 Tahapan Penelitian

Langkah atau tahapan yang dilakukan pada penelitian ini digambarkan melalui **Gambar 4.1** berikut ini:



Gambar 4.1 Diagram alir penelitian

BAB V

ANALISIS DAN PEMBAHASAN

Pada penelitian ini, peneliti melakukan klasifikasi gambar *ballpoint*, penggaris, dan penghapus dengan menggunakan *Convolutional Neural Network* (CNN). Proses utama dalam pembuatan model klasifikasi metode CNN adalah perancangan klasifikasi. Prinsip dasar dari proses ini adalah dengan melakukan pelatihan pada CNN untuk menghasilkan model yang paling bagus dalam hal akurasi yang baik. Dalam pelatihan CNN terdapat data *train* dan data *test* atau *validation*, data *train* digunakan untuk melakukan *training* pada model dan data *test* digunakan untuk validasi dari data training. Akurasi model tersebut dapat ditentukan dengan melakukan validasi dengan menggunakan data uji. Pelatihan model menggunakan *library keras* pada *Rstudio* dengan *back-end Tensorflow*. *Google* mengembangkan *Keras* sebagai pembungkus *Tensorflow* untuk menerapkan dan memberikan bagian dari *folder contrib* dalam kode *Tensorflow*. *Keras* dirancang untuk menghilangkan kode *boilerplate*. Beberapa baris kode *keras* akan menghasilkan lebih banyak daripada kode *Tensorflow* asli, sehingga dapat dengan mudah merancang CNN dan RNN serta dapat menjalankannya di GPU atau CPU.

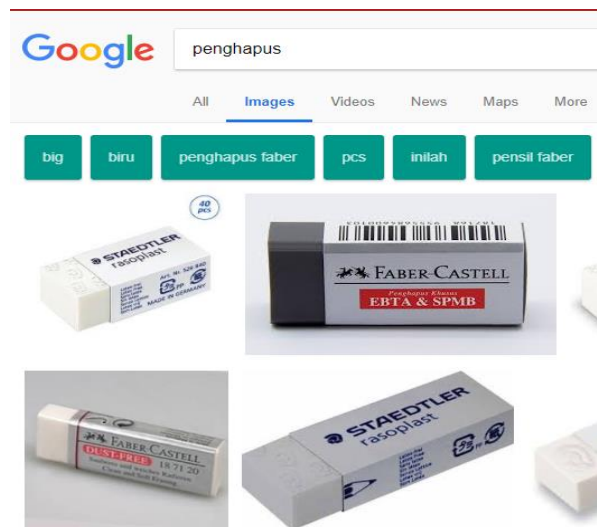
5.1 Pembuatan Dataset

Peneliti mengumpulkan dan membuat *dataset* berupa gambar yang di dapat dari *google image*. *Deep learning* terutama CNN biasanya membutuhkan banyak data gambar, untuk *mendownload* gambar pada *google image* secara *manual* tentu membutuhkan banyak waktu dan membosankan. Sehingga perlu ditemukan solusi untuk memanfaatkan *google image* sebagai saran membuat *dataset* yang baik dan cepat. Peneliti menggunakan program *javascript* dan *python* untuk mengumpulkan gambar dari *google image*. Pada program *javascript* ditujukan untuk mengambil *URL* gambar yang ada di *google image* kemudian program *python* yang melakukan eksekusi untuk *mendownload* gambar tersebut.

5.1.1 Program Javascript

Program ini digunakan untuk mengambil *URL* pada *google image* dengan menggunakan *library javascriptscript* yaitu *jQuery*. *jQuery* adalah perpustakaan *Javascript* yang cepat, kecil, dan kaya fitur. *jQuery* membuat hal-hal seperti *traversal* dan manipulasi dokumen HTML, penanganan *event*, animasi, dan *Ajax* lebih sederhana dengan API (*application programming interface*) yang mudah digunakan yang bekerja di banyak browser. Dengan kombinasi fleksibilitas dan kemampuan besar, *jQuery* telah mengubah cara jutaan orang menulis *JavascriptScript*.

1. Langkah pertama dalam menggunakan *google image* untuk mengumpulkan data pelatihan untuk membuat model *Deep Learning* adalah menuju ke *google image* dan memasukkan *query*. Dalam hal ini kita akan menggunakan istilah *query* "Penghapus"



Gambar 5.1 Google Image Search Penghapus

2. Kemudian menggunakan *javascripts* untuk mengumpulkan *URL* gambar yang kemudian nantinya akan *download* menggunakan program *python*. Sebelumnya peneliti menentukan batas *URL* yang akan *download*. *Javascriptscript* di *input* pada *console* yang ada pada *developer tools* pada browser, peneliti menggunakan software *google chrome*.

5.1.2 Program python

Peneliti telah mempunyai *file urls* yang telah *download*, kemudian menggunakan program *python* untuk *download* gambar dari masing – masing *urls*. Dalam program *python* ini menggunakan *library request*.

Tabel 5.1 Import Packages

```
1 # import the necessary packages
2 from imutils import paths
3 import argparse
4 import requests
5 import cv2
6 import os
```

Packages atau *module* yang digunakan yaitu *request* untuk *download* gambar yang ada di *urls*, kemudian *argsparse* digunakan untuk membuat argument menulis *inputan* kode, *cv2* digunakan untuk membaca lokasi *output* gambar. Kemudian paket *os* digunakan untuk menyimpan gambar pada *folder output* yang sudah di tentukan. Dan yang terakhir *imutils* digunakan pada saat *list file* gambar pada *folder output* untuk dilakukan perulangan.

Tabel 5.2 Membuat Argument

```
8 # arguments
9 ap = argparse.ArgumentParser()
10 ap.add_argument("-u", "--urls", required=True,
11                 help="path to file containing image URLs")
12 ap.add_argument("-o", "--output", required=True,
13                 help="path to output directory of images")
14 args = vars(ap.parse_args())
15
16 # list URL dari input data kemudian di inisialisasi untuk menentukan
17 # total gambar
18 rows = open(args["urls"]).read().strip().split("\n")
19 total = 0
```

Selanjutnya, mengurai argumen baris perintah dan memuat *urls* dari *disk* ke memori. Urutan baris perintah *parsing* dilakukan pada baris 9-14 disini menggunakan dua *parsing*:

--urls : *Path* dari *file* yang berisi *urls* gambar yang dihasilkan oleh *Javascript* di atas.

--output : *Path* dari *output* untuk menyimpan gambar yang *download* dari *Google Images*.

Pada baris 18 digunakan untuk memuat setiap *urls* dari *file* ke dalam daftar, kemudian juga menginisialisasi sebuah *counter*, *total*, untuk menghitung *file* yang

telah kami *download*. Pada baris 18 terlihat proses yaitu membuka *parsing urls* kemudian membaca *file* tersebut (*.read()*). Selanjutnya mengembalikan salinan *string* dengan karakter terdepan dan *trailing* yang dihapus (berdasarkan argumen *string* yang dilewati) (*.strip()*). Kemudian menghapus “\n” pada data *urls*.

Tabel 5.3 *Perulangan Download URL*

```

21 # Perulangan URLs
22 for url in rows:
23     try:
24         # download image
25         r = requests.get(url, timeout=60)
26
27         # save image
28         p = os.path.sep.join([args["output"], "{}.jpg".format(
29             str(total).zfill(8))])
30         f = open(p, "wb")
31         f.write(r.content)
32         f.close()
33
34         # update the counter
35         print("[INFO] downloaded: {}".format(p))
36         total += 1
37
38     # jika terdapat kesalahan dalam proses download
39     except:
40         print("[INFO] error downloading
    {}".format(p))

```

Dengan menggunakan *requests*, disini hanya perlu menentukan *urls* dan *timeout* untuk *download*. Peneliti mencoba mendownload *file* gambar ke dalam variabel, *r*, yang menampung *file* biner (bersama dengan *header HTTP*, dll.) dalam memori sementara atau biasa disebut RAM (*Random Access Memory*) (Kode baris 25). Selanjutnya menyimpan gambar ke *disk*, hal pertama yang diperlukan adalah *path* dan nama *file* yang *valid*. Kode baris 28-29 menghasilkan *path* + *filename*, *p*, yang akan menghitung secara bertahap dari 00000000.jpg. Peneliti kemudian membuat sebuah *file pointer*, *f*, menentukan *path output*, *p*, dan menunjukkan peneliti ingin menulis *mode* dalam format biner ("wb") pada kode baris 30. Selanjutnya, menulis isi *file* dari *f* (*r.content*) dan kemudian menutup *file* (Kode baris 31 dan 32). Dan akhirnya, memperbarui jumlah total gambar yang diunduh pada kode baris 35 dan 36). Jika ada kesalahan yang ditemukan di sepanjang proses pengunduhan, maka sebuah pesan dicetak ke terminal (Kode baris 39 dan 40. Peneliti akan mengulang semua *file* yang baru saja didownload dan mencoba

membukanya dengan *OpenCV*. Jika *file* tidak bisa dibuka dengan *OpenCV*, maka akan dihapus dan dilanjutkan. Ini tercakup dalam *blok kode* berikut:

Tabel 5.4 *Perulangan Load Image*

```

42 # perulangan image path
43 for imagePath in paths.list_images(args["output"]):
44     # inisialisasi jika gambar di hapus atau tidak
45     delete = False
46
47     # load image
48     try:
49         image = cv2.imread(imagePath)
50
51         # jika gambar tidak ada dari disk
52         # maka akan di hapus
53         if image is None:
54             delete = True
55
56         # jika opencv tidak bisa meload gambar
57         # seperti gambar tersebut corrupt maka akan dihapus
58     except:
59         print("Except")
60         delete = True
61
62     # check image yang di delete
63     if delete:
64         print("[INFO] deleting {}".format(imagePath))
65         os.remove(imagePath)

```

Saat melakukan perulangan setiap *file*, akan diinisialisasi *delete* ke *False* (Kode baris 45). Kemudian peneliti akan mencoba memuat *file* gambar di Kode baris 49. Jika gambar dimuat sebagai *None*, atau jika ada pengecualian, maka akan menyetel *delete = True* (Kode baris 53-54 dan kode baris 58-60). Alasan umum gambar tidak dapat dimuat yaitu kesalahan selama unduhan (seperti *file* yang tidak diunduh sepenuhnya), gambar rusak, atau format *file* gambar yang tidak dapat dibaca *OpenCV*. Terakhir jika *flag delete* diset, kita panggil *os.remove* untuk menghapus gambar pada kode baris 63-65.

5.2 Dataset Untuk Pelatihan CNN

Potongan program untuk melakukan proses pembuatan dataset ditunjukkan pada **Tabel 5.5**, menggunakan *software Rstudio*. Pada kode baris 1 digunakan untuk menentukan lokasi gambar yang telah *download* menggunakan program *python* dan kode baris 3 digunakan untuk menentukan lokasi penyimpanan gambar baru yang akan dibuat menjadi *dataset*. Untuk lokasi gambar hasil *download* berada pada file **gambar_awal** dan untuk *folder* penyimpanan data baru berada pada **data_set**. Kemudian melakukan *list file* pada *folder gambar_awal*, digunakan untuk

membaca semua *file* gambar yang ada pada *folder gambar_awal* seperti ditunjukkan kode baris 5 program. Selanjutnya kode baris 7-9 yaitu pengaturan ukuran gambar, peneliti menggunakan 100x100 untuk membuat *dataset*. Untuk *dataset* yang akan disimpan yaitu 100x100 karena lebih fleksibel untuk mengganti resolusi didalam program CNN. Kode baris 12-29 merupakan rangkaian dari perulangan yang digunakan untuk melakukan *resize* gambar. Pada kode baris 15 *tryCatch* digunakan untuk memanggil *image* kemudian didalam *tryCatch* ada dua proses, yang pertama yaitu proses mengubah ukuran gambar dan yang kedua yaitu *error* untuk proses mengubah ukuran gambar. Kemudian memberi nama gambar pada kode baris 17 dan membaca gambar seperti pada kode baris 19. Mengubah ukuran gambar ditunjukkan pada kode baris 21, menggunakan perintah *resize* dengan memasukkan gambar dan ukuran yang diinginkan. Selanjutnya meletakkan gambar hasil *resize* pada path *save_in* yaitu pada folder **data_set** menggunakan kode baris 22. Kemudian pada kode baris 24 digunakan untuk mengeksekusi dimana menyimpan gambar hasil *resize* pada **data_set** dengan kualitas 70. Konfirmasi hasil *resize* ditunjukkan pada kode baris 26, dan pada baris 28 menunjukkan *function* untuk *error*.

Tabel 5.5 Dataset Untuk Pelatihan

```

1 setwd("E://r/fixkeras/download_gambar/images/fixx/gambar_awal/")
2 # Set d where to save images
3 save_in <- ("E://r/fixkeras/download_gambar/images/fixx/data_set/")
4 # Load images names
5 images <- list.files()
6 # Set width
7 w <- 100
8 # Set height
9 h <- 100
10
11 # Main loop resize images
12 for(i in 1:length(images))
13 {
14   # Try-catch is necessary since some images
15   result <- tryCatch({
16     # Image name
17     imgname <- images[i]
18     # Read image
19     img <- readImage(imgname)
20     # Resize image 100x100
21     img_resized <- resize(img, w = w, h = h)
22     path <- paste(save_in, imgname, sep = "")
23     # Save image
24     writeImage(img_resized, path, quality = 70)
25     # Print status

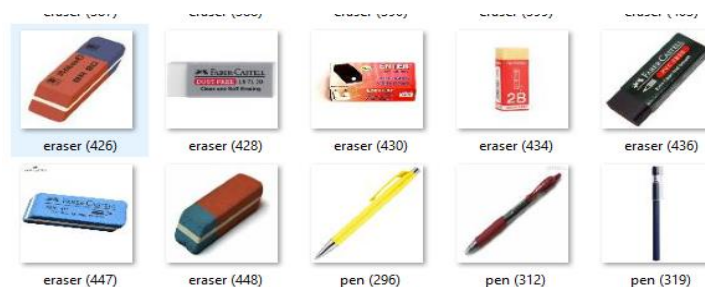
```

```

26 print(paste("Done",i,sep = " ")),
27 # Error function
28 error = function(e){print(e)}
29 }

```

Dari hasil *output*, kemudian peneliti melakukan seleksi gambar yang akan digunakan. Ciri – ciri gambar yang baik yaitu terdapat hanya *object* tersebut pada gambar dan *background* dari objek adalah putih. Peneliti menyeleksi 300 gambar untuk dijadikan data *train* dan *test*, dikarenakan keterbatasan kemampuan dari komputer yang digunakan yaitu hanya mampu melakukan *train* maksimal 300 gambar. Sehingga masing – masing kategori menggunakan 80 gambar untuk *train* dan 20 gambar untuk data *test*.



Gambar 5.4 Output Dataset untuk Training

5.3 Preprocessing CNN

Tabel 5.6 Mempersiapkan Gambar

```

1 # Convolutional Neural Networks
2 # packages
3 library(EBImage)
4 library(keras)
5
6 #set path
7 setwd("E://r/fixkeras/download_gambar/images/Final/")
8
9 # Read Images
10 images <- list.files()
11 summary(images)
12 list_of_images = lapply(images, readImage)
13 display(list_of_images[[15]])
14 print(list_of_images[[10]])
15 #create train
16 train <- list_of_images[c(1:80, 101:180,201:280)]
17 display(train[[5]])
18
19 #create test
20 test <- list_of_images[c(81:100,181:200,281:300)]
21 display(test[[10]])
22
23 #plot
24 par(mfrow = c(6, 10))
25 for (i in 1:60) plot(test[[i]])

```

Setelah membuat *dataset*, selanjutnya menyiapkan data gambar yang akan digunakan untuk membuat model CNN. *Package* yang digunakan yaitu *keras* dan *EBImage*, *keras* merupakan API dari jaringan saraf tiruan yang mampu berjalan diatas *tensorflow*. Untuk mengaktifkan *packages* tersebut maka dipanggil menggunakan perintah *library* seperti yang ditunjukkan pada kode baris 3-4. Selanjutnya pada kode baris 7, menentukan letak atau lokasi dari gambar yang telah diseleksi yaitu pada folder **Final**. Kemudian melakukan *list files* untuk membuat daftar semua gambar yang ada dalam *folder* tersebut, dan melihat deskripsi dari *file* yang ditunjukkan pada kode baris 10-11. *Output* dari *summary* yaitu banyaknya gambar, *class* dan *mode* dimana berturut –turut 300, *character* dan *character*.

Selanjutnya pada baris 12 digunakan untuk membaca gambar menggunakan *function readImage* dari data gambar image menggunakan *lapply*. Pada kode baris 13-14 digunakan untuk menampilkan gambar ke 15 dan melihat *properties* dari gambar ke 10 pada data *list_of_image*.

```
> print(list_of_images[[10]])
Image
  colorMode      : color
 storage.mode    : double
    dim         : 100 100 3
 frames.total    : 3
 frames.render   : 1

imageData(object)[1:5,1:6,1]
  [,1] [,2] [,3] [,4] [,5] [,6]
[1,]   1    1    1    1    1    1
[2,]   1    1    1    1    1    1
[3,]   1    1    1    1    1    1
[4,]   1    1    1    1    1    1
[5,]   1    1    1    1    1    1
```

Gambar 5.5 *Properties Gambar*

Pada gambar diatas dapat dilihat bahwa *properties* dari gambar ke-10 dari *list_of_image*, dimana dimensi yang ada dengan *pixel* 100x100 dan 3 berarti RGB. Kemudian pada *image* data dapat dilihat nilai 5 untuk dimensi 100, nilai 6 juga untuk *pixel* 100 dan nilai 1 untuk dimensi 3(RGB).

Kemudian untuk kode baris 16-17 yaitu menentukan rentang dari data *train* dan memanggil gambar dari data train urutan 15. Pada *folder dataset* diberikan nama pada masing-masing masing kategori klasifikasi untuk mempermudah pengurutan dan pembuatan data train dan test. Kemudian untuk kode baris 20-21 yaitu menentukan rentang dari data *test* dan memanggil gambar dari data *test* urutan 10. Selanjutnya membuat plot dimana ukuran yang digunakan yaitu 6x10 dengan

menggunakan perulangan untuk membuat plot dari masing-masing gambar pada data *test*, seperti ditunjukkan pada kode baris 24-25. Berikut *output* dari pembuatan plot tersebut



Gambar 5.6 *Plot Gambar Test*

Dari gambar diatas dapat dilihat ukuran dari *plot* yang dibuat yaitu 6x10 dengan memasukkan semua data *test* dalam *plot*. Kemudian selanjutnya yaitu proses *resize* gambar menggunakan kode pada **Tabel 5.7**

Tabel 5.7 *Resize dan Combine*

```

26 # Resize
27 for (i in 1:240) {train[[i]] <- resize(train[[i]], 32, 32)}
28 for (i in 1:240) {train[[i]] <- toRGB(train[[i]])}
29 str(train)
30
31 for (i in 1:60) {test[[i]] <- resize(test[[i]],32, 32)}
32 for (i in 1:60) {test[[i]] <- toRGB(test[[i]])}
33 str(train)
34
35 # combine
36 train <- combine(train)
37 str(train)
38 x <- tile(train, 20)
39 display(x, title='train')
40
41 test <- combine(test)
42 y <- tile(test, 10)
43 display(y, title = 'test')
44 str(test)

```

Pada *table* diatas untuk kode baris 27 merupakan perulangan untuk mengubah ukuran gambar *train* dari yang awalnya 100x100 menjadi 32x32. Ukuran gambar untuk *input shape* harus membentuk ukuran yang simetris (100x100, 64x64, 32x32). Peneliti menggunakan resolusi 32x32 karena beberapa

pertimbangan yaitu jika menggunakan 64x64 maka akan membuat proses semakin berat karena keterbatasan RAM dan GPU komputer yang digunakan kemudian apabila menggunakan resolusi 16x16 maka informasi yang didapatkan dari gambar akan terlalu sedikit dan akan berpengaruh terhadap akurasi model. Pada kode selanjutnya yaitu baris 28, mengubah semua gambar *train* menjadi RGB(*Red*, *Green*, *Blue*), karena pada saat pelatihan nantinya menggunakan warna RGB. Kemudian pada kode baris 29 digunakan untuk melihat struktur dari data *train* yang telah diresize dan diubah menjadi format RGB.

```
> str(train)
List of 240
 $ :Formal class 'Image' [package "EBImage"] with 2 slots
  ..@ .Data      : num [1:32, 1:32, 1:3] 1 1 1 1 1 1 1 1 1 1
  ..@ colormode: int 2
```

Gambar 5.7 Struktur Train

Dari **Gambar 5.12** terdapat 240 jumlah gambar *train* kemudian pada @data, ukuran *pixel* gambar telah berubah menjadi 32x32 dan format RGB yang ditunjukkan dengan angka 1:3.

Perubahan format gambar juga berlaku untuk data *test* atau *validation*, seperti pada kode baris 31-33 sehingga perlu diterapkan kembali perulangan yang sama tetapi jumlah perulangan diganti sesuai dengan jumlah gambar pada data *test* yaitu 60 gambar. Tahapan selanjutnya yaitu *combine* gambar untuk semua kategori menjadi satu seperti pada kode baris 36-37. Pada tahapan pelatihan yang dibaca terlebih dahulu pada struktur data yaitu jumlah gambar yang digunakan oleh karena itu perlu menggabungkan semua gambar menjadi satu. Sehingga struktur yang baru dari gambar *train* seperti dibawah ini.

```
> train <- combine(train)
> str(train)
Formal class 'Image' [package "EBImage"] with 2 slots
 ..@ .Data      : num [1:32, 1:32, 1:3, 1:240] 1 1 1 1 1 1 1 1 1 1
 ..@ colormode: int 2
```

Gambar 5.8 Struktur Train

Dari gambar dapat dilihat bahwa dimensi telah bertambah menjadi 4 dimensi pada data train, dengan dimensi terakhir yaitu jumlah gambar train yang digunakan.

Kemudian selanjutnya yaitu kode baris 38-39 peneliti ingin melihat bentuk dari gambar *train* hasil dari penggabungan, dengan menggunakan 20 kolom sehingga baris yang tercipta yaitu 12.



Gambar 5.9 *Train Hasil Combine*

Pada kode selanjutnya yaitu baris 41-44 digunakan untuk menggabungkan data *test* layaknya seperti data train di atas kemudian setelah di *combine* di tampilkan dengan 10 kolom dan 6 baris sehingga didapatkan *output* gambar dibawah ini .



Gambar 5.10 *Test Hasil Combine*

Dari gambar dapat dilihat bahwa ada 20 gambar untuk masing-masing kategori pada data *test*. Pada pelatihan model, dimensi yang akan dibaca terlebih dahulu yaitu total gambar, *pixel*, dan format RGB. Sehingga perlu dilakukan pertukaran posisi dimensi pada gambar *train* dan *test* seperti kode dibawah ini.

Tabel 5.8 *Reorder Dimension*

```

45 # Reorder dimension
46 train <- aperm(train, c(4,1,2,3))
47 str(train)
48 test <- aperm(test, c(4,1,2,3))
49 str(test)
50
51 # Response
52 trainy <- c(rep(0,80),rep(1,80),rep(2,80))
53 testy <- c(rep(0,20),rep(1,20),rep(2,20))
54
55 # One hot encoding
56 trainLabels <- to_categorical(trainy)
57 testLabels <- to_categorical(testy)

```

Pada kode baris 46-48 merupakan pertukaran posisi dimensi dari data *train* dan *test* agar dapat dibaca oleh model yang akan dibuat.

```

> str(train)
  num [1:240, 1:32, 1:32, 1:3] 1 1 1
> str(test)
  num [1:60, 1:32, 1:32, 1:3] 1 1 1 1

```

Gambar 5.11 *Output Hasil Reorder Dimension*

Pada gambar output diatas dapat dilihat bahwa struktur dari dimensi yang awalnya jumlah data berada pada urutan terakhir sekarang telah menjadi urutan pertama sehingga nantinya model akan mampu membaca data yang akan dilatih. Kemudian selanjutnya pada pada kode baris 52-53 digunakan untuk membuat label dari data *train* dan *test*, dimana masing masing data telah di urutan sehingga mempermudah untuk mengatur label. Kemudian hasil pembuatan label tersebut dibuat dalam dalam bentuk kategorik yang awalnya berupa data *vector* seperti pada kode baris 56-57. Setelah data siap untuk di latih selanjutnya penentuan model yang digunakan, dimana disini menggunakan CNN menggunakan API *keras* pada *library tensorflow*.

5.4 Pembuatan Model CNN

Tabel 5.9 Create Model CNN

```

58 # Model
59 model <- keras_model_sequential()
60
61 model %>%
62   layer_conv_2d(filters = 32,
63                 kernel_size = c(3,3),
64                 activation = 'relu',
65                 input_shape = c(32, 32,3)) %>%
66   layer_conv_2d(filters = 32,
67                 kernel_size = c(3,3),
68                 activation = 'relu') %>%
69   layer_max_pooling_2d(pool_size = c(2,2)) %>%
70   layer_dropout(rate = 0.0001) %>%
71   layer_conv_2d(filters = 64,
72                 kernel_size = c(3,3),
73                 activation = 'relu') %>%
74   layer_conv_2d(filters = 64,
75                 kernel_size = c(3,3),
76                 activation = 'relu') %>%
77   layer_max_pooling_2d(pool_size = c(2,2)) %>%
78   layer_dropout(rate = 0.0001) %>%
79   layer_flatten() %>%
80   layer_dense(units = 256, activation = 'relu') %>%
81   layer_dropout(rate=0.0001) %>%
82   layer_dense(units =3, activation = 'softmax') %>%
83
84   compile(loss = 'categorical_crossentropy',
85           optimizer = optimizer_sgd(lr = 0.01,
86                                     decay = 1e-6,
87                                     momentum = 0.9,
88                                     nesterov = T),
89           metrics = c('accuracy'))
90 summary(model)

```

Dalam *keras* terdapat 2 cara dalam membangun model yaitu model *sequential* dan *function API*. *Function API* digunakan untuk menentukan model kompleks, seperti model *multi-output*, grafik asiklik terarah, atau model dengan lapisan bersama. Sedangkan *sequential model* lebih seperti tumpukan linier dari *layer*. Model CNN mempunyai beberapa *layer* diantaranya *convolution*, *pooling* *dropout*. Kode baris 59 menunjukkan pengaktifan *sequential model* yaitu model sederhana yang hanya perlu penambahan *layer* terhadap model yang ada. Pada kode baris 61-82 menggunakan beberapa *layer* diantaranya *convolution layer*, *pooling layer*, *dropout layer*, *flatten layer* dan *dense layer*. Pada model diatas untuk *convolution* pertama(kode baris 61-70) menggunakan jumlah *filter* 32, *kenel size* 3x3 dan *activation relu* dengan *input shape* 32x32x3. Kemudian untuk *convolution*

yang kedua (kode baris 71-78) menggunakan *filter* 64, dengan *pool size* 2x2 dan *learning rate* 0.0001. Kode baris 78-82 merupakan *fully-connected layer*, pada *layer dense* baris 82 menunjukkan *output* yang dihasilkan yaitu 3 kategori. Kemudian didapatkan model seperti dibawah ini.

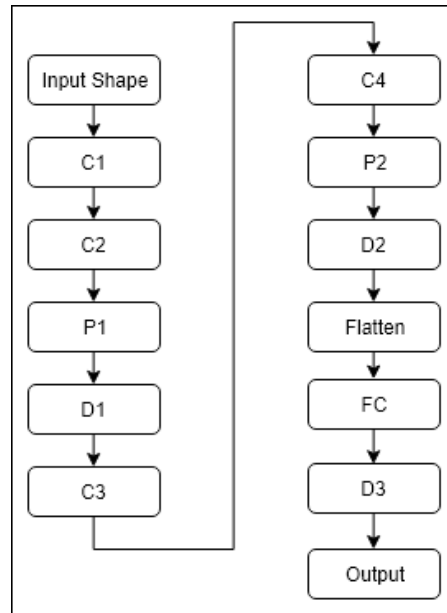
```
> summary(model)
```

Layer (type)	output shape	Param #
conv2d_28 (Conv2D)	(None, 30, 30, 32)	896
conv2d_27 (Conv2D)	(None, 28, 28, 32)	9248
max_pooling2d_14 (MaxPooling2D)	(None, 14, 14, 32)	0
dropout_21 (Dropout)	(None, 14, 14, 32)	0
conv2d_26 (Conv2D)	(None, 12, 12, 64)	18496
conv2d_25 (Conv2D)	(None, 10, 10, 64)	36928
max_pooling2d_13 (MaxPooling2D)	(None, 5, 5, 64)	0
dropout_20 (Dropout)	(None, 5, 5, 64)	0
flatten_7 (Flatten)	(None, 1600)	0
dense_14 (Dense)	(None, 256)	409856
dropout_19 (Dropout)	(None, 256)	0
dense_13 (Dense)	(None, 3)	771
Total params: 476,195		
Trainable params: 476,195		
Non-trainable params: 0		

Gambar 5.12 *Summary Model*

Dari *output* model diatas, terdapat beberapa *layer* dengan hasil *output shape* yang berbeda, perhitungan *output shape* memiliki perbedaan pada setiap jenis *layer* berikut beberapa perhitungannya.

Pada convolution ini menggunakan *input shape* 32x32x3, kemudian *filter* 32 dan *kernel* 3x3. Perhitungannya menggunakan rumus *output shape* dimana $I/p = \text{input shape}$, $C_i(\text{convolution layer } i)$, $P_i(\text{pooling layer ke } i)$, $F_c(\text{fully connected layer/dense layer})$, $D_i(\text{dropout layer ke } i)$ dan *output*.



Gambar 5.13 Proses Pembuatan Model

Dari gambar diatas dapat dilihat urutan pembuatan model. Hasil *inputan* pada model dimasukkan kedalam rumus *output shape* sehingga dapat dihitung *output shape* dari masing-masing *layer*.

I/p = 32x32x3, filter =32 kernel size 3x3, padding =0, stride =1

$$\frac{(32-3+2x0)}{1} + 1 = 30 \quad (8)$$

C1 = 30x30x32, filter =32 kernel size 3x3, padding =0, stride =1

$$\frac{(30-3+2x0)}{1} + 1 = 28 \quad (9)$$

C2 = 28x28x32, filter =32 kernel size 3x3, padding =0, stride =1

Pada *pooling layer* menggunakan dimensi 14x14 dikarenakan *stride* pada *pooling layer* = 2 sehingga *pixel* 28 pada C2 dibagi 2. Sehingga output akan menjadi 14x14.

P1 = 14x14x32

Output shape dari *dropout* mempunyai nilai sama dengan *pooling layer*

D1=14x14x32

Kemudian masuk kembali pada *convolution layer* ke 3 dengan menggunakan dimensi pada *dropout* 1 untuk di olah menjadi C3. Pada C3 peneliti menggunakan filter=64 dengan kernel 3x3 sehingga didapatkan *output shape* 12x12x64.

$$\frac{(14-3+2x0)}{1} + 1 = 12 \quad (10)$$

C3 = 12x12x64, *filter* =64 kernel size 3x3, *padding* =0, *stride* =1

$$\frac{(12-3+2 \times 0)}{1} + 1 = 10 \quad (11)$$

C4=10x10x64, *filter* =64

P2 = 2x2, 5x5x64

D2= 5x5x64

Dari hasil *droupout* 2 akan dihitung sehingga akan menemukan hasil *output shape* dari *flatten layer*.

Flat = 5x5x64=1600

FC = n=256

D3 = 256

Output = 3

Pada penelitian ini menggunakan 3 kategori sehingga pada *output* menggunakan 3. Selanjutnya yaitu melakukan proses *training* pada data menggunakan model yang telah dibuat.

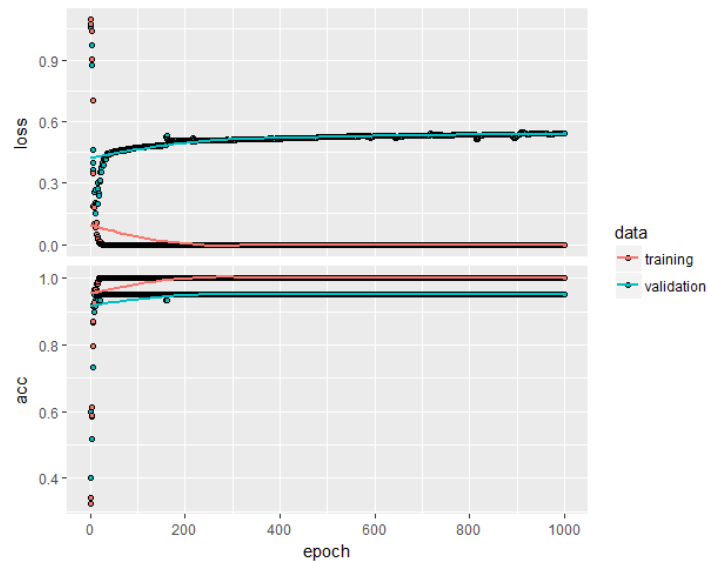
Tabel 5.10 Training Model

```

91 # Fit model
92
93 history <- model %>%
94     fit(train,
95         trainLabels,
96         epochs = 1000,
97         batch_size = 30,
98         callbacks = callback_tensorboard("logs/run_1"),
99         validation_split = 0.2,
100         validation_data = list(test, testLabels))
101 plot(history)
102 tensorboard("logs/run_1")
103

```

Fit model disini menggunakan *epoch* = 1000 kemudian *batch size* 30, kemudian hasil dari training akan ditampilkan menggunakan *tensorboard* dan *plot*. Untuk proses *training* disini menggunakan GPU sehingga membuat proses *training* semakin cepat. Kemudian setelah proses *training* didapatkan *output* berikut.



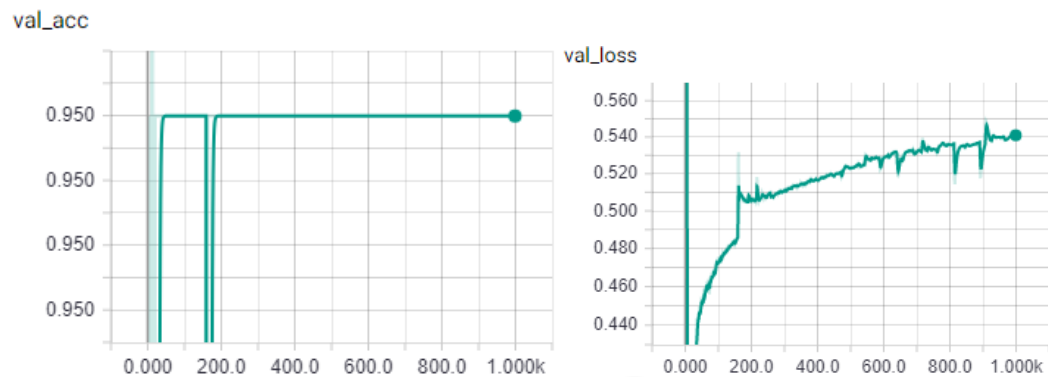
Gambar 5.14 Plot Model

Dari *plot* diatas dapat dilihat akurasi yang didapatkan, untuk titik berwarna merah menunjukkan akurasi dan *loss* data training sedangkan warna biru menunjukkan accurasi data *validation*. Dari *plot* dapat dilihat bahwa akurasi dari kedua data cenderung konstan. Akurasi akhir yang didapat untuk data *train* 1 dan data *test* 0.95, ini merupakan hasil yang cukup baik apabila dilihat dari jumlah data yang digunakan yaitu 300 data gambar. Posisi gambar juga sangat mempengaruhi akurasi apabila data sedikit.



Gambar 5.15 Output Tensorboard Akurasi Train

Dari *output tensorboard* dapat dilihat bahwa pada *epoch* sekitar 50 akurasi data telah stabil. Kemudian untuk *loss* semakin banyak *epoch* *loss* yang dihasilkan semakin kecil.



Gambar 5.16 Output Tensorboard Akurasi Test

Akurasi data test 0.95 didapat dengan *epoch* 1000, sedangkan untuk *loss*nya didapat 0.54. *Loss* semakin besar mengikuti banyaknya *epoch*.

5.5 Evaluasi dan Prediksi Hasil Train

Tabel 5.11 Evaluate and Prediction

```

104 # Evaluation & Prediction - train data
105 model %>% evaluate(train, trainLabels)
106 pred <- model %>% predict_classes(train)
107 table(Predicted = pred, Actual = trainy)
108
109 prob <- model %>% predict_proba(train)
110 cbind(prob, Predicted_class = pred, Actual = trainy)
111
112 # Evaluation & Prediction - test data
113 model %>% evaluate(test, testLabels)
114 pred <- model %>% predict_classes(test)
115 table(Predicted = pred, Actual = testy)
116
117 prob <- model %>% predict_proba(test)
118 cbind(prob, Predicted_class = pred, Actual = testy)
119
120 #save model
121 save_model_hdf5(model,filepath =
122 'E:/r/fixkeras/download_gambar/images/dinda0.85.hdf5')

```

Setelah pembuatan model selanjutnya melakukan prediksi dari model baik data *train* maupun data *test*. Pada kode baris 104-107 yaitu melakukan evaluasi dari data train dengan melihat akurasi dan *loss* dari data *train*, kemudian membuat hasil prediksi dari model terhadap data *train* dalam bentuk *table* seperti dibawah ini.

```

> model %>% evaluate(train, trainLabels)
240/240 [=====] - 0s 636us/step
$loss
[1] 4.262369e-06

$acc
[1] 1

> pred <- model %>% predict_classes(train)
> table(Predicted = pred, Actual = trainy)
      Actual
Predicted 0  1  2
      0 80  0  0
      1  0 80  0
      2  0  0 80

```

Gambar 5.17 *Output Evaluasi Model*

Dari gambar diatas dapat dilihat akurasi untuk data *train* yaitu 1, sehingga dari hasil prediksi tidak terdapat kesalahan seperti yang ditunjukkan pada *table* bahwa masing-masing kategori berada pada tempatnya. Kemudian selanjutnya prediksi model terhadap data *test* yaitu ditunjukkan pada kode baris 112-115 dimana didapatkan *output* berikut.

```

> model %>% evaluate(test, testLabels)
60/60 [=====] - 0s 451us/step
$loss
[1] 0.5409927

$acc
[1] 0.95

> pred <- model %>% predict_classes(test)
> table(Predicted = pred, Actual = testy)
      Actual
Predicted 0  1  2
      0 19  0  0
      1  1 18  0
      2  0  2 20

```

Gambar 5.18 *Output Evaluasi Model*

Dari gambar diatas didapatkan *output* bahwa akurasi untuk data *test* yaitu 0.95 sehingga dari total 60 data *test* terdapat 3 data masuk kategori yang salah. Akurasi ini bisa dibilang cukup baik mengingat jumlah data yang digunakan Cuma 300 gambar.

5.6 Perbandingan Parameter CNN

Untuk memastikan hasil dari parameter CNN diatas adalah yang terbaik, diperlukan suatu perbandingan parameter yang lainnya. Dalam perbandingan parameter ini yang akan dibandingkan adalah ukuran filter, jumlah data dan *scenario* pelatihan data.

5.6.1 Penentuan Ukuran Filter

Ukuran filter yang sering digunakan pada umumnya yaitu 3x3, 5x5, dan 7x7. Untuk menentukan ukuran filter yang digunakan, percobaan dilakukan dengan menggunakan 2 lapisan konvolusi dengan 32 filter dan 2 konvolusi dengan 64 filter.

Tabel 5.12 *Penentuan Ukuran Filter*

Ukuran Filter	Akurasi Testing
3x3	95%
5x5	93%

Hasil percobaan ditunjukkan pada tabel 5.12. Pada tabel tersebut, ukuran filter yang menghasilkan hasil terbaik adalah 3x3 dengan 95%. Untuk ukuran filter 5x5 didapatkan akurasi sebesar 93%. Seperti yang diketahui menggunakan ukuran filter lebih sedikit memang membuat kita memiliki banyak parameter tetapi kadang ada kasus dimana dengan sedikit parameter lebih baik. Sedangkan ukuran filter 7x7 sendiri tidak bisa digunakan pada penelitian ini dikarenakan ukuran gambar yang *relative* kecil dan jumlah *layer* konvolusi cukup banyak yaitu 4 *layer*.

5.6.2 Pengaruh Jumlah Data Terhadap Akurasi

Peneliti melakukan percobaan terhadap jumlah data yang digunakan pada pelatihan CNN, dengan menggunakan 90, 150 dan 300 data. *Scenario* pelatihan yang digunakan yaitu 80% data *training* dan 20% data *testing*. Setelah dilakukan proses training didapatkan hasil seperti pada tabel 5.13

Tabel 5.13 *Pengaruh Jumlah Data Terhadap Akurasi*

Jumlah Data	Akurasi Testing
90	89%
150	90%
300	95%

Dari percobaan yang dilakukan dapat dilihat bahwa semakin banyak data yang digunakan maka akurasinya semakin besar, ini disebabkan karena komputer belajar dengan lebih banyak gambar atau data yang diberikan.

5.6.3 Pengaruh Skenario Pelatihan Data

Perbandingan *scenario* data *training* dan *testing* dilakukan untuk melihat akurasi terbaik dari masing – masing *scenario*. Jumlah data yang digunakan yaitu 300 karena dari hasil pengujian sebelumnya memiliki akurasi yang paling tinggi. Hasil dari skenario yang dibuat ditunjukkan pada tabel 5.4 berikut.

Tabel 5.14 Skenario Pelatihan dan Pengujian Data

Jumlah Data	Skenario Data Training : Data Testing	Jumlah Data Training : Data Testing	Akurasi Testing
300	60% : 40%	180:120	84%
300	70% : 30%	210:90	87%
300	80% : 20%	240:60	95%

Dari percobaan yang dilakukan didapatkan bahwa dengan 80% training dan 20% testing memiliki tingkat akurasi yang paling tinggi. Ini disebabkan proses pembelajaran lebih banyak dari *scenario* yang lainnya sehingga akurasi yang didapat juga cukup baik.

Kemudian setelah didapatkan model terbaik yaitu dengan akurasi 95%, peneliti ingin menguji model menggunakan data baru. Gambar yang digunakan berjumlah 3 gambar dengan 1 gambar masing-masing kategori (*ballpoint*, penghapus, dan penggaris). Gambar baru yang digunakan harus melalui proses preprocessing terlebih dahulu diantaranya meresize dan mengubah gambar ke RGB. Kemudian memasukkan gambar kedalam model yang telah dibuat, sehingga didapatkan hasil *output* dibawah ini

```
> dinda
      Eraser      Pen      Ruler Predicted_class Actual
[1,] 9.807365e-01 2.219175e-20 1.926346e-02         0         0
[2,] 1.101458e-13 1.000000e+00 8.660677e-33         1         1
[3,] 9.999977e-01 3.498157e-07 1.961408e-06         0         2
```

Gambar 5.19 Output Prediksi Data Baru

Dari *output* diatas dapat dilihat prediksi yang didapat bahwa dari 3 gambar yang dimasukkan komputer salah menebak 1 gambar yaitu pada kategori penggaris, dimana komputer menebak penghapus.

Dari hasil model diatas dapat dilihat jika penerapan metode ini untuk klasifikasi cukup baik sehingga penggunaan untuk membedakan alat tulis sebagai alat membantu pemilik toko cukup baik. Sehingga penelitian ini bisa digunakan untuk membantu mengklasifikasikan berbagai alat tulis khususnya ballpoint, penghapus dan penggaris.

BAB VI

PENUTUP

6.1 Kesimpulan

Berdasarkan hasil analisis yang telah dilakukan, diperoleh beberapa kesimpulan sebagai berikut :

1. Implementasi metode *Convolutional Neural Network (CNN)* untuk klasifikasi alat tulis dilakukan menggunakan *package Keras* pada *software RStudio* versi 1.1.383. Banyaknya *layer* konvolusi yang digunakan yaitu sebanyak empat *layer* konvolusi
2. Tingkat akurasi data *testing* yang didapatkan dari model yang terbentuk yaitu sebesar 95% dalam melakukan klasifikasi alat tulis.
3. Hasil klasifikasi dari data baru sebanyak 3 gambar masing – masing 1 variabel untuk menguji model yang terbentuk yaitu terdapat 1 gambar yang salah pengklasifikasian

6.2 Saran

Berdasarkan kesimpulan di atas, dapat diberikan beberapa saran sebagai berikut :

1. Diharapkan gambar yang digunakan hanya memuat objek yang diteliti sehingga membuat akurasi semakin tinggi.
2. Diharapkan data yang digunakan pada penelitian selanjutnya lebih banyak dengan kategori yang lebih banyak juga.
3. Spesifikasi komputer juga diharapkan lebih tinggi agar mampu melatih model dengan gambar yang lebih banyak. *Graphics processing unit (GPU)* dan *Random-access memory (RAM)* merupakan salah satu spesifikasi yang diperlukan untuk melatih gambar yang banyak .
4. Perbandingan antara nilai jumlah filter, *learning rate*, *epoch*, *layer dropout*, serta fungsi aktivasi yang digunakan bisa menjadi penelitian lebih lanjut untuk melihat bagaimana pengaruhnya terhadap tingkat akurasi yang diperoleh

5. Untuk penelitian selanjutnya diharapkan menggunakan perbandingan dari beberapa percobaan dengan melakukan perubahan pada model.
6. Untuk penelitian selanjutnya diharapkan melakukan percobaan untuk mendeteksi alat tulis secara *real time*

DAFTAR PUSTAKA

- Aloysius Sigit W. 2011. Pemrograman Web Aplikatif dengan Java. Jakarta. PT. Elex Media Komputindo.
- Alpaydin, E., 2009. Introduction to Machine Learning, Second Edition. London: MIT Press.
- Budi S dan M. Zufar. 2016. *Convolutional Neural Networks untuk Pengenalan Wajah Secara Real-Time*. JURNAL SAINS DAN SENI ITS Vol. 5 No. 2 (2016) 2337-3520 (2301-928X Print)
- Apriliya dkk. 2015. *Penentuan Model Terbaik pada Metode Naive Bayes Classifier dalam Menentukan Status Gizi Balita dengan Mempertimbangkan Independensi Parameter*. JURNAL ITSMART Vol 4. No 1. Juni 2015
- Danukusumo, Kevin Pudi. 2017. *Implementasi Deep Learning Menggunakan Convolutional Neural Network untuk Klasifikasi Citra Candi Berbasis GPU*. Tugas Akhir. Program Studi Teknik Informatika Fakultas Teknologi Industri Universitas Atma Jaya Yogyakarta
- Erlangung, Zur. 2007. *Face Image Analysis With Convolutional Neural Networks*. Disertasi, Breisgan, Universitas Freiburg
- Fajri, N., 2011, *Prediksi Suhu dengan Menggunakan Algoritma-Algoritma yang Terdapat pada Artificial Neural Network*, Thesis, Bandung, Indonesia, Institut Teknologi Bandung.
- Farsiah Iaina, Taufik fuadi A, dan Khairul Munadi. 2013. *Klasifikasi Gambar Berwarna Menggunakan K-Nearest Neighbor dan Support Vector Machine*. SNASTIKOM 2013
- Gilbert, Daniel. 2016. *Convolutional Neural Networks for Malware Clasification*. Thesis, Barcelona, Spanyol, Universitas Barcelona
- Goodfellow dkk , 2016, Deep Learning, <http://www.deeplearningbook.org/>, diakses 30 Januari 2018.
- Gummesson, Anna. 2016. Prostate Cancer Clasification Using *Convolutional Neural*. Thesis, Lund, Swedia, Universitas Lund
- H. A. Simon. 1987. Sistem pakar teori dan aplikasi. Andi Yogyakarta : Yogyakarta.
- Han, Jiawei dan Kamber, Micheline. 2006. Data Mining: Concepts dan Techniques, Second Edition. Morgan Kaufmann Publishers, San Francisco
- Haykin. S. 2009. Nueral Networks and Learning Machines. United State of America: Pearson.
- Hermawan, A., 2006, Jaringan Saraf Tiruan: Teori dan Aplikasi, Andi, Yogyakarta.
- Kusumadewi, S., 2004, Membangun Jaringan Syaraf Tiruan, Yogyakarta: Graha Ilmu.

- Medium. *Convolution Neural Network (CNN)*.. Diakses pada 30 Januari 2018 dari <https://medium.com/@samuelsena/pengenalan-deep-learning-part-7-convolutional-neural-network-cnn-b003b477dc94>
- Medium. *Neural Network*.. Diakses pada 30 Januari 2018 dari <https://medium.com/@samuelsena/pengenalan-deep-learning-8fbb7d8028ac>
- Mohri, dkk. 2012. *Foundations of machine learning*. Cambridge: MIT Press.
- Rich, Elaine, dan Kevin Knight. 1991. *Artificial Intelligence*. McGraw-Hill Inc. New York.
- Rismiyati. 2016. Implementasi Convolution Neural Network Untuk Sortasi Mutu Salak Ekspor Berbasis Citra Digital. Thesis, Yogyakarta, Indonesia, Universitas Gajah Mada.
- School of Computer Science Binus. *Dasar Pemahaman Neural Network*. Diakses pada 28 Januari 2018 dari <https://socs.binus.ac.id/2012/07/26/konsep-neural-network/>
- Serra Xavier. 2017. *Face Recognition Using Deep Learning*. Thesis, Catalonia, Spanyol, Universitas Polytechnic
- Shrinivasacharya , Purohit dan M V Sudhamani. 2013. *An Image Crawler For Content Based Image Retrieval System. International Journal of Research in Engineering and Technology eISSN: 2319-1163 | pISSN: 2321-7308*
- Suartika ,I Wayan, Arya Yudhi Wijaya, dan Rully Soelaiman. 2016. *Klasifikasi Citra Menggunakan Convolutional Neural Network (Cnn) pada Caltech 101*. JURNAL TEKNIK ITS. Vol. 5, No. 1.
- Toni Kun, (2010), *Membuat Website Canggih dengan jQuery untuk Pemula*, Mediakita, Jakarta.
- Wikipedia. *Kecerdasan Buatan*. Diakses pada 28 Januari 2018 dari https://id.wikipedia.org/wiki/Kecerdasan_buatan.
- Wikipedia. *Komputer*. Diakses pada 28 Januari 2018 dari <https://id.wikipedia.org/wiki/Komputer>.
- Wikipedia. *Pembelajaran Mesin*. Diakses pada 28 Januari 2018 dari https://id.wikipedia.org/wiki/Pembelajaran_mesin.
- Wikipedia. *Alat Tulis*.. Diakses pada 30 Januari 2018 dari https://id.wikipedia.org/wiki/Alat_tulis
- Wikipedia. *Web Crawler*. Diakses pada 12 February 2018 dari https://en.wikipedia.org/wiki/Web_crawler

LAMPIRAN

Lampiran 1 *Script Javascript Create Dataset*

```
// pull down jquery into the JavaScript console
var script = document.createElement('script');
script.src =
"https://ajax.googleapis.com/ajax/libs/jquery/2.2.0/jquery.min.js";
document.getElementsByTagName('head')[0].appendChild(script);

// grab the URLs
var urls = $('.rg_di .rg_meta').map(function() { return
JSON.parse($(this).text()).ou; });

// write the URLs to file (one per line)
var textToSave = urls.toArray().join('\n');
var hiddenElement = document.createElement('a');
hiddenElement.href = 'data:attachment/text,' +
encodeURIComponent(textToSave);
hiddenElement.target = '_blank';
hiddenElement.download = 'urls.txt';
hiddenElement.click();
```

Lampiran 2 Script Python Crawling Data

```
# import the necessary packages
from imutils import paths
import argparse
import requests
import cv2
import os

# construct the argument parse and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-u", "--urls", required=True,
                help="path to file containing image URLs")
ap.add_argument("-o", "--output", required=True,
                help="path to output directory of images")
args = vars(ap.parse_args())

# grab the list of URLs from the input file, then initialize the
# total number of images downloaded thus far
rows = open(args["urls"]).read().strip().split("\n")
total = 0

# loop the URLs
for url in rows:
    try:
        # try to download the image
        r = requests.get(url, timeout=60)

        # save the image to disk
        p = os.path.sep.join([args["output"], "{}.jpg".format(
            str(total).zfill(5))])
        f = open(p, "wb")
        f.write(r.content)
```

```

f.close()

# update the counter
print("[INFO] downloaded: {}".format(p))

total += 1

# handle if any exceptions are thrown during the download process
except:

    print("[INFO] error downloading {}...skipping".format(p))

# loop over the image paths we just downloaded
for imagePath in paths.list_images(args["output"]):

    # initialize if the image should be deleted or not
    delete = False

    # try to load the image
    try:

        image = cv2.imread(imagePath)

        # if the image is `None` then we could not properly load it
        # from disk, so delete it
        if image is None:

            print("None")

            delete = True

    # if OpenCV cannot load the image then the image is likely
    # corrupt so we should delete it
    except:

        print("Except")

        delete = True

# check to see if the image should be deleted
if delete:

    print("[INFO] deleting {}".format(imagePath))

    os.remove(imagePath)

```

Lampiran 3 Script R Persiapan Gambar

```
# Convolutional Neural Networks

# packages

library(EBImage)

library(keras)


# Read Images

images <- list.files()

images

summary(images)

list_of_images = lapply(images, readImage)

display(list_of_images[[15]])

summary(list_of_images)

print(list_of_images[[10]])


#create train

train <- list_of_images[c(1:80, 101:180,201:280)]

str(train)

train

display(train[[5]])


#create test

test <- list_of_images[c(81:100,181:200,281:300)]

test

display(test[[10]])

#plot

par(mfrow = c(6, 10))

for (i in 1:60) plot(test[[i]])


# Resize

for (i in 1:240) {train[[i]] <- resize(train[[i]], 32, 32)}

for (i in 1:240) {train[[i]] <- toRGB(train[[i]])}
```

```

str(train)

for (i in 1:60) {test[[i]] <- resize(test[[i]],32, 32)}

for (i in 1:60) {test[[i]] <- toRGB(test[[i]])}

str(test)

#cek dimensi

for (f in 1:240) {print(dim(train[[f]]))}

display(train[[10]])


# combine

train <- combine(train)

str(train)

x <- tile(train, 20)

display(x, title='train')

test <- combine(test)

y <- tile(test, 10)

display(y, title = 'test')

str(test)


# Reorder dimension

str(train)

train <- aperm(train, c(4,1,2,3))

str(train)

test <- aperm(test, c(4,1,2,3))

str(test)

display(test)

# Response

trainy <- c(rep(0,80),rep(1,80),rep(2,80))

testy <- c(rep(0,20),rep(1,20),rep(2,20))

# One hot encoding

trainLabels <- to_categorical(trainy)

trainLabels

testLabels <- to_categorical(testy)

testLabels

```

Lampiran 4 Script R Pembuatan dan Training Model

```
# Model
model <- keras_model_sequential()

model %>%
  layer_conv_2d(filters = 32,
                kernel_size = c(3,3),
                activation = 'relu',
                input_shape = c(32, 32,3)) %>%
  layer_conv_2d(filters = 32,
                kernel_size = c(3,3),
                activation = 'relu') %>%
  layer_max_pooling_2d(pool_size = c(2,2)) %>%
  layer_dropout(rate = 0.0001) %>%
  layer_conv_2d(filters = 64,
                kernel_size = c(3,3),
                activation = 'relu') %>%
  layer_conv_2d(filters = 64,
                kernel_size = c(3,3),
                activation = 'relu') %>%
  layer_max_pooling_2d(pool_size = c(2,2)) %>%
  layer_dropout(rate = 0.0001) %>%
  layer_flatten() %>%
  layer_dense(units = 256, activation = 'relu') %>%
  layer_dropout(rate=0.0001) %>%
  layer_dense(units =3, activation = 'softmax') %>%
  compile(loss = 'categorical_crossentropy',
          optimizer = optimizer_sgd(lr = 0.01,
                                    decay = 1e-6,
                                    momentum = 0.9,
                                    nesterov = T),
          metrics = c('accuracy'))

summary(model)
```

```
# Fit model
history <- model %>%
  fit(train,
       trainLabels,
       epochs = 1000,
       batch_size = 30,
       callbacks = callback_tensorboard("logs/run_2"),
       validation_split = 0.2,
       validation_data = list(test, testLabels))
plot(history)
tensorboard("logs/run_2")
```

Lampiran 5 Script R Evaluate and Prediction

```
# Evaluation & Prediction - train data
model %>% evaluate(train, trainLabels)
pred <- model %>% predict_classes(train)
table(Predicted = pred, Actual = trainy)
prob <- model %>% predict_proba(train)
cbind(prob, Predicted_class = pred, Actual = trainy)

# Evaluation & Prediction - test data
model %>% evaluate(test, testLabels)
pred <- model %>% predict_classes(test)
table(Predicted = pred, Actual = testy)
prob <- model %>% predict_proba(test)
cbind(prob, Predicted_class = pred, Actual = testy)

#save model
save_model_hdf5(model,filepath =
'E:/r/fixkeras/download_gambar/images/dinda0.95baru.hdf5')

model<- load_model_hdf5(filepath =
'E:/r/fixkeras/download_gambar/images/dinda0.95baru.hdf5')

###data baru
model<- load_model_hdf5(filepath =
'E:/r/fixkeras/download_gambar/images/dinda0.85baru.hdf5')

# Read Images
images <- list.files()
images
summary(images)
img = lapply(images[-1], readImage)
head(img)
display(img[[1]])
```

```

# Get the image as a matrix
for (i in 1:3) {img[[i]] <- resize(img[[i]], 32, 32)}
for (i in 1:3) {img[[i]] <- toRGB(img[[i]])}
display(img[[1]])

#combine
fixx <- combine(img)
y <- tile(fixx, 3)
display(y, title = 'Pics')
str(fixx)
got <- aperm(fixx, c(4, 1, 2, 3))
str(got)
testyg <- c(0,1, 2)

# One hot encoding
testLabels <- to_categorical(testyg)
model %>% evaluate(got, testLabels)
pred <- model %>% predict_classes(got)
table(Predicted = pred, Actual = testyg)

prob <- model %>% predict_proba(got)
colnames(prob) <- c("Eraser", "Pen", "Ruler")
prob
dinda <- cbind(prob, Predicted_class = pred, Actual = testyg)
dinda

```