

**PENERAPAN *IMAGE CLASSIFICATION*  
DENGAN *PRE-TRAINED* MODEL *MOBILENET*  
DALAM *CLIENT-SIDE MACHINE LEARNING***

**Skripsi**



Oleh:

Farid Evan Ramadhan

11150910000081

**PROGRAM STUDI TEKNIK INFORMATIKA  
FAKULTAS SAINS DAN TEKNOLOGI  
UNIVERSITAS ISLAM NEGERI SYARIF HIDAYATULLAH  
JAKARTA  
2020 M / 1441 H**

**PENERAPAN *IMAGE CLASSIFICATION*  
DENGAN PRE-TRAINED MODEL MOBILENET  
DALAM *CLIENT-SIDE* MACHINE LEARNING**

**Skripsi**

Diajukan sebagai salah satu syarat untuk memperoleh gelar

Sarjana Komputer (S.Kom)



Oleh:

Farid Evan Ramadhan

11150910000081

**PROGRAM STUDI TEKNIK INFORMATIKA  
FAKULTAS SAINS DAN TEKNOLOGI  
UNIVERSITAS ISLAM NEGERI SYARIF HIDAYATULLAH  
JAKARTA  
2020 M / 1441**

**LEMBAR PERSETUJUAN**  
**PENERAPAN *IMAGE CLASSIFICATION* DENGAN PRE-TRAINED**  
**MODEL MOBILENET**  
**DALAM *CLIENT-SIDE* MACHINE LEARNING**

Skripsi

Sebagai Salah Satu Syarat Untuk Memperoleh  
Gelar Sarjana Komputer (S.Kom)

Oleh :

FARID EVAN RAMADHAN  
11150910000081

Menyetujui,

Pembimbing I

Pembimbing II

**Dr. Imam Marzuki Shofi, M.T**  
NIP. 197202052008011010

**Anif Hanifa Setianingrum, M.Si**  
NIDN. 0410116402

Mengetahui,  
Ketua Program Studi Teknik Informatika

**Dr. Imam Marzuki Shofi, M.T**  
NIP. 197202052008011010

## LEMBAR PENGESAHAN

Skripsi yang berjudul “Penerapan *Image Classification* Dengan Pre Trained Model Mobilenet Dalam *Client-Side* Machine Learning” telah diujikan dalam sidang munaqasyah Fakultas Sains dan Teknologi UIN Syarif Hidayatullah Jakarta pada --- 2019. Skripsi ini telah diterima sebagai salah satu syarat memperoleh gelar Sarjana Komputer (S.Kom) pada program Studi Teknik Informatika

Jakarta, 2020

Tim penguji,

Penguji I

Penguji II

**Nama dosen I**  
NIP.

**Nama dosen II**  
NIP.

Tim Pembimbing,

Pembimbing I

Pembimbing II

**Dr. Imam Marzuki Shofi, M.T.**  
NIP. 197202052008011010

**Anif Hanifa Setianingrum, M.Si**  
NIDN. 0410116402

Mengetahui,

Dekan

Ketua Prodi Teknik Informatika

**Prof. Dr. Lily Surayya Eka Putri**  
**M.Env.Stud**  
NIP. 196904042005011003

**Dr. Imam Marzuki Shofi, M.T.**  
NIP. 197202052008011010

## **PERNYATAAN ORISINALITAS**

Dengan ini saya menyatakan bahwa:

1. Skripsi ini merupakan hasil karya asli saya yang diajukan untuk memenuhi salah satu persyaratan memperoleh gelar strata 1 di UIN Syarif Hidayatullah Jakarta.
2. Semua sumber yang saya gunakan dalam penulisan ini telah saya cantumkan sesuai dengan ketentuan yang berlaku di UIN Syarif Hidayatullah Jakarta.
3. Jika di kemudian hari terbukti bahwa karya ini bukan hasil karya asli saya atau merupakan hasil jiplakan dari karya orang lain, maka saya bersedia menerima sanksi yang berlaku di UIN Syarif Hidayatullah Jakarta.

Jakarta, 2020

Farid Evan Ramadhan

## **PERNYATAAN PERSETUJUAN PUBLIKASI SKRIPSI**

Sebagai civitas akademik UIN Syarif Hidayatullah Jakarta, saya yang bertanda tangan di bawah ini:

Nama : Farid Evan Ramadhan

NIM : 11150910000081

Program Studi : Teknik Informatika

Fakultas : Sains dan Teknologi

Jenis Karya : Skripsi

Demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Islam Negeri Syarif Hidayatullah Jakarta Hak Bebas Royalti Noneksklusif (*Non-exclusive Royalty Free Right*) atas karya ilmiah saya yang berjudul:

### **Penerapan *Image Classification* Dengan Pre-Trained Model Mobilenet Dalam *Client-Side* Machine Learning**

Dengan Hak Bebas Royalti Non eksklusif ini Universitas Islam Negeri Syarif Hidayatullah Jakarta berhak menyimpan, mengalih media/formatkan, mengelola dalam bentuk pangkalan data (database), merawat, dan mempublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Ciputat, 2020

(Farid Evan Ramadhan)

**Penulis** : Farid Evan Ramadhan  
**Program Studi** : Teknik Informatika  
**Judul** : Penerapan *Image Classification* Dengan *Pre-Trained Model Mobilenet* Dalam *Client-Side Machine Learning*

### ABSTRAK

Data menjadi sebuah aset yang berharga saat ini. Dengan banyaknya data, teknologi saat ini bisa membuka kemungkinan untuk mengolah dan membuat prediksi. *Machine Learning* (ML) adalah bagian dari *Artificial Intelligence* (AI) yang telah menjadi alat penting dalam sistem perangkat lunak dalam mengoptimalkan. *Image Classification* menjadi salah satu topik dari *Machine Learning* yang banyak dilakukan. Penerapan *image classification* pada *server-side* sudah sering kali dilakukan dalam setiap masalah klasifikasi yang kompleks karena di butuhkan daya komputasi yang besar. Banyaknya pengembang JavaScript menjadi salah satu di gunakannya *client-side machine learning* dalam membuat model klasifikasi. Untuk menunjang dilakukannya *machine learning* di dalam *client-side*, adanya *pre-trained* model menjadi salah satu faktor berhasilnya *client-side machine learning* dapat bekerja. Hasil dari penelitian ini adalah membuktikan *client-side* bisa dilakukan dan bisa lebih unggul dari *server-side machine learning* dengan selisih kecepatan *training model* 142 detik dan selisih rata-rata *loss function* sebesar 0,013679. Hasil pengujian dari *client-side machine learning* dengan Server-Side Machine Learning membuktikan bahwa Client-Side jauh lebih unggul dalam kecepatan melatih model sedangkan Server-Side unggul sedikit dalam nilai error yang di hasilkan.

**Kata Kunci** : *Machine Learning, Image Classification, Server-side, Client-side, Loss Function, Training model.*  
**Jumlah Pustaka** : 14 Buku, 13 Jurnal, dan 7 Web  
**Jumlah halaman** : 111 Halaman

**Author** : Farid Evan Ramadhan  
**Study Program** : Teknik Informatika  
**Title** : Application of Image Classification with Mobilenet Pre-Trained Models in Client-Side Machine Learning

#### **ABSTRACT**

Data is a valuable asset at this time. With the amount of data, current technology can be published to process and make predictions. Machine Learning (ML) is a part of Artificial Intelligence (AI) which has become important in optimizing software systems. Image Classification is one of the topics of Machine Learning that is mostly done. Classification of the application image on the server side has often been done on any complicated classification problems because it requires large computing power. The large number of JavaScript developers is one that is used in client-side machine learning in creating classification models. To support the success of machine learning on the client side, there is a pre-trained model to be one of the factors that successful client-side machine learning can work. The results of this study prove the client side can do and can be better than server-side machine learning with a speed difference of 142 seconds training model and the average difference in the loss function of 0.013679. Test results from the client side of machine learning with Server-Side Machine learning prove the client side is far more waiting in the speed of training the model while the Server side is less in the value of the error generated.

**Keywords** : *Machine Learning, Image Classification, Server-side, Client-side, Loss Function, Training model.*

**Bibliography** : 14 Book, 13 Journal, dan 7 Website

**Number of Pages** : 111 Pages



## KATA PENGANTAR

*Assalamu'alaikum Wr. Wb.*

Puji syukur penulis panjatkan kehadiran Allah SWT yang telah melimpahkan rahmat, taufik serta hidayah-Nya sehingga penulis dapat melaksanakan dan menuliskan skripsi ini pada waktu dan tempat yang tepat dan menyelesaikan tugas akhir skripsi dengan baik. Sholawat dan salam penulis haturkan kepada junjungan kita baginda Nabi Muhammad SAW beserta keluarganya, para sahabatnya serta umatnya hingga akhir zaman. Skripsi ini merupakan salah satu tugas akhir wajib bagi mahasiswa sebagai persyaratan untuk mendapatkan gelar Sarjana Komputer (S.Kom) pada program studi Teknik Informatika, Fakultas Sains dan Teknologi, Universitas Islam Negeri Syarif Hidayatullah Jakarta. Sebagai bahan penulisan skripsi ini, penulis melakukannya berdasarkan hasil penelitian, pengembangan aplikasi, dan beberapa sumber literatur. Tak lupa pula penulis ingin mengucapkan banyak terima kasih kepada pihak-pihak terkait lainnya yang telah banyak membimbing penulis dalam melakukan penulisan skripsi ini, karena tanpa bimbingan dan dorongan dari semua pihak, maka penulisan skripsi ini tidak akan berjalan dengan lancar. Selanjutnya penulis menyampaikan ucapan terima kasih kepada :

1. Orang Tua tercinta yang telah memberikan dukungan moril dan materil ke penulis. Tiada tutur kata selain terima kasih kepada ayah, ibu, kakak, abang, adik, dan keponakan – keponakan tersayang dan rasa syukur kepada Allah S.W.T yang telah menitipkan penulis di keluarga yang sangat penulis cintai.
2. Ibu Prof Dr. Lily Surayya Eka Putri M.Env.Stud, selaku Dekan Fakultas Sains dan Teknologi, Universitas Islam Negeri Syarif Hidayatullah Jakarta.
3. Bapak Imam Marzuki Shofi, M.T. selaku Ketua Program Studi Teknik Informatika, dan Pembimbing I penulis yang senantiasa membimbing penulis, dan memotivasi dalam menyelesaikan skripsi ini.
4. Ibu Anif Hanifa Setianingrum, M.Si selaku Dosen Pembimbing II yang senantiasa meluangkan waktu dan memberikan bimbingan, bantuan, semangat dan motivasi dalam menyelesaikan skripsi ini.

5. Seluruh dosen dan staff UIN Jakarta khususnya Fakultas Sains dan Teknologi yang telah memberikan ilmu dan pengalaman yang sangat berharga bagi penulis.
6. Seluruh sahabat-sahabat terbaik dari Teknik Informatika angkatan 2015, khususnya semua anak kelas TI C 2015 yang sudah lulus dan masih berjuang (Rival, Afie, Rifqi, Kunhadji, Ilham, Raihan, Zaenal, Bayu, Alif, Ayu, Azter, Abi, Bima, Chintya, Agung, Dhiyaulhaq, Elda, Fatim, Fazri, Ismail, Azza, Opi, Dieqy, dan Yusran), serta teman-teman TI Angkatan 2015 yang tidak bisa disebutkan satu persatu, Terima kasih atas semangatnya!

Serta seluruh pihak yang tidak dapat disebutkan oleh penulis satu persatu didalam selemba kertas A4. Penulis menyadari bahwa penulisan skripsi ini masih jauh dari kata sempurna. Untuk itu, penulis memohon kritik dan saran yang membangun untuk penulis.

Akhir kata, semoga laporan skripsi ini dapat bermanfaat bagi penulis dan orang banyak.

*Wassalamualaikum, Wr. Wb.*

Jakarta, Januari 2020

Farid Evan Ramadhan

## DAFTAR ISI

LEMBAR PERSETUJUAN.....	ii
LEMBAR PENGESAHAN .....	iii
PERNYATAAN ORISINALITAS .....	iv
PERNYATAAN PERSETUJUAN PUBLIKASI SKRIPSI .....	v
ABSTRAK.....	vi
ABSTRACT.....	vii
KATA PENGANTAR .....	viii
DAFTAR ISI.....	x
DAFTAR GAMBAR .....	xiii
DAFTAR TABEL.....	xvi
DAFTAR GRAFIK.....	xvii
BAB 1 PENDAHULUAN .....	1
1.1    Latar Belakang .....	1
1.2    Rumusan Masalah .....	5
1.3    Tujuan Penelitian.....	5
1.4    Batasan Masalah.....	6
1.5    Manfaat Penelitian.....	6
1.5.1    Bagi Peneliti:.....	6
1.5.2    Bagi Universitas:.....	6
1.6    Sistematika Penelitian .....	7
BAB 2 LANDASAN TEORI DAN TINJAUAN PUSTAKA.....	8
2.1    Artificial Intelligence (AI).....	8
2.1.1 <i>Machine Learning</i> (ML) .....	10
2.2 <i>Deep Learning</i> (DL) .....	13
2.3 <i>Artificial Neural Network</i> (ANN).....	15
2.4 <i>Convolution Neural Network</i> (CNN) .....	18
2.4.1    Convolution Layer .....	19
2.4.2    Rectified Linear Units (ReLU) Activation.....	20
2.4.3    Cross Entropy Loss Function.....	21
2.4.4    Fully Connected Layer .....	22
2.4.5    Fungsi aktivasi Softmax.....	23

2.5	Website .....	24
2.5.1	HTML .....	25
2.5.2	CSS.....	25
2.5.3	Javascript.....	26
2.6	Node.js.....	27
2.6.1	npm.....	28
2.6.2	Express .....	28
2.7	Client-Side Machine Learning .....	29
2.7.1	WebGL.....	30
2.7.2	Asynchronus Function .....	30
2.7.3	Tensorflow.js.....	31
2.7.4	ml5.js.....	32
2.7.5	MobileNet Model .....	33
2.7.6	Transfer Learning.....	35
2.8	Rapid Aplication Development (RAD).....	37
2.8.1	Tahapan Pengembangan Sistem.....	38
2.9	Metode Pengujian Sistem.....	40
2.9.1	Blackbox Testing .....	40
2.9.2	Keuntungan Black Box Testing .....	40
2.10	Tinjauan Pustaka.....	41
BAB 3 METODOLOGI PENELITIAN.....		44
3.1	Metode Pengumpulan Data .....	44
3.2.1	Studi Pustaka.....	44
3.2.2	Studi Literatur .....	44
3.2	Metode Pengembangan Sistem .....	44
3.2.1	Fase <i>Requirment Planning</i> .....	44
3.2.2	Fase <i>Workshop Design</i> .....	45
3.2.3	Fase <i>Implementation</i> .....	45
3.3	Kerangka Berpikir .....	46
BAB 4 IMPLEMENTASI.....		48
4.1	Fase <i>Requirment Planning</i> .....	48
4.1.1	Analisis Sistem Yang Berjalan .....	48

4.1.2	Identifikasi Masalah .....	49
4.1.3	Analisis Sistem Usulan .....	50
4.1.3.1	Analisis Fitur.....	51
4.2	Fase <i>Workshop Design</i> .....	53
4.2.1	Perancangan Model Klasifikasi Gambar Dengan Pre Trained Model MobileNet .....	53
4.2.2	Perancangan Arsitektus Sistem .....	67
4.2.3	Perancangan UML .....	69
4.2.4	Perancangan <i>User Interface</i> .....	83
4.2.5	Pengkodean .....	87
4.3	Fase Implementasi .....	92
BAB 5 HASIL DAN PEMBAHASAN.....		95
5.1	Hasil Tampilan <i>User Interface</i> .....	95
5.2	Pengujian Sistem Dengan Skenario .....	96
5.3	Perbandingan Client-Side dan Server-side Machine Learning .....	102
BAB 6 KESIMPULAN DAN SARAN .....		109
6.1	Kesimpulan.....	109
6.2	Saran .....	110
DAFTAR PUSTAKA .....		111

## DAFTAR GAMBAR

Gambar 1.1 Survey Stack Overflow (Stack Overflow, 2018) .....	2
Gambar 2.1 Turing test <a href="https://cdn.ttgtmedia.com/rms/onlineImages/crm-turing_test.jpg">https://cdn.ttgtmedia.com/rms/onlineImages/crm-turing_test.jpg</a> .....	8
Gambar 2.2 Diagram venn yang menunjukkan tingkatan dalam AI (Goodfellow et al., 2016).....	9
Gambar 2.3 Cabang Studi Machine Learning <a href="https://i.vas3k.ru/7vx.jpg">https://i.vas3k.ru/7vx.jpg</a> .....	11
Gambar 2.4 Arsitektur neural network (Sudarsono, 2016) .....	15
Gambar 2.5 Model neuron sederhana (Sudarsono, 2016).....	16
Gambar 2.6 Single layer neural network (Hermawan, 2006) .....	17
Gambar 2.7 Multiple layer (Hermawan, 2006).....	17
Gambar 2.8 Competitive layer (Hermawan, 2006).....	18
Gambar 2.9 Arsitektur CNN (Aurélien, 2017).....	19
Gambar 2.10 Convolutional layer (Aurélien, 2017) .....	20
Gambar 2.11 Convolutional layer dengan <i>feature map</i> (Aurélien, 2017).....	20
Gambar 2.12 ReLU (Agarap, 2018).....	21
Gambar 2.13 Contoh fully connected layer pada MNIST (Uniqtech, 2018) .....	22
Gambar 2.14 Softmax Activation (Uniqtech, 2018) .....	24
Gambar 2.15 Ilustrasi kerja website (IdHost, 2018) .....	25
Gambar 2.16 Arsitektur Tensorflow.js (Smilkov et al., 2019) .....	31
Gambar 2.17 Konvolusi standard (a) dibagi menjadi dua lapisan: depthwise convolution (b) dan pointwise convolution (c) untuk membuat filter terpisah secara mendalam (depthwise) (Howard et al., 2017).....	34
Gambar 2.18 Arsitektur MobileNet (Howard et al., 2017) .....	35
Gambar 2.19 Hasil penelitian Yosinski mengenai <i>transfer learning</i> (Yosinski et al., 2014).....	36
Gambar 2.20 Siklus RAD (Kendall & Kendall, 2010) .....	38
Gambar 3.1 Kerangka berfikir (Peneliti) .....	47
Gambar 4.1 server-side machine learning.....	48
Gambar 4.2 Client-side machine learning usulan .....	51
Gambar 4.3 Block Diagram Sistem .....	53
Gambar 4.4 Flowchart Klasifikasi Gambar .....	54
Gambar 4.5 Akses Kamera .....	55
Gambar 4.6 (1) Menambahkan Label. (2) Menambahkan Gambar .....	56

Gambar 4.7 Sebelum dan sesudah <i>resize</i> .....	56
Gambar 4.8 (1) Hasil <i>Standard Convolution</i> . 1 (2) Hasil <i>Depthwise</i> dan <i>Pointwise Convolution</i> .....	59
Gambar 4.9 Matriks pada gambar .....	60
Gambar 4.10 <i>Convolution</i> dengan Stride 2 .....	60
Gambar 4.11 Hasil <i>Convolution</i> .....	61
Gambar 4.12 <i>Convolution</i> ke 13 pada MobileNet .....	61
Gambar 4.13 penerapan <i>flattening</i> .....	62
Gambar 4.14 Rancangan <i>Fully Connected layer</i> pada penelitian .....	64
Gambar 4.15 Hasil prediksi model klasifikasi .....	66
Gambar 4.16 File Hasil Ekspor Model Klasifikasi .....	67
Gambar 4.17 Arsitektur sistem .....	67
Gambar 4.18 Flowchart Aplikasi .....	69
Gambar 4.19 <i>Usecase Diagram</i> .....	70
Gambar 4.20 Activity Diagram Create Model .....	78
Gambar 4.21 Activity Diagram Input Image .....	79
Gambar 4.22 Activity Diagram Train Dataset .....	80
Gambar 4.23 Activity Diagram Predict .....	81
Gambar 4.24 Activity Diagram Save Model .....	82
Gambar 4.25 Activity Diagram Load Model .....	83
Gambar 4.26 Desain <i>Interface</i> Halaman <i>Home</i> .....	84
Gambar 4.27 Desain <i>Interface</i> Halaman <i>Create Model</i> .....	85
Gambar 4.28 Desain <i>Interface</i> Halaman Create Model - Input Image .....	86
Gambar 4.29 Desain <i>Interface</i> Halaman <i>Predict</i> .....	87
Gambar 4.30 Dashboard Heroku .....	92
Gambar 5.1 Tampilan Halaman <i>Home</i> .....	95
Gambar 5.2 Tampilan Halaman <i>Create Model</i> .....	95
Gambar 5.3 Tampilan Halaman <i>Predict</i> .....	96
Gambar 5.4 Gambar pada skenario .....	97
Gambar 5.5 Contoh Dataset .....	103
Gambar 5.6 <i>import dependencies</i> .....	104
Gambar 5.7 import MobileNet .....	105

Gambar 5.8 menyiapkan model dan pengaturan.....	106
Gambar 5.9 Melakukan training pada model. ....	106
Gambar 5.10 Hasil <i>training</i> per epoch.....	107



## DAFTAR TABEL

Tabel 2.1 Perbandingan Literatur.....	42
Tabel 4.1 Tabel Analisis Fitur.....	52
Tabel 4.2 Tabel Aktor .....	69
Tabel 4.3 <i>Usecase Scenario Create Model</i> .....	71
Tabel 4.4 <i>Usecase Scenario Input Image</i> .....	72
Tabel 4.5 <i>Usecase Train Dataset</i> .....	73
Tabel 4.6 <i>Usecase Predict</i> .....	74
Tabel 4.7 <i>Usecase Save Model</i> .....	75
Tabel 4.8 <i>Usecase Load Model</i> .....	76
Tabel 4.9 Tabel Hasil Pengujian .....	92
Table 5.1 Data Percobaan .....	96
Tabel 5.1 Hasil Perbandingan Error.....	108
Tabel 5.2 Hasil Perbandinagn Waktu (detik).....	108

## DAFTAR GRAFIK

Grafik 4.1 <i>Loss Function</i> pada percobaan sistem .....	65
Grafik 5.1 Skenario pada 2 label.....	98
Grafik 5.2 Skenario pada 4 label.....	98
Grafik 5.3 Pengaruh <i>epoch</i> terhadap <i>loss function</i> .....	99
Grafik 5.4 Skenario pada 2 label.....	99
Grafik 5.5 Skenario pada 4 label.....	100
Grafik 5.6 Pengaruh <i>learning rate</i> terhadap <i>loss function</i> .....	100
Grafik 5.7 Skenario pada 2 label.....	101
Grafik 5.8 Skenario pada 4 label.....	101
Grafik 5.9 Pengaruh banyak gambar terhadap <i>loss function</i> .....	102

# **BAB 1**

## **PENDAHULUAN**

### **1.1 Latar Belakang**

Data menjadi sebuah aset yang berharga saat ini. Dengan banyaknya data, teknologi saat ini bisa membuka kemungkinan untuk mengolah dan membuat prediksi berdasarkan informasi yang di peroleh oleh data-data tersebut menjadi sesuatu yang bernilai dan menguntungkan bagi beberapa pihak. Data tersebut juga bisa menjadi suatu alat untuk membantu manusia di bidang kesehatan, bencana, identifikasi dan hampir semua aspek kehidupan (Goodfellow, Bengio, & Courville, 2016). Menurut lembaga riset teknologi Gartner, *Augmented analytics* menjadi trend teknologi nomor 2 setelah trend nomor 1 yaitu *Autonomous thing* tahun 2019. *Augmented analytics* menggunakan *Machine Learning* dan *Natural Language Processing* untuk mengotomatisasi analisis data dan presentasi pengetahuan. Gartner adalah perusahaan riset dan penasihat global yang memberikan wawasan, saran, dan alat untuk para pemimpin di bidang TI, Keuangan, SDM, Layanan dan Dukungan Pelanggan, Hukum, Pemasaran, Penjualan, dan fungsi Rantai Suplai di seluruh Dunia. Menurut Gartner pada tahun 2020, lebih dari 40% pekerjaan dalam bidang data science akan menjadi otomatis di tahun tersebut (Gartner & Panetta, 2018).

*Machine Learning* telah menaklukkan industry dan sekarang menjadi jantung dari banyak produk berteknologi tinggi saat ini, salah satunya yaitu memberi peringkat pada hasil pencarian *website*, memungkinkan menggunakan pengenalan suara pada ponsel cerdas, merekomendasikan video, bahkan mengalahkan juara dunia dalam permainan Go (Aurélien, 2017). Sering kali *Machine Learning* sulit untuk di pelajari karena sulitnya dalam memahami pengkodean yang dilakukan untuk mengenali pola pada dunia nyata, maka dari itu muncullah banyak *library* atau pustaka yang memudahkan pengembang aplikasi dalam membuat program *machine learning*. *Library machine learning* berk ualitas produksi biasanya ditulis untuk pengembang dengan bahasa Python dan C++. Namun, ada banyak komunitas

*frontend developer* dan *backend developer* dalam Pengembangan *JavaScript* (JS) yang terus tumbuh dengan cepat. Ada 2,3 juta permintaan *pull* (mengunduh dan berkontribusi pada project di *repository*) GitHub di JS pada 2018, dibandingkan dengan 1 juta di Python (GitHub, 2018). Di kutip dari websitenya GitHub sendiri adalah sebuah perusahaan Amerika penyedia *hosting website* dan *software development version control* yang dibuat pada tahun 2007 dan digunakan lebih dari 36 juta *developer* diseluruh dunia untuk mengembangkan perangkat lunaknya. Menurut Survei Pengembang Stack Overflow pada tahun 2018, JS adalah bahasa pemrograman yang paling umum digunakan. Survey yang dilakukan Stack Overflow pada 78,334 programmer, JS unggul di posisi pertama dengan 69,8% (Stack Overflow, 2018). Dikutip dari websitenya, *Stack Overflow* yang didirikan pada 2008 adalah komunitas online terbesar dan paling tepercaya bagi siapa pun yang berniat untuk belajar dan berbagi pengetahuan. Lebih dari 50 juta pengunjung datang ke *Stack Overflow* setiap bulan untuk membantu memecahkan masalah pengkodean, mengembangkan keterampilan baru, dan menemukan peluang kerja.



Gambar 1.1 Survey Stack Overflow  
(Stack Overflow, 2018)

Pada umumnya proses pengolahan *dataset* (kumpulan data mentah) dan *training model* (proses melatih mesin untuk memahami suatu pola) pada *machine learning* di lakukan di *server-side* sedangkan *prediction* yang merupakan tahap akhir biasanya di lakukan di *client-side* atau *interface* pengguna. Keuntungan menggunakan *machine learning* di *client-side* di bandingkan *server-side*

diantaranya adalah *On-device Computation*, yaitu kemampuan untuk melakukan komputasi di dalam perangkat sendiri sehingga data lebih aman tanpa harus di kirim ke server, *Shareability*, dengan menjalankan *machine learning* di browser tanpa tambahan instalasi alat-alat lainnya memudahkan untuk menjalankan *machine learning* di berbagai *platform*, dan *Interactivity*, interaktifitas dari web browser memungkinkan melakukan *machine learning* tanpa mengetahui pengkodean sedikitpun sehingga membuka kesempatan edukasi dan penelitian lebih besar (Smilkov et al., 2019).

Menjalankan aplikasi *machine learning* dengan di *client-side* dan menggunakan *browser* telah menarik perhatian dari berbagai komunitas penelitian termasuk AI, rekayasa perangkat lunak, *web browser*, dan bahkan arsitektur komputer. Sebagai hasilnya, berbagai arsitektur JavaScript berbasis *Deep Learning/Machine Learning* dan library telah dirancang. Pada tahun 2015, Karpathy dan Li merancang ConvNetJS, dikenal sebagai library pertama yang di buat untuk DL/ML di dalam browser (ConvNetJS, 2018). Library lain pun bermunculan seperti *WebDNN*, *Keras.js* dan *Mind*, yang dikhususkan untuk mendukung *Deep Learning* (DL) dalam browser. Pada awal tahun 2018, Google merilis *TensorFlow.js*, sebagai lompatan untuk mempromosikan dan memindahkan DL/ML ke *client-side* atau *browser* dalam jurnalnya yang berjudul “*moving deep learning to browser and how far can we go?*” (Ma, Xiang, Zheng, Tian, & Liu, 2019).

Untuk menyelesaikan masalah klasifikasi yang merupakan bagian dari *supervised learning* maka di butuhkan *dataset*, *model*, serta *library* JS tersebut (Dangeti, 2017). Oleh karena itu penelitian ini menggunakan *pre-trained model* *Mobilnet* dan *ml5.js* yang merupakan Layer API library dari *TensorFlow.js* dapat mempermudah pengkodean *machine learning*. *Pre-trained* model *MobileNet* adalah merupakan salah satu arsitektur *convolutional neural network* (CNN) yang dapat digunakan untuk mengatasi kebutuhan akan *computing resource* berlebih. Di penelitian ini *MobileNet* dipilih karena ukurannya yang kecil, akurasi yang cukup besar dan paling cocok untuk melakukan *image classification* di dalam *browser* berdasarkan penelitian yang dilakukan oleh Google yang berjudul “*MobileNets: efficient CNN for mobile vision applications*” membandingkan *MobileNet* dengan

beberapa *pre-trained* model terkenal lainnya seperti VGG, GoogleNet, Inception V3, dan lain-lain (Howard et al., 2017).

Membuat sebuah model untuk mengklasifikasikan gambar memiliki banyak potensi untuk membuat berbagai macam aplikasi. Klasifikasi gambar sudah bekerja di aplikasi foto, dan bisa menandai foto yang di klasifikasi sebagai wajah atau semacamnya. Klasifikasi Gambar dapat digunakan untuk mengenali sel-sel kanker, untuk mengenali kapal dalam citra satelit, atau untuk secara otomatis mengklasifikasikan gambar pada *Pinterest*. Klasifikasi bahkan dapat digunakan di luar ranah gambar, menganalisis peta panas dari aktivitas manusia untuk mendeteksi potensi pencurian, atau transformasi gelombang audio Fourier (Aurélien, 2017). Dalam penelitian ini peneliti menggunakan *Pre-trained Model* MobileNet yang memiliki dasar CNN yang merupakan salah satu metode *Deep Learning* yang sering digunakan dalam pengklasifikasian gambar.

*Deep Learning* telah menjadi suatu topik yang hangat diperbincangkan di dunia *Machine Learning* karena kapabilitasnya yang signifikan dalam memodelkan berbagai data kompleks seperti citra dan suara. Metode *Deep Learning* yang memiliki hasil paling signifikan didalam pengenalan citra adalah *Convolutional Neural Network* (CNN). Karena metode CNN berusaha meniru cara pengenalan citra pada *visual cortex* yang sama dengan manusia sehingga mampu mengolah informasi yang sama (Arfian, 2018). Contohnya Penelitian yang dilakukan oleh Rismiyati dan Arfian yang berjudul “*Implementasi Convolutional Neural Network Untuk Sortasi Mutu Salak Ekspor Berbasis Citra Digital*” dan “*Implementasi Convolutional Neural Network Terhadap Transportasi Tradisional Menggunakan Keras*” menggunakan metode *Convolutional Neural Network* untuk membedakan salak dan transportasi tradisional, memperoleh akurasi masing-masing sebesar 81,5 % dan 82%, karena itu CNN di gunakan dalam penelitian ini (Rismiyati, 2016).

Penelitian yang menggunakan *client-side machine learning* di lakukan oleh Yiyun Liang dengan judul “*CNNs for NLP in the Browser: Client-Side Deployment and Visualization Opportunities*” yang menjalankan CNN menggunakan JavaScript dan sepenuhnya di browser dengan tujuan mengetahui kemampuan JavaScript dan browser dalam menjalankan *deep learning* dan juga melakukan

visualisasinya di berbagai platform dan web browser (Liang, Tu, Huang, & Lin, 2018). Penelitian lainnya juga dilakukan oleh Aurelia Michel dengan judul “*MobileNet Convolutional Neural Networks and Support Vector Machines for Palmprint Recognition*” yang menggunakan *Pre Trained* model MobileNet dan SVM untuk melakukan identifikasi telapak tangan dengan hasil akurasi 100% (Michele, Colin, & Santika, 2019).

Dari penelitian diatas maka peneliti mencoba membuat sebuah sistem *machine learning* yang bertujuan untuk tugas klasifikasi gambar menggunakan *Transfer Learning* dan *Pre-Trained* model MobileNet menggunakan JavaScript sebagai bahasa pemrogramannya dan ml5 sebagai *library machine learning* di web browser (*client-side*). Peneliti bertujuan untuk membuat sebuah sistem klasifikasi yang dapat melakukan pembuatan model klasifikasi tanpa membutuhkan kode sehingga user dapat membuat model klasifikasinya sendiri. Didalam web tersebut user bisa melatih dan membuat modelnya sendiri dengan mengambil gambar menggunakan webcam browser sebagai masukan data. Dari latar belakang dan penjelasan di atas maka peneliti memilih judul sebagai berikut “**Penerapan Image Classification Dengan Pre-Trained Model MobileNet dalam Client-Side Machine Learning**”.

## 1.2 Rumusan Masalah

Ditinjau dari latar belakang tersebut, maka dapat dirumuskan permasalahan yang akan dikaji lebih lanjut dalam skripsi ini yaitu Bagaimana menerapkan *Image Classification* dengan *pre-trained* model *MobileNet* dalam *client-side* Machine Learning?

## 1.3 Tujuan Penelitian

Tujuan dari penelitian ini adalah sebagai berikut:

1. Menerapkan Machine Learning di browser sepenuhnya.
2. Mengetahui cara kerja klasifikasi gambar dengan Pretrained model MobileNet.
3. Menunjukkan kelebihan *client-side* dari *server-side machine learning*

#### 1.4 Batasan Masalah

Berdasarkan rumusan masalah yang sudah didapat, maka pada penelitian ini didapat batasan masalah sebagai berikut:

1. Klasifikasi hanya dilakukan di browser chrome desktop/laptop.
2. Machine Learning menggunakan library ML5.
3. Pengklasifikasian hanya menggunakan Pre-trained model MobileNet.
4. Dataset / gambar yang digunakan hanya diambil dari webcam browser.
5. Training dan klasifikasi dilakukan sepenuhnya browser.
6. Aplikasi hanya mengeluarkan nilai *error / loss function* pada model ketika dilakukan *training* pada model.
7. Klasifikasi gambar hanya digunakan pada gambar/benda secara *general* dan tidak bisa dipakai untuk identifikasi hal spesifik seperti penyakit dan lain-lain.

#### 1.5 Manfaat Penelitian

Manfaat yang diharapkan dari penelitian ini adalah sebagai berikut:

##### 1.5.1 Bagi Peneliti:

1. Menerapkan ilmu – ilmu yang diperoleh selama masa kuliah.
2. Memperdalam dan memahami ilmu tentang *machine learning* dan *image classification* di dalam client-side serta menerapkannya dalam kehidupan nyata.

##### 1.5.2 Bagi Universitas:

1. Mengetahui kemampuan mahasiswa dalam menguasai teori yang telah diperoleh selama kuliah.
2. Mengetahui kemampuan mahasiswa dalam menerapkan ilmunya dan sebagai bahan evaluasi.
3. Memberikan gambaran kepada mahasiswa sebagai kesiapannya untuk menghadapi dunia kerja.
4. Menambah referensi tentang penggunaan *machine learning* dan *image classification* di dalam client-side dalam penerapannya pada list pembelian produk.



## 1.6 Sistematika Penelitian

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

### BAB I PENDAHULUAN

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

### BAB II LANDASAN TEORI DAN TINJAUAN PUSTAKA

Bab ini akan diuraikan literatur apa saja yang akan digunakan dalam penelitian ini, serta teori-teori yang berkenaan dan berhubungan dengan penelitian ini.

### BAB III METODOLOGI PENELITIAN

Bab ini akan diuraikan mengenai penyelesaian permasalahan dengan menggunakan metodologi yang dipilih, serta memuat unsur-unsur pengumpulan data, serta pelaksanaan implementasi.

### BAB IV IMPLEMENTASI

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

### BAB V HASIL DAN PEMBAHASAN

Bab ini akan diuraikan mengenai hasil penerapan yang sudah dilakukan, melakukan pencatatan kekurangan dari hasil analisis yang mungkin harus mendapat perhatian.

### BAB VI PENUTUP

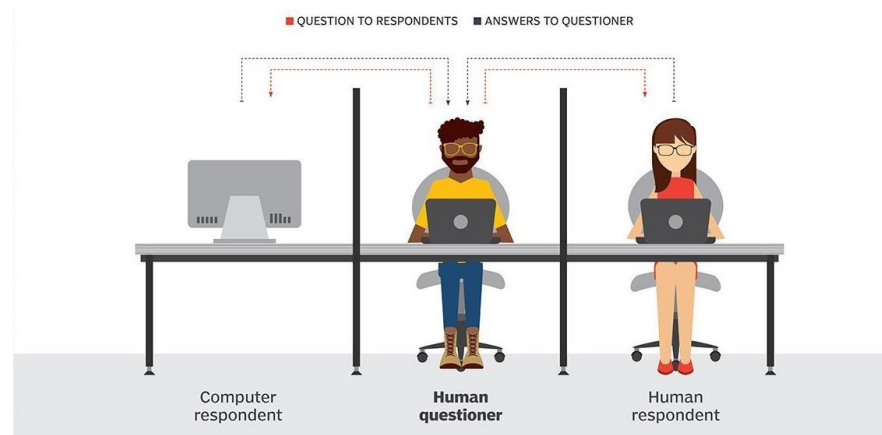
Bab ini akan dibahas mengenai kesimpulan yang diperoleh dari hasil penelitian dan analisis data yang telah dilakukan serta saran-saran yang dapat diterapkan dari hasil pengolahan data yang dapat menjadi masukan yang berguna kedepannya

## BAB 2

### LANDASAN TEORI DAN TINJAUAN PUSTAKA

#### 2.1 Artificial Intelligence (AI)

*Artificial Intelligence* dimulai pada tahun 1950 ketika Alan Turing mengembangkan tes kecerdasan buatan, yang ditujukan untuk ilmuwan komputer yang berupaya menerapkan kecerdasan buatan pada komputer. Tes Turing adalah tes operasional; yaitu, menyediakan cara konkret untuk menentukan apakah entitas (keberadaan) tersebut memiliki kecerdasan. Tes ini melibatkan seorang penguji manusia yang berada di ruangan pertama dan manusia lain di ruangan kedua, dan entitas yang di uji di ruangan ketiga. Lalu penguji di minta untuk berkomunikasi melalui console dengan kedua ruangan lainnya. Jika introgator tidak bisa membedakan entitas dan manusia maka tes berhasil dan entitas dinyatakan memiliki kecerdasan.



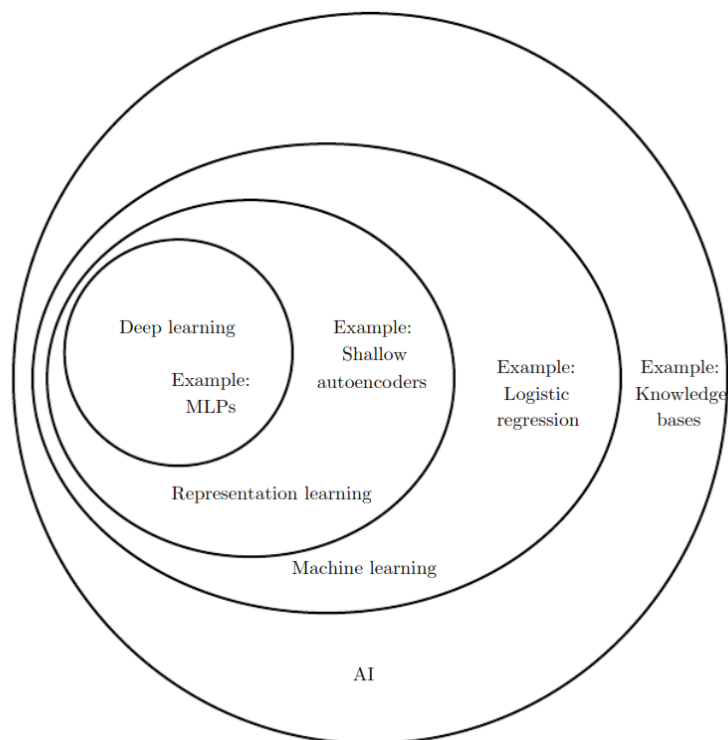
Gambar 2.1 Turing test

[https://cdn.ttgtmedia.com/rms/onlineImages/crm-turing\\_test.jpg](https://cdn.ttgtmedia.com/rms/onlineImages/crm-turing_test.jpg)

*Artificial Intelligence* (AI) menurut Mariusz Flasiński di bukunya yang berjudul *Introduction to Artificial Intelligence* memiliki dua makna dasar. Pertama AI merupakan bidang penelitian umum di bidang ilmu komputer dan robotika, dimana pengembangan sistemnya bisa melakukan tugas-tugas yang

membutuhkan kecerdasan ketika di lakukan oleh manusia yang merupakan tujuan dari penelitian tersebut. Kedua, fitur sistem buatan yang memungkinkan mereka untuk melakukan tugas-tugas yang membutuhkan kecerdasan, ketika dibuat oleh manusia. Jadi, dalam artian ini, kecerdasan buatan bukanlah benda, melainkan properti sistem tertentu, seperti halnya mobilitas adalah properti robot yang memungkinkan mereka bergerak (Flasiński, 2016).

Jadi dapat di simpulkan AI merupakan suatu sistem yang memiliki kecerdasan atau setidaknya dapat menyelesaikan tugas-tugas secara eksplisit dan membutuhkan kecerdasan manusia tanpa harus di program secara spesifik. Salah satu cabang AI adalah *Machine Learning* yang merupakan pokok pembahasan pada penelitian ini. Dibawah ini adalah gambaran besar posisi AI, *Machine Learning*, dan *Deep Learning*.

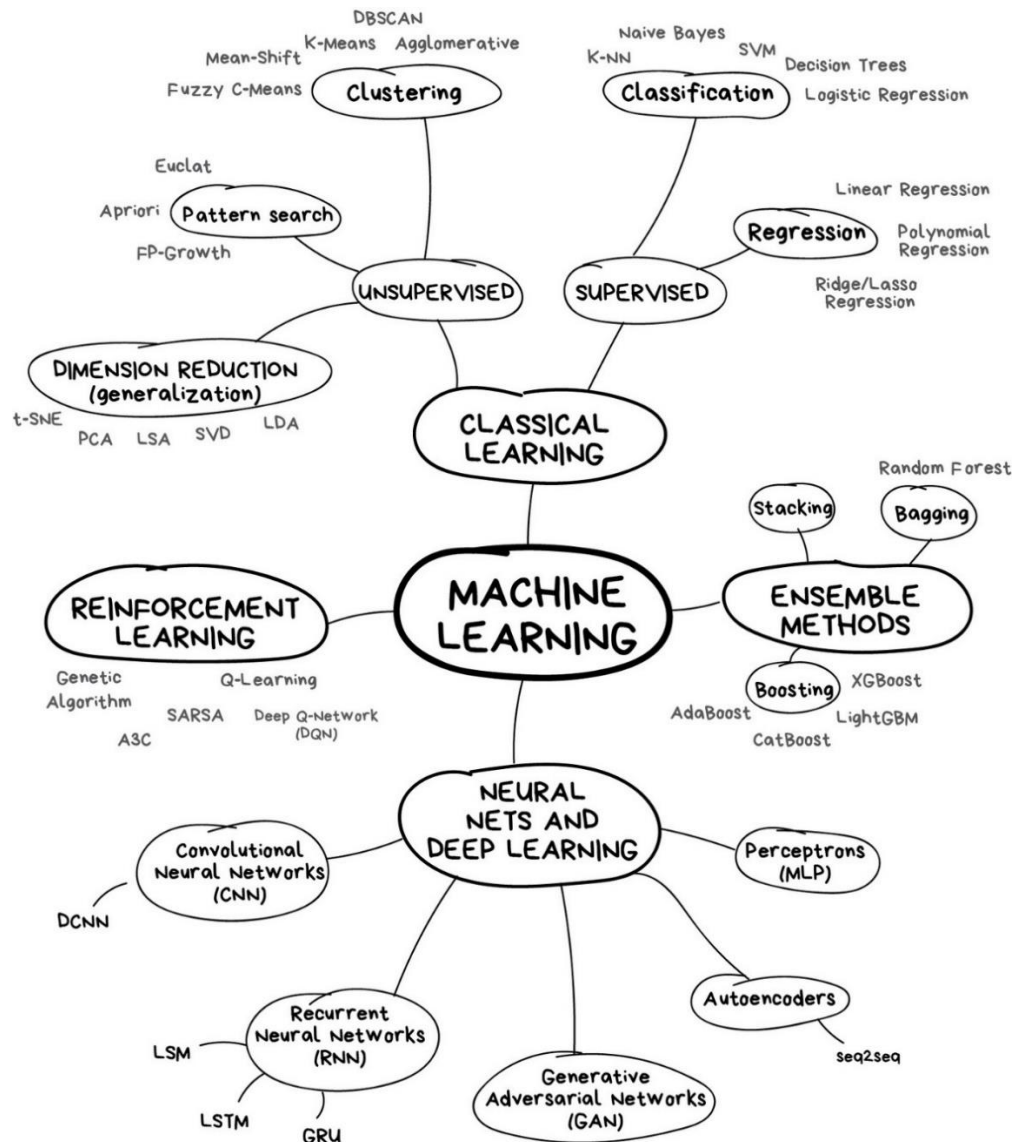


Gambar 2.2 Diagram venn yang menunjukan tingkatan dalam AI (Goodfellow et al., 2016)

### 2.1.1 *Machine Learning* (ML)

*Machine Learning* merupakan program komputer yang belajar dari data yang di berikan. Menurut Arthur Samuel - 1959 *Machine Learning* adalah bidang studi yang memberikan computer kemampuan untuk belajar tanpa secara eksplisit di program. Sementara menurut Tom Mitchell – 1997 Suatu program komputer dikatakan belajar dari pengalaman E sehubungan dengan beberapa tugas T dan beberapa ukuran kinerja P, jika kinerjanya pada T, yang diukur dengan P, meningkat dengan pengalaman E. Contohnya filter spam, yang merupakan program Machine Learning yang dapat belajar menandai spam yang diberikan contoh email spam dan contoh email biasa. Contoh yang digunakan sistem untuk belajar disebut set pelatihan atau *Dataset*. Setiap contoh pelatihan disebut *Trainingset* (atau sampel). Dalam hal ini, tugas T adalah menandai spam untuk email baru, pengalaman E adalah data pelatihan, dan ukuran kinerja P merupakan rasio email yang diklasifikasikan dengan benar. Ukuran kinerja khusus ini disebut akurasi dan sering digunakan dalam tugas klasifikasi (Aurélien, 2017).

Di dalam machine learning terdapat banyak metode dan algoritma yang secara garis besar terbagi menjadi 3 bagian yaitu *Supervised Learning*, *Unsupervised Learning*, dan *Reinforcement Learning*.



Gambar 2.3 Cabang Studi Machine Learning

<https://i.vas3k.ru/7vx.jpg>

#### ▪ Supervised Learning

*Supervised Learning* didasarkan pada pelatihan sampel data dari sumber data dengan klasifikasi yang sudah ditugaskan. Teknik tersebut digunakan dalam model feedforward atau MultiLayer Perceptron (MLP). MLP ini memiliki tiga karakteristik berbeda:

1. Satu atau lebih lapisan neuron tersembunyi yang bukan bagian dari lapisan input atau output jaringan yang memungkinkan jaringan untuk belajar dan menyelesaikan masalah kompleks
2. Nonlinier yang tercermin dalam aktivitas neuron dapat dibedakan dan,
3. Model interkoneksi jaringan menunjukkan tingkat konektivitas yang tinggi. (Sathya & Abraham, 2013)

*Supervised Learning* dapat belajar menyelesaikan masalah rumit melalui latihan atau *training* di dalam sistem syaraf buatan menggunakan data yang sudah di berikan dan dikategorikan atau *labeling*.

- Unsupervised Learning

*Self-Organizing neural networks* belajar menggunakan algoritma *Unsupervised Learning* untuk mengidentifikasi pola-pola tersembunyi dalam data input tidak berlabel. *Unsupervised learning* mengacu pada kemampuan untuk mempelajari dan mengatur informasi tanpa memberikan error untuk mengevaluasi solusi. Karakteristik utama dari Self-Organizing Maps (SOM) adalah:

1. Mengubah pola sinyal yang masuk dari dimensi tidak tetap menjadi peta satu atau 2 dimensi dan melakukan transformasi secara adaptif.
2. Jaringan mewakili struktur feedforward dengan lapisan komputasi tunggal yang terdiri dari neuron yang tersusun dalam baris dan kolom.
3. Pada setiap tahap representasi, setiap sinyal input disimpan dalam konteks yang tepat dan,
4. Neuron yang berhubungan dengan potongan-potongan informasi yang saling berkaitan erat dan mereka

berkomunikasi melalui koneksi sinaptik. (Sathya & Abraham, 2013)

*Unsupervised Learning* dapat mengolah informasi data yang tidak berlabel melalui pola data tersebut. Kadang situasi seperti ini menguntungkan karena algoritma ini dapat melihat pola data yang sebelumnya belum terlihat atau terpikirkan.

- Reinforcemen Learning

Di dalam *Reinforcemen Learning*, algoritma berinteraksi dengan lingkungan dengan menghasilkan urutan tindakan  $a_1, a_2, a_3, \dots, a_n$ ; seiring waktu. Tindakan-tindakan ini mempengaruhi lingkungan, yang menghasilkan hadiah atau hukuman dalam setiap batas waktu  $t$ . Tujuan dari algoritma ini adalah untuk belajar bertindak dengan cara yang kemungkinan akan memaksimalkan beberapa ukuran utilitas di masa depan. Penguatan tidak harus berupa hadiah atau hukuman; melainkan, dapat berupa tanggapan balik yang berguna untuk menentukan tindakan di masa mendatang. (Richard E. Neapolitan, 2018) Dengan kata lain *Reinforcemen Learning* dapat belajar dari kesalahan dan mengandalkan tanggapan balik maupun itu positif atau negatif untuk berkembang, dan biasanya algoritma ini di pakai dalam Game, atau mengotomatiskan suatu perintah seperti mobil otomatis dan lain-lain.

## 2.2 Deep Learning (DL)

Setelah AI menjadi dominan pada 1950-an, jaringan saraf popularitasnya menurun. Namun, algoritma baru untuk pelatihan jaringan saraf dan peningkatan kecepatan pemrosesan komputer secara dramatis mengakibatkan munculnya kembali penggunaan jaring saraf di bidang yang disebut *deep learning* (DL). Arsitektur DL *neural network* berbeda dari jaringan neural lama karena mereka sering memiliki lapisan tersembunyi. Selain itu, jaringan DL dapat dilatih dengan metode *unsupervised* ataupun *supervised learning*. DL telah digunakan untuk menyelesaikan tugas-tugas seperti *computer vision* dan

*voice recognition*, yang sulit dengan pendekatan lainnya. *Deep Learning* merupakan salah satu bidang dari *Machine Learning*. DL menerapkan arsitektur yang sangat kuat karena memanfaatkan jaringan syaraf tiruan yang sangat cocok untuk *supervised learning*, dan dapat menyelesaikan masalah kompleks seperti yang di sebutkan di paragraph sebelumnya. Dengan memanfaatkan *hidden layer* yang banyak deep learning bisa membuat model yang lebih baik dan menghasilkan akurasi yang lebih tinggi dari *neural network* biasa.

Sumber kesulitan utama dalam banyak aplikasi kecerdasan buatan adalah banyak faktor variasi memengaruhi setiap bagian data yang dapat kita amati. Masing-masing piksel dalam gambar mobil merah mungkin sangat dekat dengan hitam di malam hari. Bentuk siluet mobil tergantung pada sudut pandang. Sebagian besar aplikasi mengharuskan kita untuk memisahkan faktor-faktor variasi dan membuang faktor-faktor yang tidak kita pedulikan. Tentu saja, bisa sangat sulit untuk mengekstrak fitur abstrak tingkat tinggi seperti itu dari data mentah. Banyak dari faktor variasi ini, seperti aksen bicara, dapat diidentifikasi hanya dengan menggunakan pemahaman data tingkat manusia yang canggih dan nyaris manusiawi. *Deep Learning* dapat memecahkan masalah sentral ini dalam pembelajaran representasi dengan memperkenalkan representasi yang diekspresikan dalam bentuk representasi lain yang lebih sederhana. Pembelajaran yang mendalam memungkinkan komputer untuk membangun konsep yang kompleks dari konsep yang lebih sederhana seperti sudut dan kontur yang di namakan *Feature Engineering* (Goodfellow et al., 2016).

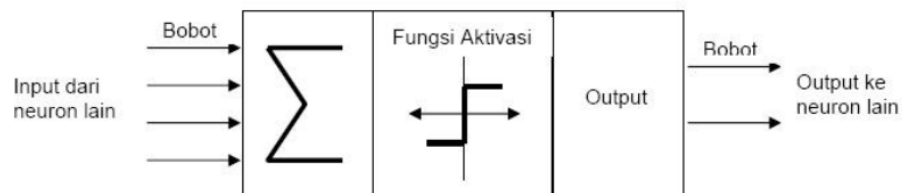
*Feature Engineering* adalah salah satu fitur utama dari Deep Learning untuk mengekstrak pola yang berguna dari data yang akan memudahkan model untuk membedakan kelas. Feature Engineering juga merupakan teknik yang paling penting untuk mencapai hasil yang baik pada tugas prediksi. Namun, sulit untuk 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 cocok dalam menemukan fitur yang



baik pada citra 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).

### 2.3 Artificial Neural Network (ANN)

Beberapa algoritma pembelajaran paling awal yang kita kenali saat ini dimaksudkan sebagai komputasi yang berjalan pada biologis. Yaitu, bagaimana pembelajaran terjadi atau dapat terjadi di otak. Akibatnya, salah satu nama DL adalah *Artificial Neural Network* (ANN). Jaringan syaraf tiruan terinspirasi dari bagaimana jaringan orak berkerja baik manusia maupun hewan. *Neural Network* memiliki berbagai macam tipe, akan tetapi hampir semua komponen yang dimiliki sama. Seperti halnya jaringan syaraf pada otak manusia, neural network juga terdiri dari beberapa neuron yang saling behubungan. Masing-masing neuron akan menerima informasi dan mengakumulasikannya dari neuron yang tersambung sebelumnya. Hubungan ini disebut dengan sebutan bobot (Weight). Informasi tersebut disimpan pada suatu nilai tertentu pada bobot tertentu. Berikut adalah struktur Neuron pada neural network:

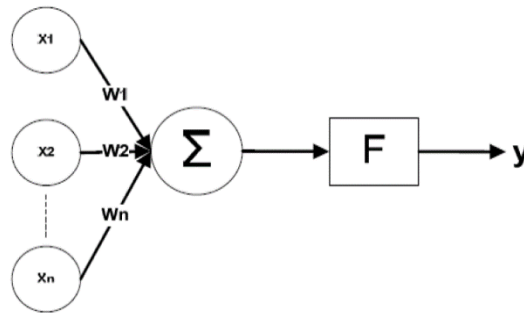


Gambar 2.4 Arsitektur neural network  
(Sudarsono, 2016)

Pada jaringan syaraf, neuron-neuron akan dikumpulkan dalam lapisan – lapisan yang disebut dengan lapisan neuron. Biasanya neuron pada satu lapisan akan dihubungkan dengan lapisan sebelum atau sesudahnya terkecuali lapisan masukan dan lapisan keluaran. Informasi yang diberikan pada jaringan syaraf akan dirambatkan dari lapisan ke lapisan, melalui dari lapisan masukan sampai lapisan keluaran melalui lapisan tersembunyi. Algoritma pembelajaran menentukan informasi akan dirambatkan kearah mana, gambar 2.4

menunjukkan neuron jaringan syaraf sederhana dengan fungsi aktivasi F. Pada gambar 2.5 sebuah neuron akan mengolah N masukan ( $X_1, X_2, X_3, \dots, X_n$ ) yang masing-masing memiliki bobot  $W_1, W_2, W_3, \dots, W_n$  dengan rumus:

$$y_{in} = \sum_{i=1}^n x_i w_i$$

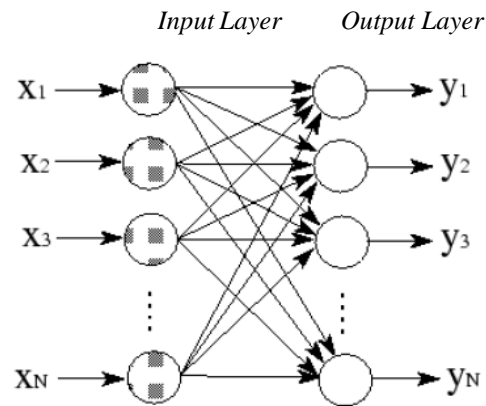


Gambar 2.5 Model neuron sederhana  
(Sudarsono, 2016)

Pada Neural Network, neuron-neuron yang ada pada lapisan yang sama memiliki keadaan yang sama. Terdapat faktor penting dalam menentukan sifat suatu neuron yaitu bobot (Weight) dan penggunaan fungsi aktivasi dari neuron tersebut. Setiap lapisan pada neuron memiliki fungsi aktivasi yang sama. Arsitektur yang dapat dibentuk oleh ANN bermacam-macam yaitu *single layer*, *multiple layer*, dan *competitive layer*. Semakin rumit suatu arsitektur dari NN maka akan semakin rumit dan luas pula masalah yang bisa di kerjakan. Namun terdapat kelemahan yaitu semakin rumit suatu NN maka kebutuhan proses training dan simulasi (testing) yang akan memerlukan waktu lebih lama. Menurut (Hermawan, 2006), Arsitektur neural network dapat dibagi berdasarkan jumlah lapisannya diantaranya:

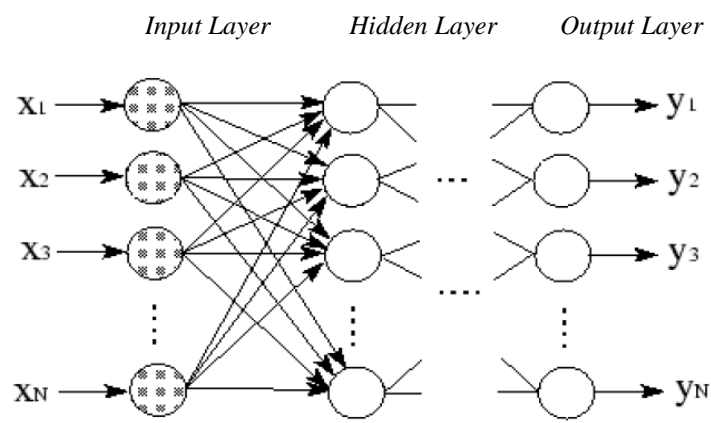
1. *Single Layer Neural Network*: Jaringan dengan lapisan tunggal terdiri dari 1 lapisan input dan 1 lapisan output. Setiap neuron yang terdapat di dalam lapisan input selalu terhubung dengan setiap neuron yang terdapat pada lapisan output. Jaringan ini hanya menerima input kemudian secara

langsung akan mengolahnya menjadi output tanpa harus melalui *hidden layer*.



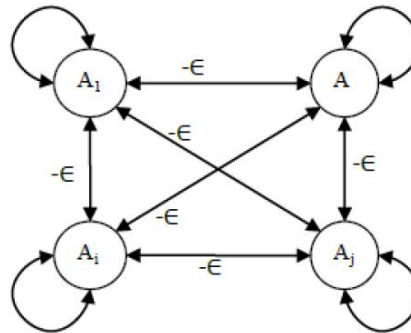
Gambar 2.6 Single layer neural network  
(Hermawan, 2006)

2. *Multiple Layers Neural Network*: memiliki 3 jenis lapisan yakni lapisan input, lapisan output, dan lapisan tersembunyi yang bisa diatur banyaknya. NN yang memiliki banyak lapisan dapat menyelesaikan permasalahan yang lebih kompleks dibandingkan jaringan dengan lapisan tunggal. Akan tetapi, proses pelatihan membutuhkan waktu yang cenderung lama, dan tergantung pada kekuatan *processing* komputer.



Gambar 2.7 Multiple layer  
(Hermawan, 2006)

3. *Competitive Layers*: Pada jaringan ini sekumpulan neuron bersaing untuk mendapatkan hak menjadi aktif. Contoh algoritma yang menggunakan jaringan ini adalah LVQ.



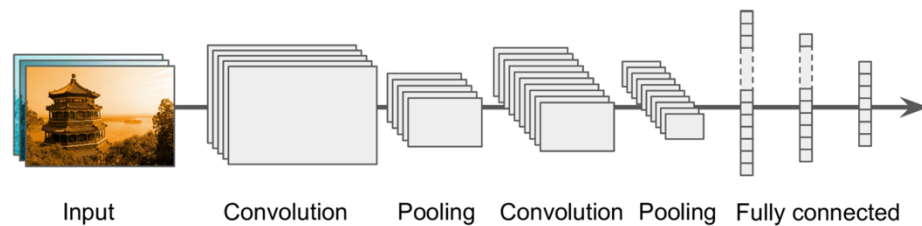
Gambar 2.8 Competitive layer  
(Hermawan, 2006)

## 2.4 Convolution Neural Network (CNN)

Jaringan saraf convolutional (CNN) muncul dari studi korteks visual otak, dan telah digunakan dalam pengenalan gambar sejak 1980-an. Dalam beberapa tahun terakhir, berkat peningkatan daya komputasi, jumlah data yang tersedia, dan Teknik-teknik untuk melatih *deep learning*, CNN telah berhasil meniru bahkan melebihi manusia pada beberapa tugas visual yang kompleks. CNN mendukung layanan seperti pencarian gambar, mobil otomatis, sistem klasifikasi video otomatis, dan banyak lagi. Selain itu, CNN tidak terbatas pada persepsi visual: CNN juga berhasil pada tugas-tugas lain, seperti pengenalan suara atau pemrosesan bahasa (NLP).

Arsitektur CNN standarnya menumpuk beberapa lapisan konvolusional (masing-masing umumnya diikuti oleh layer ReLU), kemudian *pooling layer*, lalu beberapa lapisan konvolusional lainnya (+ ReLU), lalu *pooling layer* lain, dan seterusnya. Gambar semakin kecil dan semakin kecil seiring banyaknya jaringan, dan juga akan semakin dalam dan dalam dengan kata lain lebih banyak layer (dengan lebih banyak peta fitur) berkat lapisan konvolusional. Di bagian atas CNN, terdapat *feed forward neural network regular* ditambahkan, terdiri dari beberapa layer fully connected (+ ReLUs),

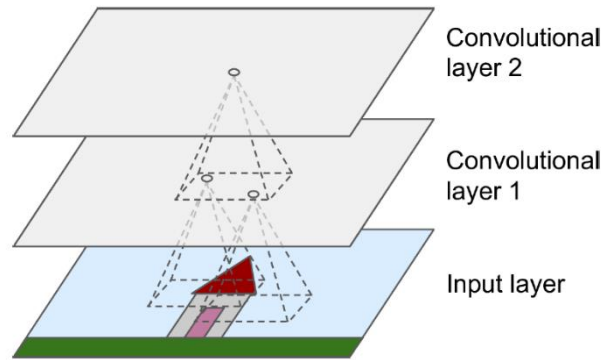
dan lapisan akhir menampilkan prediksi (lapisan softmax yang menghasilkan perkiraan probabilitas kelas) (Aurélien, 2017).



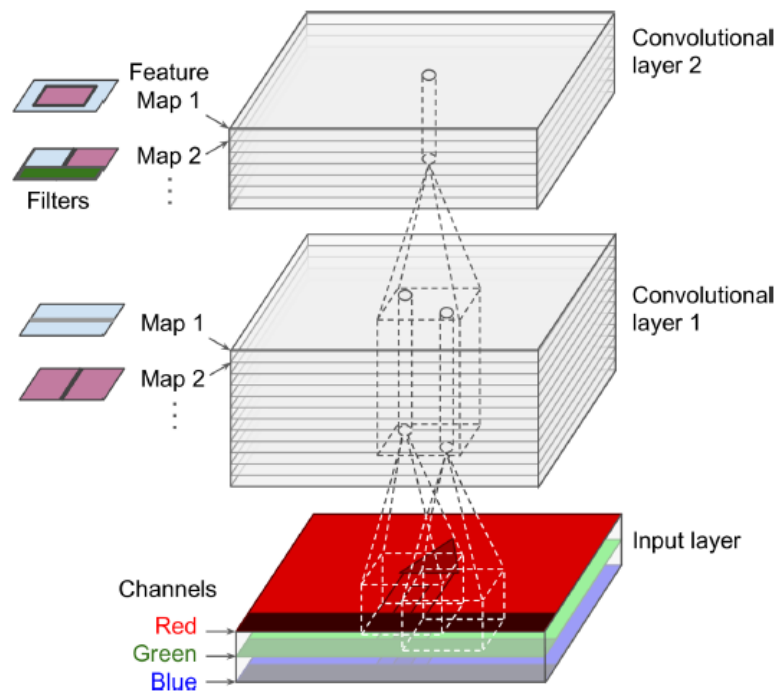
Gambar 2.9 Arsitektur CNN  
(Aurélien, 2017)

#### 2.4.1 Convolution Layer

Salah satu hal yang paling penting dari Arsitektur CNN adalah lapisan *convolution layer*. *Convolution* adalah operasi matematika yang menggeser satu fungsi ke fungsi lainnya dan mengukur integral dari perkalian titik-titiknya. Lapisan konvolusional sebenarnya menggunakan korelasi silang, yang sangat mirip dengan konvolusi. Pada CNN, neuron di lapisan konvolusional pertama tidak terhubung ke setiap piksel tunggal atau *full connected*, tetapi hanya piksel di bidang. Setiap neuron dalam lapisan konvolusional kedua terhubung hanya ke neuron yang terletak di dalam persegi panjang kecil di lapisan pertama. Arsitektur ini memungkinkan jaringan untuk lebih fokus pada fitur tingkat rendah di *hidden layer* pertama, lalu membuatnya menjadi fitur tingkat tinggi di *hidden layer* berikutnya, dan seterusnya. Struktur hierarkis ini umum digunakan dalam gambar di dunia nyata, yang merupakan salah satu alasan mengapa CNN bekerja dengan baik untuk pengenalan gambar (Aurélien, 2017).



Gambar 2.10 Convolutional layer  
(Aurélien, 2017)



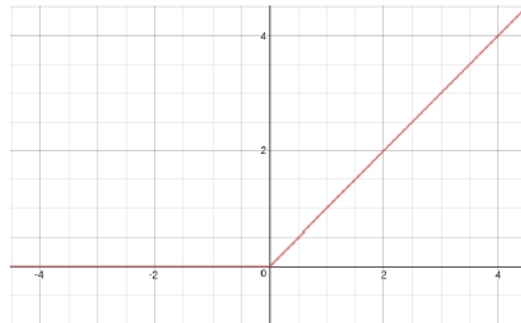
Gambar 2.11 Convolutional layer dengan *feature map*  
(Aurélien, 2017)

#### 2.4.2 Rectified Linear Units (ReLU) Activation

ReLU adalah fungsi aktivasi yang diperkenalkan oleh Richard HR Hahnloser, yang memiliki dasar biologis dan matematika yang kuat. Pada 2011, hal itu ditunjukkan untuk lebih meningkatkan pelatihan jaringan saraf yang dalam. Ia bekerja dengan menetapkan nilai pada 0, yaitu  $f(x) = \max(0, x)$ .

Sederhananya, output 0 ketika  $x < 0$ , dan sebaliknya, output fungsi linear ketika  $x \geq 0$  (lihat Gambar 2.12 untuk representasi visual). ReLU secara konvensional digunakan sebagai fungsi aktivasi untuk *neural network*, dengan softmax menjadi fungsi klasifikasinya di akhir *node*. Kemudian, jaringan tersebut menggunakan fungsi *cross-entropy* softmax untuk mempelajari parameter berat  $\theta$  dari jaringan saraf (Agarap, 2018). ReLU di gambarkan dengan rumus dan grafik di bawah:

$$R(z) = \max(0, z)$$



Gambar 2.12 ReLU  
(Agarap, 2018)

Keterangan: fungsi aktivasi menghasilkan 0 sebagai output ketika  $x < 0$ , dan kemudian menghasilkan linier dengan kemiringan 1 ketika  $x > 0$ .

#### 2.4.3 Cross Entropy Loss Function

Loss Function atau Cost Function merupakan fungsi dari *neural network* yang menggambarkan kerugian yang terkait dengan semua kemungkinan yang dihasilkan oleh model. Loss Function bekerja ketika model pembelajaran memberikan kesalahan yang harus diperhatikan. Loss Function yang baik adalah fungsi yang menghasilkan error yang paling rendah. Ketika suatu model memiliki kelas yang cukup banyak, perlu adanya cara untuk mengukur perbedaan antara probabilitas hasil hipotesis dan probabilitas kebenaran yang asli, dan selama pelatihan banyak

algoritma yang dapat menyesuaikan parameter sehingga perbedaan ini diminimalkan. *Cross entropy* adalah pilihan yang masuk akal. Gambaran umum algoritma ini adalah meminimalkan kemungkinan log negatif dari dataset, yang merupakan ukuran langsung dari performa prediksi model (Aurélien, 2017).

$$H(p, q) = -\sum_x p(x) \log q(x)$$

Keterangan:

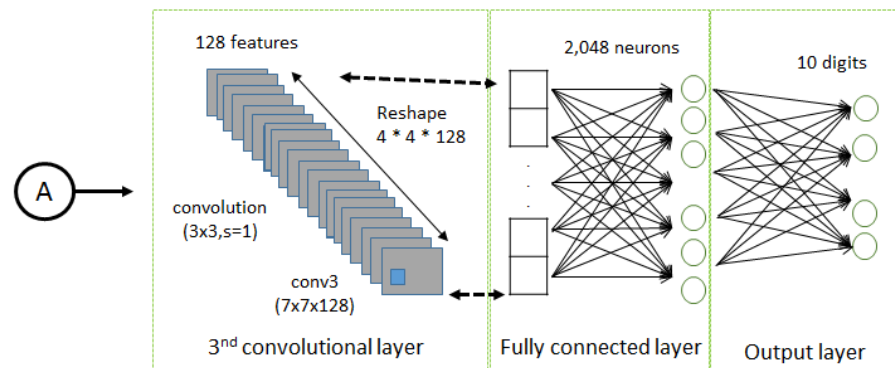
H = loss function

p = Hasil akhir

q = hasil prediksi

#### 2.4.4 Fully Connected Layer

Proses ini bertujuan untuk melakukan transformasi pada dimensi data agar data dapat diklasifikasikan secara linear.



Gambar 2.13 Contoh fully connected layer pada MNIST (Uniqtech, 2018)

Gambar di atas merupakan proses *converting* hasil dari fitur map max-pooling menjadi *flatten* atau vector. Dalam proses ini nilai input matriks dari layer sebelumnya akan diubah menjadi vector. Proses ini sama dengan Proses MLP (*Multilayer Perceptron*). Jaringan ini umumnya menggunakan lapisan yang terhubung sepenuhnya di mana setiap piksel dianggap sebagai neuron terpisah. Dalam proses ini biasanya diterapkan metode “dropout”. Metode ini bertujuan untuk

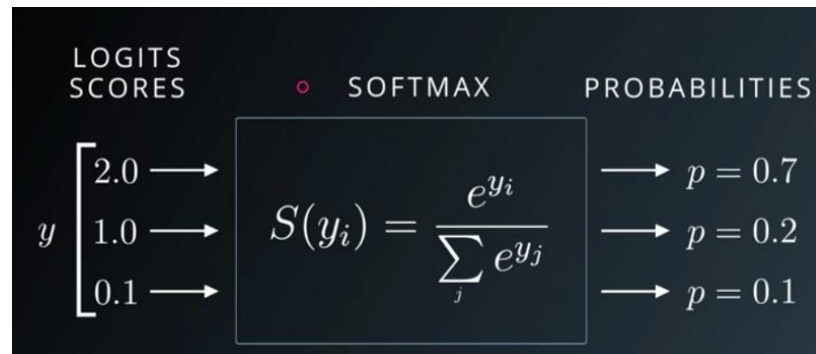


menonaktifkan beberapa edge yang terhubung ke setiap neuron untuk menghindari overfitting. Setelah itu proses terakhir adalah klasifikasi. Dalam proses ini digunakan aktivasi fungsi softmax.

#### 2.4.5 Fungsi aktivasi Softmax

Softmax adalah fungsi aktivasi yang mengubah angka alias log menjadi probabilitas yang berjumlah satu. Ini juga merupakan elemen inti yang digunakan dalam tugas klasifikasi di *Deep Learning*. Softmax adalah fungsi aktivasi yang merupakan bagian dari ReLU dan Sigmoid. Ini sering digunakan dalam klasifikasi. Output softmax besar jika skor (input disebut logit) besar. Outputnya kecil jika nilainya kecil. Proporsi tidak seragam. Softmax bersifat eksponensial, dapat memperbesar perbedaan - mendorong satu hasil lebih dekat ke 1 sementara yang lain lebih dekat ke 0. Ternyata skor alias log menjadi probabilitas. Cross entropy (fungsi biaya) sering dihitung untuk output Softmax label softmax dan label sebenarnya dengan (One Hot Encoding). (Uniqtech, 2018)

Softmax juga memberikan hasil yang lebih intuitif dan juga memiliki interpretasi probabilistik yang lebih baik dibanding algoritma klasifikasi lainnya. Softmax memungkinkan kita untuk menghitung probabilitas untuk semua label. Dari label yang ada akan diambil sebuah vektor nilai bernilai riil dan merubahnya menjadi vektor dengan nilai antara nol dan satu yang bila semua dijumlah akan bernilai satu.



Gambar 2.14 Softmax Activation  
(Uniqtech, 2018)

Keterangan:

S: Probabilitas

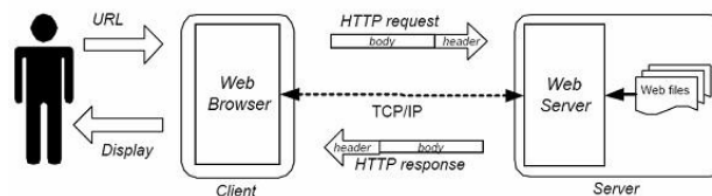
$e^{y_i}$ : Bilangan Euler atas vektor logit

$\sum_j e^{y_j}$ : Jumlah bilangan Euler atas semua vektor logit

## 2.5 Website

Situs web (*website*) adalah suatu halaman web yang saling berhubungan yang umumnya berada pada peladen yang sama berisikan kumpulan informasi yang disediakan secara perorangan, kelompok, atau organisasi. Sebuah situs web biasanya ditempatkan setidaknya pada sebuah server web yang dapat diakses melalui jaringan seperti Internet, ataupun jaringan wilayah lokal (LAN) melalui alamat Internet yang dikenali sebagai URL. Gabungan atas semua situs yang dapat diakses publik di Internet disebut pula sebagai World Wide Web atau lebih dikenal dengan singkatan WWW. Meskipun setidaknya halaman beranda situs Internet umumnya dapat diakses publik secara bebas, pada prakteknya tidak semua situs memberikan kebebasan bagi publik untuk mengaksesnya, beberapa situs web mewajibkan pengunjung untuk melakukan pendaftaran sebagai anggota, atau bahkan meminta pembayaran untuk dapat menjadi anggota untuk dapat mengakses isi yang terdapat dalam situs web tersebut, misalnya situs-situs yang menampilkan pornografi, situs-situs berita, layanan surel (e-mail), dan lain-lain. Pembatasan-pembatasan ini umumnya dilakukan karena alasan keamanan, menghormati privasi, atau karena tujuan komersil tertentu.

Dalam konsep kerjanya, user/pengguna yang akan mengakses suatu website berupa url melalui browser (yaitu media untuk menuju url yang diakses), kemudian browser tersebut mengirimkan permintaan/ request berupa http request kepada server melalui layer-layer TCP/IP, kemudian server memberikan files yang direquest jika ada. Files yang telah diberikan tadi tidak langsung ditampilkan/didisplay begitu saja, namun server memberikan respon kembali ke browser melalui http response yang juga melalui layer-layer TCP/IP,36 yang kemudian baru di terima oleh browser, dan kemudian dikirimkan kepada user berupa display (IdHost, 2018).



Gambar 2.15 Ilustrasi kerja website  
(IdHost, 2018)

### 2.5.1 HTML

HTML adalah singkatan dari *Hyper Text Markup Language*, yaitu bahasa (aturan) standar yang digunakan untuk menampilkan teks, gambar, video dan audio ke dalam halaman web. HTML merupakan *file* teks yang tersusun atas elemen-elemen yang disebut dengan tag. Tag HTML diapit dengan tanda lebih kecil (<) dan tanda lebih besar (>), misalnya <html>, <head>, <body>, <p>, dan lain-lain. Tag HTML ada yang memiliki pasangan, ada juga yang tidak. Jika suatu tag memiliki pasangan, maka tag penutup akan disertai dengan tanda slash (/), misalnya : </html>, </head>, </body>, </p>, dan lain-lain. Dokumen atau *file* HTML dapat dibuat dengan menggunakan aplikasi *Text Editor* apa saja, dan disimpan dengan ekstensi .html atau .htm (Budi, 2011a)

### 2.5.2 CSS

*Cascading Style Sheet* (CSS) adalah suatu bahasa yang bekerja sama dengan dokumen HTML untuk mendefinisikan cara bagaimana suatu isi halaman web ditampilkan atau dipresentasikan. Presentasi ini meliputi

*style* atau gaya teks, *link*, maupun tata letak (*layout*) halaman. Dengan adanya teknologi seperti ini, kita dapat memilah atau memisahkan antara kode untuk isi halaman web dan kode yang diperlukan khusus untuk menangani tampilan

Kode CSS tersusun atau *selector* dan deklarasi. *Selector* adalah tag HTML yang akan diberi atau dikenai CSS, sedangkan deklarasi adalah *property* dan nilai yang akan ditentukan untuk tag bersangkutan. Sebagai contoh perhatikan kode CSS berikut: (Budi, 2011a)

```
body {
    background:
    black;
}
```

### 2.5.3 Javascript

Javascript adalah bahasa yang berfungsi untuk membuat skrip-skrip program yang dapat dikenal dan dieksekusi oleh web browser dengan tujuan untuk menjadikan halaman web lebih bersifat interaktif. Meskipun banyak fitur dari bahasa Java yang diadopsi oleh Javascript, namun Javascript dikembangkan secara terpisah dan independen. Jadi perlu Anda catat bahwa Javascript itu bukan Java; Javascript dan Java merupakan dua hal yang konsepnya sangat berbeda meskipun ada kemiripan dalam penelitian sintaksnya. Javascript dikembangkan oleh Netscape dan merupakan bahasa yang bersifat terbuka (open) sehingga setiap orang dapat menggunakannya tanpa harus membeli lisensi (Budi, 2011)

Beberapa contoh penggunaan javascript yang sering dijumpai dalam halaman web adalah:

1. Menampilkan pesan peringatan atau konfirmasi ke user
2. Menampilkan popop window
3. Membuat menu dropdown

4. Melakukan validasi pada saat user memasukkan dalam dalam suatu form.
5. Menampilkan tanggal dan waktu

Namun dalam penelitian ini selain menjadi DOM, Javascript juga di gunakan sebagai bahasa pemrograman machine learning untuk pengklasifikasian gambar.

## 2.6 Node.js

Node.js adalah perangkat lunak yang didesain untuk mengembangkan aplikasi berbasis web dan ditulis dalam sintaks bahasa pemrograman JavaScript. Bila selama ini kita mengenal JavaScript sebagai bahasa pemrograman yang berjalan di sisi client / browser saja, maka Node.js ada untuk melengkapi peran JavaScript sehingga bisa juga berlaku sebagai bahasa pemrograman yang berjalan di sisi server, seperti halnya PHP, Ruby, Perl, dan sebagainya. Node.js dapat berjalan di sistem operasi Windows, Mac OS X dan Linux tanpa perlu ada perubahan kode program. Node.js memiliki pustaka server HTTP sendiri sehingga memungkinkan untuk menjalankan server web tanpa menggunakan program server web seperti Apache atau Nginx.

Untuk mengeksekusi Javascript sebagai bahasa server diperlukan engine yang cepat dan mempunyai performansi yang bagus. Engine Javascript dari Google bernama V8-lah yang dipakai oleh Node.js yang juga merupakan engine yang dipakai oleh browser Google Chrome. Dalam buku yang di tulis oleh Krishna Rungta Node.js adalah program open-source yang digunakan untuk pengembangan aplikasi web sisi-server. Aplikasi Node.js ditulis dalam JavaScript dan dapat dijalankan pada berbagai sistem operasi. Node.js didasarkan pada arsitektur yang digerakkan oleh *event* dan API I/O non-blocking yang dirancang untuk mengoptimalkan throughput dan skalabilitas aplikasi untuk aplikasi web real-time. Nodejs sangat cocok dengan penelitian ini karena nodejs juga berdasarkan menjalankan servernya melalui asynchronous function. Dalam penelitian ini di butuhkan module nodejs yaitu Express dan NPM untuk menjalankan web server (Rungta, 2016).

### 2.6.1 npm

npm adalah sistem manajemen paket dan distribusi untuk Node.js. Ini telah menjadi standar *de facto* untuk mendistribusikan modul (paket) untuk digunakan dengan Node.js. Secara konseptual, ini mirip dengan alat seperti apt-get (Debian), rpm / yum (Red Hat / Fedora), MacPorts (macOS), CPAN (Perl), atau PEAR (PHP). Tujuannya adalah menerbitkan dan mendistribusikan paket Node.js melalui Internet menggunakan antarmuka baris perintah sederhana. Dengan npm, Anda dapat dengan cepat menemukan paket untuk melayani tujuan tertentu, mengunduhnya, menginstalnya, dan mengelola paket yang sudah Anda instal. Aplikasi npm meluas pada format paket untuk Node.js, yang pada gilirannya sebagian besar didasarkan pada spesifikasi paket CommonJS. Ini menggunakan file `package.json` yang sama yang didukung secara asli oleh Node.js, tetapi dengan bidang tambahan untuk membangun fungsionalitas tambahan (Herron, 2018). npm adalah singkatan dari Node Package Manager yang mengatur segala modul yang akan atau sudah di install di dalam projek node. NPM merupakan projek Open-Source yang memiliki lebih dari 11.000.000 module dan tools yang bisa di gunakan di dalam node. Sehingga memudahkan node developer untuk membuat website.

### 2.6.2 Express

Express.js adalah Node.js web application server framework, yang dirancang khusus untuk membangun aplikasi web satu halaman, multi-halaman, dan hybrid. Ini telah menjadi kerangka kerja server standar untuk node.js. Express adalah bagian backend dari sesuatu yang dikenal sebagai tumpukan MEAN. MEAN adalah tumpukan aplikasi open-source JavaScript untuk membangun situs web dan aplikasi web dinamis. Express digambarkan seperti Sinatra, merujuk pada *framework* aplikasi Ruby yang populer, dan itu bukan *framework* yang dapat di pertanyakan, artinya pembuat *framework* tidak memaksakan

pendapat mereka tentang penataan aplikasi. Ini berarti Express sama sekali tidak ketat tentang bagaimana kode disusun, *developer* cukup menuliskannya dengan cara yang menurutnya terbaik (Herron, 2018).

## 2.7 Client-Side Machine Learning

Dari beberapa jurnal, buku mengenai *machine learning*, dan beberapa upaya dari para peneliti untuk memindahkan *machine learning* ke *client-side* / *serverless*, peneliti menyimpulkan untuk menggunakan *client-side machine learning* adalah sebuah upaya memindahkan *machine learning* dan *deep learning* kedalam sebuah entitas yang tidak memerlukan *server* sebagai daya komputasinya, sehingga dapat dilakukan dengan biaya seminimal mungkin dan menjaga data pribadi pengguna pada alat komputasinya masing-masing. *Machine learning* dan *deep learning* membutuhkan daya komputasi yang signifikan ketika menerapkan model neural network, bahasa seperti *Javascript* tidak dibangun untuk itu. Tetapi kolaborasi antara *Javascript* dan *machine learning* membuktikan hal ini bisa dilakukan. *Javascript* dan *machine learning* dapat bekerja cukup baik bersama, untuk mengembangkan kemampuan browser web yang lebih menarik dan canggih (Ma et al., 2019). Manfaat yang jelas dari melakukan ML di *web (client-side)* adalah tidak memerlukan instalasi atau pengaturan rumit untuk menggunakan aplikasi tersebut dan mudah diakses di banyak *platform*. Aplikasi *web* juga menjadi lebih canggih. Karena itu, penting bagi pengembangan pembelajaran paralel dengan kemajuan dalam teknologi aplikasi *web*. Ada jutaan pengembang web yang mungkin memiliki kegunaan menarik untuk aplikasi *machine learning*, tetapi mungkin tidak memiliki pengetahuan untuk menggabungkan kedua bidang tersebut. Di penelitian kali ini peneliti menggabungkan *machine learning* dengan *website* dengan mengembangkan aplikasi web untuk melihat harga dari sebuah produk. User dari aplikasi web ini dapat menentukan sendiri produk-produk yang akan dia klasifikasi dan latih di *client-side machine learning* tersebut dengan tampilan yang mudah di mengerti dan tanpa membutuhkan pengkodean.

### 2.7.1 WebGL

Dengan kemajuan *deep learning* dan komputasi ilmiah secara umum, dan kemajuan dalam arsitektur GPU modern, penggunaan awal komputasi secara umum pada GPU (GPGPU) telah berkembang sangat pesat. Sementara mesin virtual JS modern dapat mengoptimalkan JS biasa secara luas, kinerjanya jauh di bawah daya komputasi yang disediakan GPU. Untuk menggunakan GPU, TensorFlow.js menggunakan WebGL, standar web lintas platform yang menyediakan API grafik 3D level rendah. Tidak seperti OpenCL dan CUDA, API WebGL didasarkan pada spesifikasi OpenGL ES yang tidak memiliki dukungan eksplisit untuk GPGPU.

WebGL biasanya di gunakan untuk keperluan grafis pada web browser dan menggunakan kanvas pada penerapannya. Untuk mengatasi keterbatasan dan kompleksitas WebGL, tim Tensorflow.js menulis lapisan abstraksi yang disebut GPGPUContext yang mengeksekusi fragmen shaders WebGL yang mewakili perhitungan. Dalam program grafis, shader fragmen biasanya digunakan untuk menghasilkan warna piksel yang akan ditampilkan di layar. Fragmen shader berjalan untuk setiap piksel secara independen dan paralel; TensorFlow.js memanfaatkan paralelisasi ini untuk mempercepat perhitungan ML. (Smilkov et al., 2019)

### 2.7.2 Asynchronous Function

JavaScript berjalan dalam single thread, dibagi dengan tugas-tugas seperti *layouting* dan *event handling*. Ini berarti fungsi JS yang sudah berjalan lama dapat menyebabkan pelambatan pada suatu halaman atau penundaan untuk melakukan *event handling*. Untuk mengurangi masalah ini, pengguna JS biasanya mengandalkan callbacks function dan promises pada pengkodeannya, yang merupakan bagian dari Asynchronous Function. Dalam melakukan machine learning dalam JS di perlukan fungsi berikut untuk memproses perhitungan secara cepat. Fungsi async dapat berisi ekspresi menunggu yang menghentikan sementara eksekusi fungsi async dan menunggu resolusi dari Promise

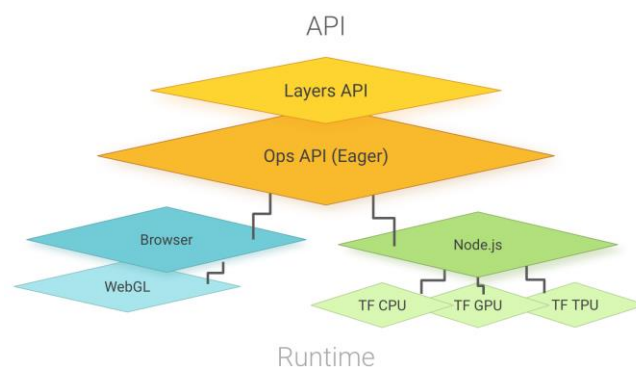


yang menunggu di penuhi, dan kemudian melanjutkan eksekusi fungsi async dan memberikan nilai yang dihasilkannya dan berlaku sebaliknya. Berikut merupakan contoh dari penggunaan Async pada javascript:

```
async function myFirstAsyncFunction() {
  try {
    const fulfilledValue = await promise;
  }
  catch (rejectedValue) {
    // ...
  }
}
```

### 2.7.3 Tensorflow.js

Dikutip dari websitenya Tensorflow.js adalah library untuk mengembangkan dan melatih model Machine Learning dalam JavaScript, dan menggunakannya di browser atau di Node.js. Tensorflow sendiri merupakan library yang ditulis dengan bahasa C++ dan biasanya digunakan dengan bahasa pemrograman Python. Dengan adanya tensorflow.js, kita sekarang sudah bisa menggunakan beberapa fitur tensorflow di sisi web browser tanpa harus dibebani oleh instalasi dependency yang biasanya di butuhkan dalam menjalankan machine learning dan membuat sebuah model. Tensorflow sendiri terbagi menjadi Layer API dan Core API.



Gambar 2.16 Arsitektur Tensorflow.js  
(Smilkov et al., 2019)

- *Core API*

Core API merupakan API yang langsung berhubungan dengan *low-level code* dan membuat dengan sedetail mungkin. Seperti membuat tensor dan scalar secara manual, dan biasanya di gunakan ketika kita menginginkan untuk membuat model sendiri dengan kebutuhan spesial dan belum ada di dalam *library layer API*.

- *Layer API*

*Layer API* merupakan *library API* yang di buat menggunakan *Core Api* dalam tensorflow.js yang merupakan *high-level code*. *Layer API* di buat menjadi sederhana agar pemula dapat menggunakan machine learning dengan mudah. *Layer API* pada Tensorflow adalah KERAS dan biasanya di gunakan untuk hal-hal yang sudah biasa di lakukan di dalam machine learning seperti klasifikasi, regresi dan lain-lain.

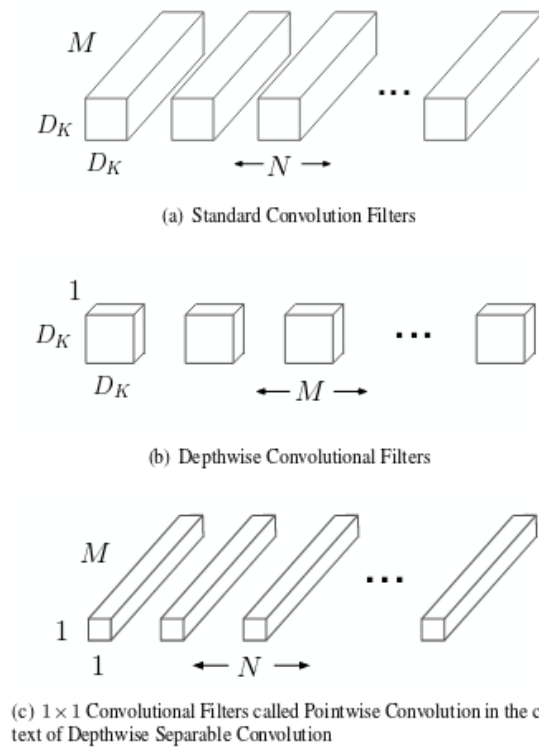
#### 2.7.4 ml5.js

ml5 adalah Layer API yang di kembangkan oleh pada dosen dan murid di New York University Interactive Telecommunications Program atau bisa di singkat NYU-ITP yang merupakan Layer API dari Tensorflow.js. ml5 dirancang bertujuan untuk membuat *machine learning* dapat didekati oleh banyak kalangan seperti seniman, coders kreatif, dan siswa. *Library* ml5 dikembangkan di New York University dan telah dirilis untuk umum pada Juli 2018. Library ml5 menyediakan akses ke algoritma ML, tugas dan model di *browser*, dibangun di atas TensorFlow.js tanpa ketergantungan eksternal lainnya. ML5 dapat dibandingkan dengan Keras. ml5.js dibangun di atas TensorFlow.js dan menggunakan fungsi TensorFlow.js di backend dengan membuat pengkodean lebih mudah bagi orang-orang yang baru mengenal arena Machine Learning. ml5.js sangat terinspirasi oleh sintaksis, pola dan gaya library p5.js. Namun, ada beberapa perbedaan dalam cara operasi async yang dimiliki oleh ml5.js. ml5.js mendukung error-first callbacks dan Promises di setiap metode (NYU, 2018). Pada penelitian ini peneliti menggunakan beberapa fungsi dari ml5 antara lain

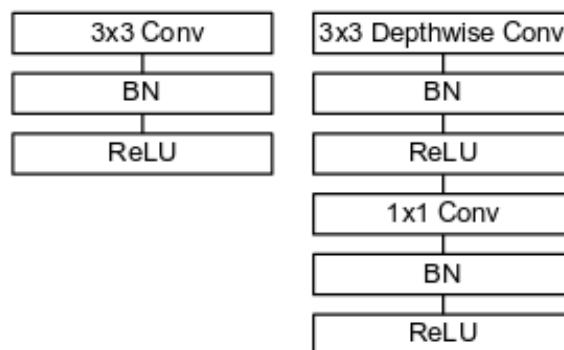
- `.addImage()`  
Berfungsi untuk menambahkan gambar + label ke dalam dataset, serta melakukan resize gambar untuk proses selanjutnya.
- `featureExtractor`  
Berfungsi untuk melakukan fungsi transfer learning serta melakukan import beban dari pretrained model
- `.train()`  
Berfungsi untuk melakukan fungsi training pada model klasifikasi.
- `.predict()`  
Berfungsi untuk melakukan prediksi pada model klasifikasi
- `.save()`  
Berfungsi untuk menyimpan model klasifikasi yang sudah di train.
- `.load()`  
Berfungsi untuk memuat model klasifikasi yang sudah di train.

#### 2.7.5 MobileNet Model

MobileNets, merupakan salah satu arsitektur *convolutional neural network* (CNN) yang dapat digunakan untuk mengatasi kebutuhan akan computing resource berlebih. Seperti namanya, Mobile, para peneliti dari Google membuat arsitektur CNN yang dapat digunakan untuk kebutuhan mobile. Perbedaan mendasar antara arsitektur MobileNet dan arsitektur CNN pada umumnya adalah penggunaan lapisan atau layer konvolusi dengan ketebalan filter yang sesuai dengan ketebalan dari input image. MobileNet membagi konvolusi menjadi *depthwise convolution* dan *pointwise convolution* seperti pada gambar berikut (Howard et al., 2017) .



Gambar 2.17 Konvolusi standard (a) dibagi menjadi dua lapisan: depthwise convolution (b) dan pointwise convolution (c) untuk membuat filter terpisah secara mendalam (depthwise) (Howard et al., 2017)



Gambar 2.18 Kiri: lapisan konvolusi standard dengan batchnorm dan ReLU. Kanan: Depthwise convolution dan Pointwise convolution dengan batchnorm dan ReLU. (Howard et al., 2017)

Table 1. MobileNet Body Architecture

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
5×	Conv dw / s1	$3 \times 3 \times 512$ dw
	Conv / s1	$1 \times 1 \times 512 \times 512$
	Conv dw / s2	$3 \times 3 \times 512$ dw
	Conv / s1	$1 \times 1 \times 512 \times 1024$
	Conv dw / s2	$3 \times 3 \times 1024$ dw
	Conv / s1	$1 \times 1 \times 1024 \times 1024$
	Avg Pool / s1	Pool $7 \times 7$
	FC / s1	$1024 \times 1000$
	Softmax / s1	Classifier

Gambar 2.18 Arsitektur MobileNet  
(Howard et al., 2017)

### 2.7.6 Transfer Learning

Pada awalnya, algoritma ML dirancang untuk menangani tugas atau masalah secara terpisah. Bergantung pada persyaratan kasus penggunaan dan data yang ada, suatu algoritma diterapkan untuk melatih model untuk tugas tertentu yang diberikan. *Transfer learning* membawa proses belajar selangkah lebih maju dan lebih sejalan dengan cara manusia memanfaatkan pengetahuan di seluruh tugas. Dengan demikian, transfer learning adalah metode penggunaan kembali model atau pengetahuan untuk tugas terkait lainnya. *Transfer learning* juga dianggap sebagai perpanjangan dari algoritma ML yang ada. Penelitian dan pekerjaan ekstensif sedang dilakukan dalam konteks pembelajaran transfer dan pada pemahaman bagaimana pengetahuan dapat ditransfer

antar tugas (Dipanjan, Bali, & Ghosh, 2018). Sejak itu, istilah-istilah seperti *learning to learn*, Konsolidasi Pengetahuan, dan Transfer Induktif telah digunakan secara bergantian dengan pembelajaran transfer. Para peneliti dan teks akademik yang berbeda memberikan definisi dari konteks yang berbeda. Dalam buku, *Deep Learning*, (Goodfellow et al., 2016). *Transfer learning* dalam konteks general. Definisi mereka adalah situasi di mana apa yang telah dipelajari dalam satu pengaturan dieksploitasi untuk meningkatkan generalisasi di pengaturan lain.

*Deep learning* telah berhasil digunakan untuk berbagai tugas *computer vision*, seperti *identification* dan *object recognition*, menggunakan arsitektur CNN yang berbeda. Dalam makalahnya, *How transferable are features in deep neural networks*, Yosinski dan rekan penelitiannya (Yosinski, Clune, Bengio, & Lipson, 2014) mempresentasikan temuan mereka tentang bagaimana lapisan bawah bertindak sebagai ekstraktor fitur visi komputer konvensional, seperti sebagai pendeteksi tepi, sedangkan lapisan terakhir bekerja ke arah fitur spesifik tugas.

## How transferable are features?

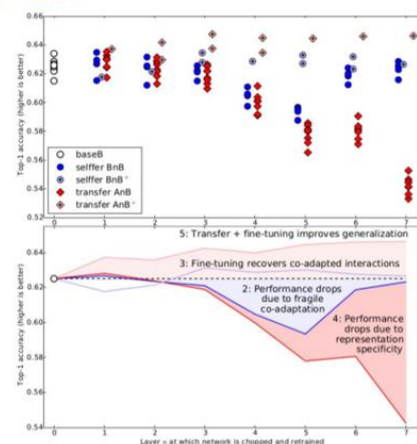
Transferability is negatively affected by two distinct issues:

- The specialization of higher layer neurons
- Optimization difficulties related to splitting networks between co-adapted neurons

Fine-tuning improves generalization when sufficient examples are available.

Transfer learning and fine tuning often lead to better performance than training from scratch on the target dataset.

Even features transferred from distant tasks are often better than random initial weights!



Gambar 2.19 Hasil penelitian Yosinski mengenai *transfer learning* (Yosinski et al., 2014)

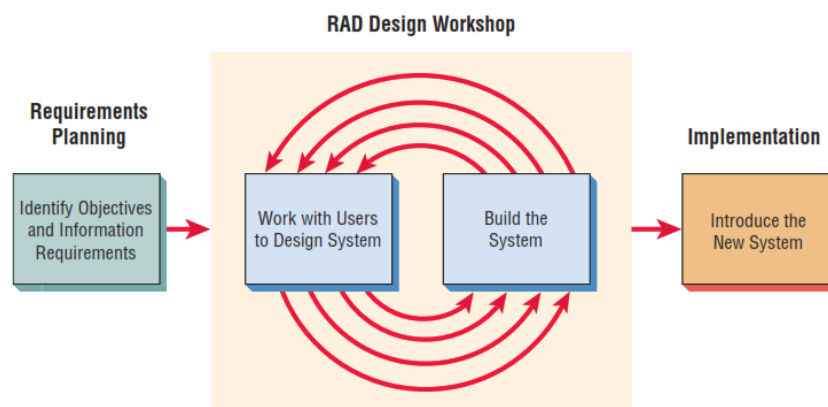
Dapat di simpulkan transfer lerning dapat digunakan untuk meningkatkan akurasi dan kinerja dari model *deep learning* yang sedang di buat dan mempercepat pembuatan model itu sendiri tanpa harus membuatnya dari awal. Dalam penelitian ini *transfer learning* sangat di butuhkan untuk mempersingkat waktu dan menekan ukuran dari aplikasi *deep learning* yang di buat untuk menerapkan *image classification*.

## 2.8 Rapid Application Development (RAD)

Metode pengembangan yang di gunakan pada penelitian ini adalah Rapid Application Development (RAD). Rapid Application Development (RAD) merupakan metode pengembangan sistem informasi dengan waktu singkat, sehingga dinilai tepat digunakan dalam penelitian ini. RAD menggunakan metode iteratif (berulang) dalam mengembangkan sistem dimana model yang bekerja dalam sistem dikonstruksikan di awal tahap pengembangan dengan tujuan menetapkan kebutuhan (requirement) pengguna dan selanjutnya disingkirkan. Working model digunakan kadang-kadang saja sebagai basis desain dan implementasi sistem final. Aspek terpenting bagi model ini untuk berhasil adalah dengan meyakinkan bahwa prototype yang dikembangkan akan dapat digunakan kembali. (Carol Britton, 2001) Menurut Kendall RAD adalah pendekatan berorientasi objek untuk pengembangan sistem yang mencakup metode pengembangan serta perangkat lunak (Kendall & Kendall, 2010). Pressman juga mengatakan RAD adalah proses model proses perangkat lunak incremental yang memiliki siklus pengembangan yang singkat. Model RAD adalah adaptasi dari model Waterfall, dimana perkembangan pesat dapat dicapai dengan menggunakan pendekatan konstruksi berbasis komponen. Jika kebutuhan dan batas ruang lingkup sudah diketahui, proses RAD memungkinkan tim pengembang untuk menciptakan system yang berfungsi penuh dalam jangka waktu yang sangat singkat (Pressman & Maxim, 2015).

Profesor Clifford Kettemborough dari College Whitehead, University of Redlands, mendefinisikan Rapid Application Development sebagai “pendekatan untuk membangun sistem komputer yang

menggabungkan Computer Assisted Software Engineering (CASE) tools dan teknik, user-driven prototyping. RAD meningkatkan kualitas sistem secara drastis dan mengurangi waktu yang diperlukan untuk membangun sistem. Dari penjelasan berikut dapat diketahui implementasi metode RAD dapat dilakukan secara maksimal jika pengembang aplikasi sudah merumuskan kebutuhan dan ruang lingkup aplikasi dengan baik, metode ini juga efektif jika digunakan dalam sebuah aplikasi yang memiliki skala kecil, sehingga tidak membutuhkan banyak sumber daya dan dapat dilakukan dengan cepat.



Gambar 2.20 Siklus RAD  
(Kendall & Kendall, 2010)

### 2.8.1 Tahapan Pengembangan Sistem

Menurut buku “*Systems Analysis and Design, 8th Edition*” yang di tulis Kendall Ada tiga fase luas untuk RAD yang melibatkan pengguna dan analis dalam penilaian, desain, dan implementasi. Gambar 2.21 menggambarkan ketiga fase ini. RAD melibatkan pengguna di setiap bagian dari upaya pengembangan, dengan partisipasi intens dalam bagian bisnis desain. Ketiga tahapan tersebut adalah *requirement planning* (perencanaan kebutuhan), *RAD design workshop* (*workshop* desain RAD), dan *implementation* (implementasi). Tahapan metode pada RAD menurut Kendall adalah sebagai berikut:



a. Requirement Planning

Dalam fase perencanaan persyaratan, pengguna dan analis bertemu untuk mengidentifikasi:

- Tujuan-tujuan aplikasi di bentuk.
- Syatar-syarat informasi yang di timbulkan dari tujuan-tujuan tersebut.

Orientasi dalam fase ini adalah penyelesaian masalah bisnis. Meskipun teknologi dan sistem informasi dapat mendorong beberapa solusi yang diusulkan, fokusnya akan selalu tetap pada pencapaian tujuan bisnis.

b. RAD Design Workshop

Pada tahap ini dilakukan design dan penyempurnaan yang paling baik dicirikan sebagai *workshop*.

- Merancang desain proses dan desain pemrograman untuk data-data yang telah didapatkan.
- Membangun dan mendemonstrasikan visual desain dan pola kerja pada pengguna.
- Memperbaiki segala sesuatu yang perlu di perbaiki

Selama tahapan ini pengguna merespon prototype yang ada dan penganalisis memperbaiki modul-modul yang di rancang berdasarkan respon pengguna. Namun apabila pengembang ataupun penggunanya merupakan orang yang berpengalaman, Kendall menilai usaha kreatif ini dapat mendorong pengembangan pada tingkat terakselerasi (Kendall & Kendall, 2010).

c. Implementation

Segera setelah aspek-aspek tahap sebelumnya disepakati dan sistem dibangun dan disempurnakan, sistem baru atau bagian dari sistem diuji dan kemudian diperkenalkan ke organisasi (Kendall & Kendall, 2010).

## 2.9 Metode Pengujian Sistem

Pengujian adalah sebuah proses terhadap aplikasi/program untuk menemukan segala kesalahan dan segala kemungkinan yang akan menimbulkan kesalahan sesuai dengan spesifikasi perangkat lunak yang telah ditentukan sebelum aplikasi tersebut diserahkan kepada pelanggan (Khalilulah, 2016).

Sementara itu *testing* adalah sebuah proses yang diejawantahkan sebagai siklus hidup dan merupakan bagian dari proses rekayasa perangkat lunak secara terintegrasi demi memastikan kualitas dari perangkat lunak serta memenuhi kebutuhan teknis yang telah disepakati dari awal.

Dari kedua pendapat diatas dapat disimpulkan bahwa pengujian sistem (*testing*) adalah salah satu proses penting dalam pengembangan sebuah perangkat lunak untuk mencari kesalahan yang ada dalam sistem dan melakukan pengecekan kualitas dari sebuah perangkat lunak.

### 2.9.1 Blackbox Testing

*Black box testing* adalah tipe testing yang memperlakukan perangkat lunak yang tidak diketahui kinerja internalnya. Sehingga para tester memandang perangkat lunak seperti layaknya ”kotak hitam” yang tidak penting dilihat isinya, tapi cukup dikenai proses testing di bagian luar (Khalilulah, 2016).

Metode ujicoba *blackbox* memfokuskan pada keperluan fungsional dari *software*. Karna itu uji coba *blackbox* memungkinkan pengembang *software* untuk membuat himpunan kondisi input yang akan melatih seluruh syarat-syarat fungsional suatu program. Uji coba *blackbox* bukan merupakan alternatif dari uji coba *whitebox*, tetapi merupakan pendekatan yang melengkapi untuk menemukan kesalahan lainnya, selain menggunakan metode *whitebox*.

### 2.9.2 Keuntungan Black Box Testing

Keuntungan dari *black box testing* antara lain :

- Anggota tim tester tidak harus dari seseorang yang memiliki kemampuan teknis di bidang pemograman.
- Kesalahan dari perangkat lunak ataupun bug seringkali ditemukan oleh komponen tester yang berasal dari pengguna.

- Hasil dari *black box testing* dapat memperjelas kontradiksi ataupun kerancuan yang mungkin timbul dari eksekusi sebuah perangkat lunak.
- Proses testing dapat dilakukan lebih cepat dibandingkan *white box testing*.

## 2.10 Tinjauan Pustaka

Tinjauan pustaka di ambil dari beberapa skripsi dan journal yang berhubungan dengan penelitian. Setiap tinjauan akan di bahas satu persatu secara singkat tentang judul, tools, dan hasil dari setiap penelitian.

- Pada penelitian yang berjudul “*CNNs for NLP in the Browser: Client-Side Deployment and Visualization Opportunities*” tahun 2018, yang di tulis oleh Yiyun Liang, Zhuncheng Tu, Laetitia Huang, dan Jimmy Lin bertujuan untuk menguji penggunaan *Client-side machine learning* sepenuhnya dalam membuat model CNN untuk NLP di berbagai *device* yang menggunakan *browser* Chrome seperti Macbook, Desktop, dan Smartphone. Penelitian ini menyimpulkan bahwa penggunaan Client-Side machine learning memungkinkan tetapi terbilang lambat untuk melakukan komputasi dengan data yang besar.
- Pada penelitian yang berjudul “Analisis Performa dan Pengembangan Sistem Deteksi Ras Anjing pada Gambar dengan Menggunakan Pre-Trained CNN Model” tahun 2018, yang di tulis oleh Muftah Afrizal Pangestu dan Hendra Bunyamin bertujuan untuk menguji akurasi dari 3 Pre-Trained Model yang memiliki arsitektur CNN antara lain Xception, RasNet5, dan VGG16. Dari penelitian tersebut Xception memiliki akurasi terbaik dengan nilai 98% untuk mengenali anjing atau tidak dan 67% untuk mengklasifikasikan rasnya.
- Pada penelitian yang berjudul “*MobileNet Convolutional Neural Networks and Support Vector Machines for Palmprint Recognition*” tahun 2019, yang di tulis oleh Aurelia Michel, Vincent Colin, dan Diaz D. Santika bertujuan untuk melakukan klasifikasi menggunakan MobileNet V2 dan SVM dalam mengklasifikasikan telapak tangan. Hasil dari penelitian ini adalah nilai akurasi dari MobileNet V2 yang mencapai 99.5% dan kombinasi MobileNet V2 dan SVM yang mencapai 100% akurasi dari training dan validasinya.

- Pada penelitian yang berjudul “Implementasi Convolutional Neural Network Terhadap Transportasi Tradisional Menggunakan Keras” tahun 2018, yang di tulis oleh Arfian bertujuan untuk melakukan klasifikasi menggunakan CNN dan mencari pengaturan yang tepat dari kombinasi *epoch*, *batch size*, dan *validation split*, dan di hasilkan bahwa *epoch* 30, *batch size* 32, dan *validation split* 0,2 mendapatkan akurasi terbaik dengan 82%
- Pada penelitan yang berjudul “Batik Classification Using Deep Convolutional Network Transfer Learning” tahun 2018, yang di tulis oleh Yohanes Gultom, Rian Josua Masikome, dan Aniasi Murni Arymurthy bertujuan untuk menguji *Transfer Learning* menggunakan VGG16 dengan SIFT dan SURF dalam mengekstraksi fitur dan akurasi. Penilitan ini membuktikan bahwa penggunaan *Transfer Learning* lebih efisien.

Tabel 2.1 Perbandingan Literatur

No.	Judul	CNN	Transfer Leaning	Client Side
1.	Implementasi Convolutional Neural Network Terhadap Transportasi Tradisional Menggunakan Keras (2018)	√	-	-
2.	Analisis Performa dan Pengembangan Sistem Deteksi Ras Anjing pada Gambar dengan Menggunakan Pre-Trained CNN Model (2018)	√	√	-
3.	MobileNet Convolutional Neural Networks and Support Vector Machines for Palmprint Recognition (2019)	√	√	-

4.	Batik Classification Using Deep Convolutional Network Transfer Learning (2018)	√	√	-
5.	CNNs for NLP in the Browser: Client-Side Deployment and Visualization Opportunities (2018)	√	-	√
6.	Penerapan <i>Image Classification</i> Dengan Pre-Trained Model MobileNet dalam <i>Client-Side Machine Learning</i> (2019)	√	√	√

Pada penelitian yang berjudul “Penerapan Image Classification Dengan Pre-Trained Model MobileNet dalam Client-Side Machine Learning” yang peneliti buat, bertujuan untuk menerapkan *client-side machine learning* menggunakan library ml5 dan Transfer Learning dengan Pre-Trained Model MobileNet untuk mengefisienkan dan meringankan sistem, karena client-side machine learning tidak bergantung dengan conver model maka peneliti akan membuat sistem untuk membuat model klasifikasi, menyimpan model, dan memprediksi model tersebut tanpa koding dan pengetahuan machine learning.

## **BAB 3**

### **METODOLOGI PENELITIAN**

Pembahasan pada bab 3 dilakukan pengumpulan data – data dan informasi sebagai bahan yang mendukung kebenaran materi uraian pembahasan. Selain itu untuk menyelesaikan masalah yang ada dalam sebuah perancangan sistem, maka diperlukan beberapa tahap yang harus dilakukan. Dalam bab ini dijelaskan mengenai bahan dan alat, serta metodologi penelitian yang digunakan dalam pengembangan sistem.

#### **3.1 Metode Pengumpulan Data**

Pengumpulan data dimaksudkan untuk mencari dan mengumpulkan data yang terkait dengan penelitian seperti dasar teori, metodologi penelitian, metodologi proses, dan acuan penelitian sejenis. Dalam penelitian ini, metode pengumpulan data yang dilakukan adalah studi pustaka, dan studi literatur.

##### **3.2.1 Studi Pustaka**

Peneliti menggunakan studi pustaka untuk mengumpulkan teori, fakta, kasus, dan masalah yang berkaitan dengan penelitian ini. Peneliti mencari referensi yang berhubungan dengan obyek penelitian. Peneliti melakukan studi pustaka di perpustakaan, toko buku, atau melalui internet secara online. Peneliti telah mengumpulkan 14 buku, 12 jurnal dan 7 situs web dari hasil studi pustaka yang digunakan di dalam penelitian.

##### **3.2.2 Studi Literatur**

Dalam melakukan penelitian, peneliti juga mencari studi literatur dari berbagai jurnal dan skripsi yang memiliki topik yang sejenis dengan penelitian yang sedang di lakukan

#### **3.2 Metode Pengembangan Sistem**

##### **3.2.1 Fase *Requirment Planning***

Dalam fase ini peneliti melakukan langkah-langkah sebagai berikut:

1. Menganalisis sistem yang berjalan.
2. Menganalisis hasil studi pustaka dan memahami masalah yang ada pada sistem yang berjalan.
3. Menganalisis sistem usulan
4. Mengidentifikasi fitur yang akan di terapkan pada sistem yang akan di buat untuk penelitian.

### 3.2.2 Fase *Workshop Design*

Pada fase ini terdapat 2 fase yang berhubungan satu sama lain yaitu *design system* dan *build system*. Peneliti melakukan proses desain dan pengkodean di fase ini serta melakukan perbaikan apabila terjadi ketidaksesuaian dan langsung di perbaiki di fase ini. Tahapan yang di lakukan pada fase ini, yaitu:

1. Perancangan model klasifikasi dengan *Pre-Trained Model* MobileNet menggunakan library ml5 dan fungsi-fungsinya.
2. Perancangan arsitektur pada sistem.
3. Perancangan proses pada sistem menggunakan UML yaitu dengan membuat 3 macam diagram, yaitu Use Case Diagram, Activity Diagram, dan Class Diagram. Dalam perancangan UML, peneliti menggunakan *software* Visio 2016.
4. Perancangan *user interface* sistem untuk memudahkan penggunaan sistem. Dalam perancangan *user interface* peneliti menggunakan Corel Draw.
5. Tahap penulisan kode dengan menggunakan HTML, CSS, JavaScript sebagai bahasa pemrograman, NodeJs dan Express sebagai *runtime environment* web app, kemudian melakukan *client-side machine learning* peneliti menggunakan ml5.js yang merupakan *library* dari Tensorflow.js, untuk memudahkan membuat model dalam melakukan klasifikasi pada gambar.

### 3.2.3 Fase *Implementation*

Pada fase ini peneliti melakukan beberapa tahapan, antara lain:

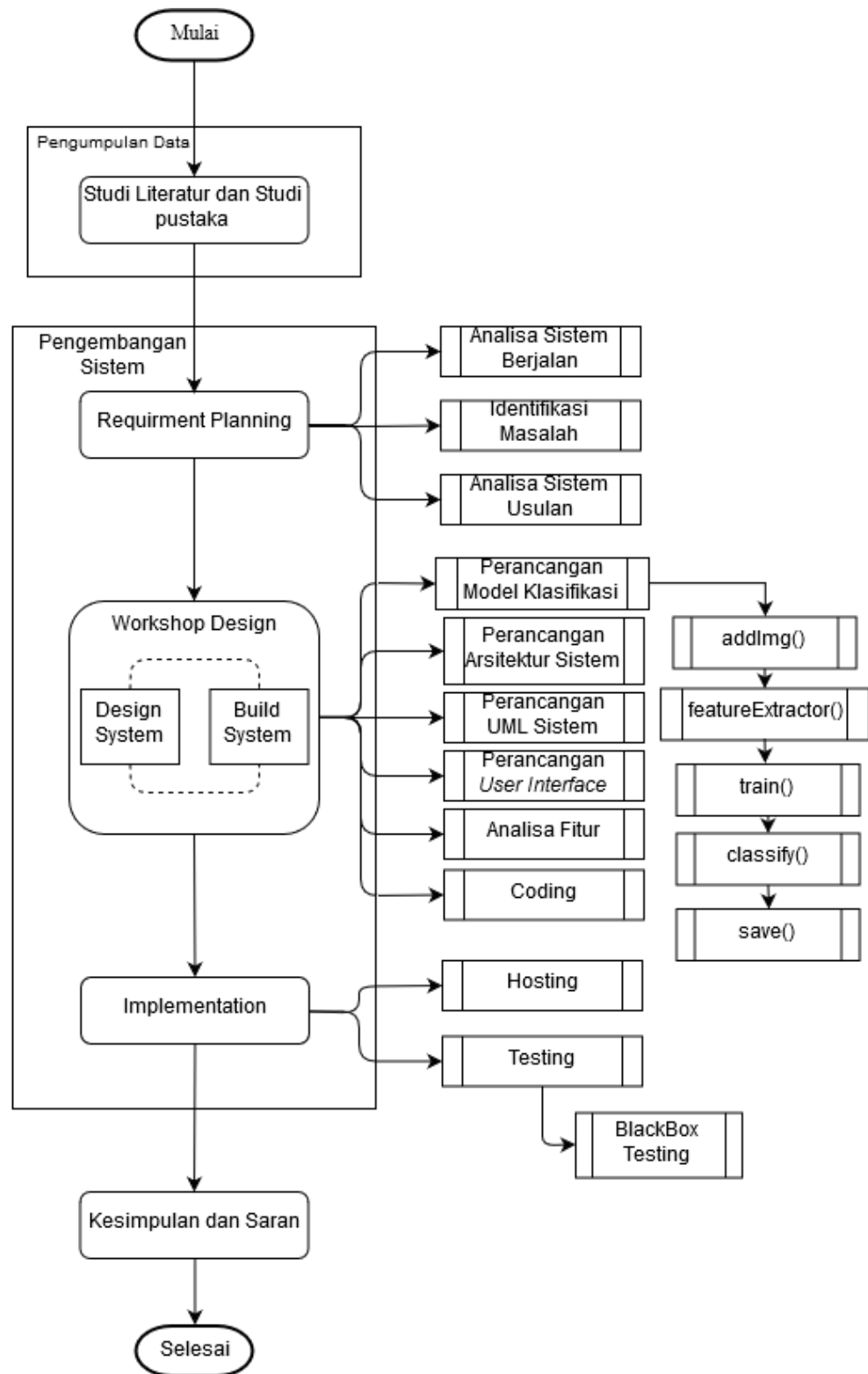
1. Tahap hosting pada Heroku sebagai hosting aplikasi, dan menampilkan hasil hari aplikasi dari setiap fitur.

2. Tahap pengujian sistem, untuk menjamin sistem dapat digunakan.  
Teknik pengujian yang peneliti pakai adalah teknik *black-box testing*.

### **3.3 Kerangka Berpikir**

Kerangka pemikiran menjelaskan tahap demi tahap yang dilakukan dalam penelitian. Berikut merupakan kerangka penelitian yang peneliti gunakan:





Gambar 3.1 Kerangka berfikir  
(Peneliti)

## BAB 4

### IMPLEMENTASI

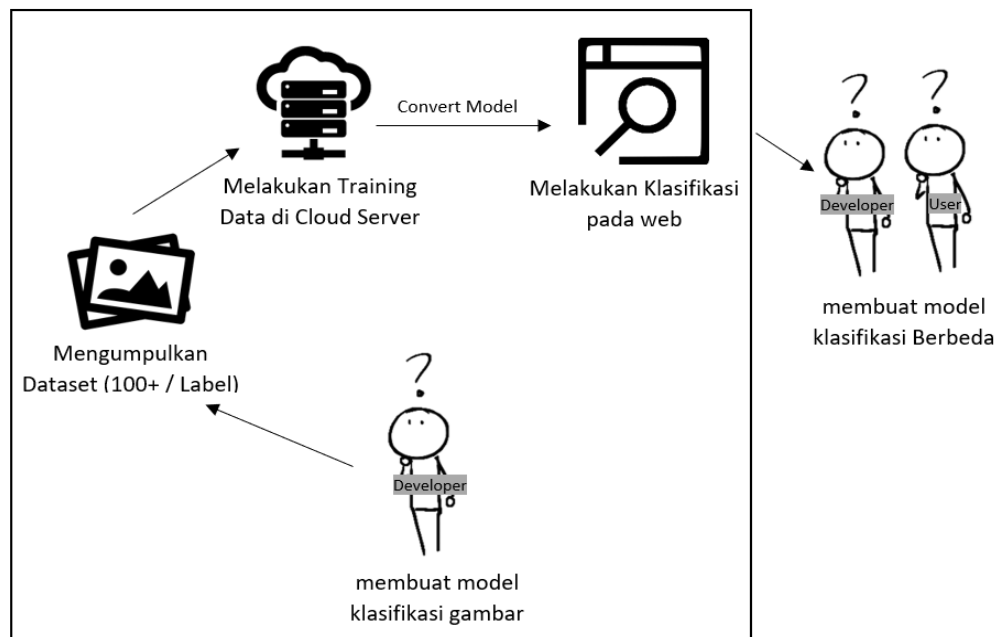
#### 4.1 Fase *Requirment Planning*

##### 4.1.1 Analisis Sistem Yang Berjalan

Machine learning khususnya klasifikasi gambar biasa di lakukan dengan metode server-side karena beberapa alasan:

1. Menggunakan deep learning seperti CNN sebagai arsitekturnya.
2. Membutuhkan banyak dataset untuk membuat suatu model.
3. Untuk melakukan training data dibutuhkan daya komputasi yang cukup besar karena banyaknya dataset.

Alur klasifikasi gambar pada umumnya bisa di lihat di bawah ini:



Gambar 4.1 server-side machine learning

- Untuk membuat model klasifikasi di butuhkan seorang developer yang mengerti programming dan alur dari pembuatan model tersebut maka itu di butuhkan developer dalam membuatnya.

- Dalam membuat model klasifikasi gambar di butuhkan banyak dataset gambar yang sudah diberikan label / kelas di setiap gambarnya.
- Setelah di dapatkan dataset maka model akan di buat di server-side yang pada umumnya menggunakan bahasa pemrograman R atau Python. Jika dataset yang di olah besar, akan di gunakan cloud server untuk melakukan komputasi seperti Google Cloud AI, ataupun Amazon Sage Maker dari AWS.
- Setelah data di olah dan menjadi model klasifikasi lalu untuk bisa di gunakan oleh user dibutuhkan sebuah platform maupun itu web ataupun android untuk menyediakan UI. Maka dari itu model akan di *convert* untuk keperluan platform yang di targetkan, pada penelitian ini website.
- Dari situ user dapat menikmati model yang developer buat secara spesifik. Jika user menginginkan model klasifikasi gambar lain maka developer harus mengulang proses dari awal.

#### 4.1.2 Identifikasi Masalah

Berdasarkan analisis dari sistem berjalan di atas, didapatkan beberapa masalah antara lain:

- User hanya dapat menggunakan model klasifikasi yang di sediakan oleh developer.
- Untuk membuat sebuah model klasifikasi gambar di butuhkan dataset yang besar.
- Dibutuhkannya daya komputasi besar untuk mengolah banyak data gambar.
- Harus di lakukan *convert* model untuk bisa melakukan prediksi di platform yang memiliki *interface*.
- Dibutuhkannya pengetahuan di bidang machine learning untuk bisa membuat model klasifikasi.

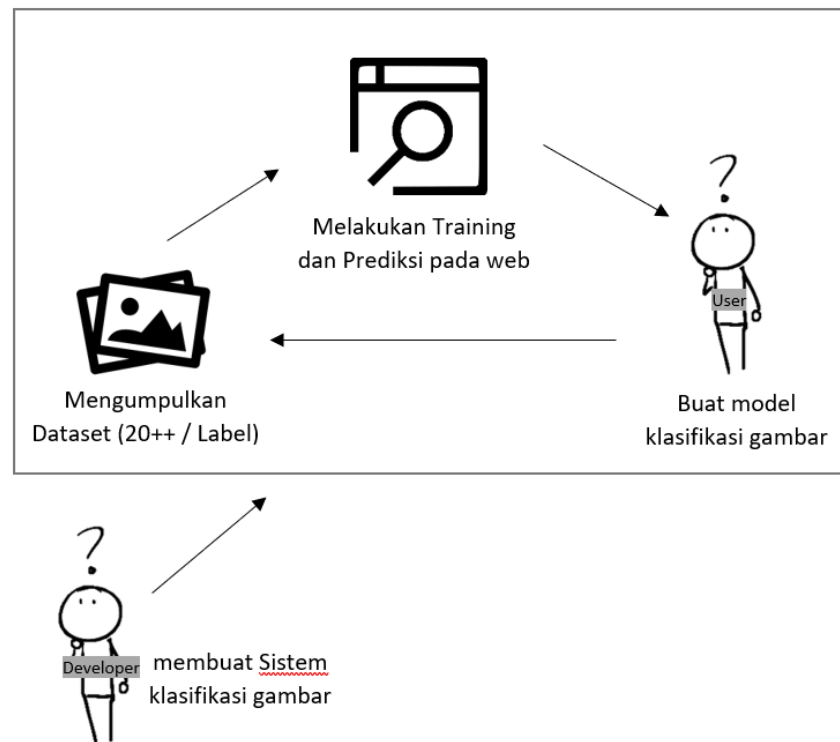
Karena itu dalam penelitian ini peneliti menggunakan ml5.js untuk melakukan client-side machine learning dan *Transfer Learning* dengan MobileNet untuk menyelesaikan masalah-masalah di atas. Dengan menggunakan *Client-side machine learning* peneliti dapat mengefisienkan

beberapa proses yang ada di sistem yang berjalan. *Client-side machine learning* dapat dilakukan karena kemampuan web browser modern yang mampu mengakses GPU, untuk melakukan animasi 3D dan perhitungan numerik dengan cepat. Untuk melakukan perhitungan numerik dengan cepat browser kini dilengkapi dengan WebGL yaitu sebuah API yang mengekspos OpenGL ke JavaScript yang memungkinkan *web browser* melakukan perhitungan menggunakan GPU.

Untuk melakukan klasifikasi gambar yang akan di bangun di web, peneliti menggunakan VSCode editor sebagai code editornya, Microsoft Visio untuk membuat UML, dan Adobe XD untuk merancang antarmuka. Sedangkan untuk web server peneliti menggunakan Node.js dan Express sebagai web server. Untuk *hosting* peneliti menggunakan Heroku karena sangat cocok untuk *hosting* web app Node.js tanpa *framework front-end*.

#### 4.1.3 Analisis Sistem Usulan

Berdasarkan identifikasi masalah di atas, peneliti bertujuan untuk mengimplementasikan klasifikasi gambar menggunakan client-side machine learning dengan pre-trained model MobilNet. Dengan menggunakan client-side machine learning agar pembuatan model klasifikasi dapat dilakukan tanpa pengkodean sedikitpun. Untuk melakukan itu peneliti menggunakan *website* sebagai *platformnya*, dengan ml5 sebagai *library* machine learning yang merupakan kumpulan kode Tensorflow.js



Gambar 4.2 Client-side machine learning usulan

- Developer membuat sebuah sistem klasifikasi gambar di dalam website (client-side) menggunakan ml5 dan MobileNet.
- User memasukan gambar datasetnya dengan jumlah sedikit.
- User melakukan training data dan menghasilkan model klasifikasi untuk selanjutnya di lakukan prediksi.
- User dapat menyimpan model klasifikasinya.
- Jika user ingin membuat model klasifikasi berbeda user dapat mengulangi proses dari awal.

#### 4.1.3.1 Analisis Fitur

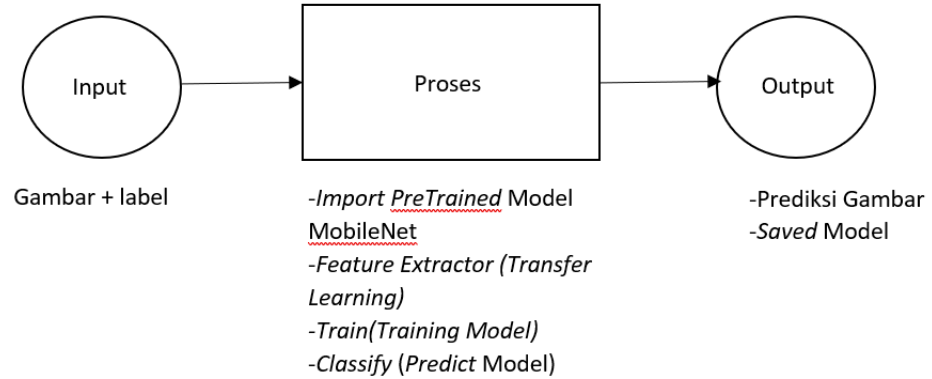
Dari analisis permasalahan yang sudah dikemukakan, penulis merancang fitur-fitur yang terdapat dalam aplikasi klasifikasi gambar. Fitur-fitur dari aplikasi pendeteksi kelelahan ini adalah *Add Label*, *Add Image*, *Training Model*, *Predict Model*, *Save Model*, dan *Load Model*.

Tabel 4.1 Tabel Analisis Fitur

No.	Fitur	Deskripsi
1	<i>Add Label</i>	Fitur ini digunakan untuk menambahkan label / class pada klasifikasi gambar.
2	<i>Add Image</i>	Fitur ini digunakan untuk menambahkan gambar pada label / class.
3	<i>Training Model</i>	Fitur ini digunakan untuk melakukan transfer learning dari <i>PreTrained</i> MobileNet ke dalam model baru yang di buat berdasarkan dataset yang user buat.
4	<i>Predict Model</i>	Fitur ini digunakan untuk mendapatkan prediksi dari model klasifikasi.
5	<i>Save Model</i>	Fitur ini digunakan untuk menyimpan model yang di <i>training</i> .
6	<i>Load Model</i>	Fitur ini digunakan untuk memuat model yang user punya untuk dilakukannya prediksi pada model klasifikasi tersebut.

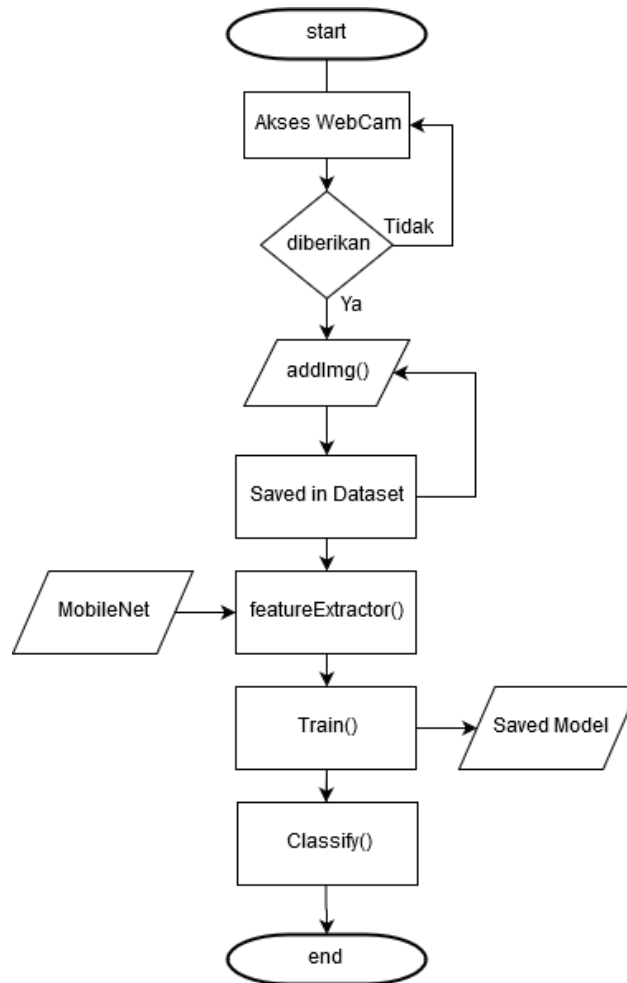
## 4.2 Fase Workshop Design

### 4.2.1 Perancangan Model Klasifikasi Gambar Dengan Pre Trained Model MobileNet



Gambar 4.3 Block Diagram Sistem

Pada sistem klasifikasi gambar dengan *client-side* dibutuhkan dataset berupa gambar yang telah di berikan label dan akan di konversi dengan ml5 menjadi 244 x 244 px pada pengambilan gambarnya untuk selanjutnya di proses oleh *feature extractor* dan MobileNet. Untuk melakukan klasifikasi pada sistem peneliti di butuh kan 3 proses dalam secara garis besar, yaitu *Import PreTrained Model*, *Feature Extractor*, *Train Model* dan *Classify*. Proses-proses tersebut di ambil dari nama fungsi di ml5.js



Gambar 4.4 Flowchart Klasifikasi Gambar

Seperti di lihat di gambar 4.4 sistem klasifikasi gambar di mulai dengan meminta izin pada user untuk menggunakan webcam pada komputer atau laptop. Lalu user dapat memasukan input gambar dan label yang akan menjadi dataset. Setelah dimasukan maka gambar-gambar tersebut akan langsung di ubah ukurannya oleh ml5 untuk selanjutnya dilakukan ekstraksi fitur pada masing-masing gambar yang ada di dalam dataset. Setelah gambar di ubah ukurannya maka dataset akan di latih menggunakan teknik *transfer learning* yang ada di dalam fungsi *featureExtractor()*. Sebelumnya *PreTrained* model MobileNet di import ke dalam sistem untuk selanjutnya di lakukan *transfer learning*. Dengan menggunakan *transfer learning* pembuatan model klasifikasi gambar hanya perlu menambahkan layer

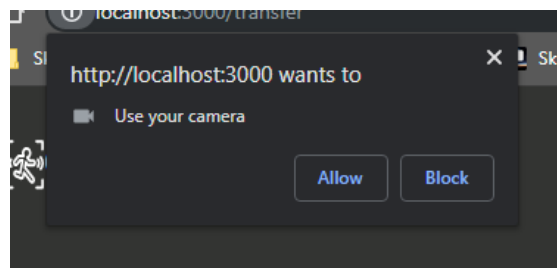


terakhir dari model yang sudah ada. Berikut adalah hasil percobaan peneliti dengan 2 label, mengikuti alur flowchart.

Dalam pembuatan klasifikasi menggunakan *Server-Side* tidak dibutuhkannya instalasi dependensi pada penggunaanya. Pengguna hanya memerlukan koneksi internet untuk mengakses *library* dan pre-trained model, browser, dan kamera webcam.

### 1. Akses Webcam

Pada awal penggunaan sistem user akan mendapatkan pop-up dari web browser untuk mengizinkan sistem menggunakan Webcam. Jika user tidak memiliki Webcam atau kamera lainnya sistem tidak akan berjalan karena input dari sistem ini hanya berasal dari kamera.



Gambar 4.5 Akses Kamera

Berikut adalah kode yang di gunakan untuk menggunakan kamera pada browser pengguna.

```
var video = document.querySelector("#videoStream");
// minta izin user
navigator.getUserMedia =
  navigator.getUserMedia ||
  navigator.webkitGetUserMedia ||
  navigator.mozGetUserMedia ||
  navigator.msGetUserMedia ||
  navigator.oGetUserMedia;
// jika user memberikan izin
if (navigator.getUserMedia) {
  // jalankan fungsi handleVideo, dan videoError jika izin ditolak
  navigator.getUserMedia(
    { video: true, Audio: false },
    handleVideo,
    videoError
  );
}
// fungsi ini akan dieksekusi jika izin telah diberikan
function handleVideo(stream) {
  video.srcObject = stream;
}
```

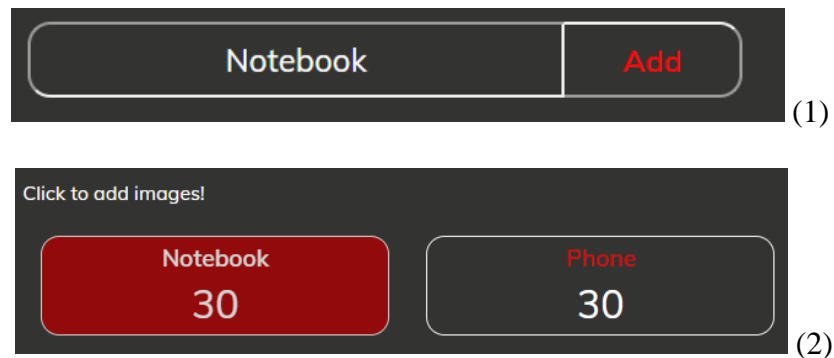
```

        video.play();
    }
    // fungsi ini akan dieksekusi kalau user menolak izin
    function videoError(e) {
        alert("Permission Denied!");
    }

```

## 2. Pengambilan Dataset

Pada proses pengumpulan data user dapat menambahkan nama label pada kolom input. Setelah menambahkan nama label maka sistem akan mengeluarkan box yang berisi nama label dan jumlah gambar yang di dalamnya.



Gambar 4.6 (1) Menambahkan Label. (2) Menambahkan Gambar  
Saat box dengan nama label di tekan maka fungsi *addImg()* akan aktif dan gambar akan di ambil dari Webcam user dan secara otomatis di *resize* dan di ubah menjadi tensor, sesuai dengan arsitektur model MobileNet yaitu 244x244px.



Gambar 4.7 Sebelum dan sesudah *resize*

Dan berikut adalah kode yang digunakan saat pengambilan gambar dan label. Dan melakukan *resize* pada gambar tersebut

```

// Capture Image dan Label
async addImageInternal(imgToAdd, label) {
  await this.ready;
  tf.tidy(() => {
    // Resize gambar
    const imageResize =
      imgToAdd === this.video ? null : [IMAGE_SIZE, IMAGE_SIZ
E];
    // Convert to Tensor
    const processedImg = imgToTensor(imgToAdd, imageResize);
    const prediction = this.mobilenetFeatures.predict(process
edImg);
    let y;
    if (this.usageType === "classifier") {
      y = tf.tidy(() =>
        tf.oneHot(tf.tensor1d([label], "int32"), this.config.
numLabels)
      );

      if (this.xs == null) {
        this.xs = tf.keep(prediction);
        this.ys = tf.keep(y);
        this.hasAnyTrainedClass = true;
      } else {
        const oldX = this.xs;
        this.xs = tf.keep(oldX.concat(prediction, 0));
        const oldY = this.ys;
        this.ys = tf.keep(oldY.concat(y, 0));
        oldX.dispose();
        oldY.dispose();
        y.dispose();
      }
    }
  });
  return this;
}

```

### 3. Transfer Learning

Setelah dataset di kumpulkan 1-20 gambar per labelnya, setiap gambar akan memasuki tahap feature mapping di sini fungsi dari `featureExtractor()` digunakan. Sebelum gambar di ekstrak fiturnya sistem akan melakukan import beban dari Pre-Trained Model MobileNet dan arsitekturnya. Dalam proses ini setiap gambar akan mengikuti arsitektur MobileNet dan melakukan *Convolution* sesuai arsitektur pada MobileNet. Berikut adalah kode yang digunakan untuk melakukan import pada model MobileNet dan konfigurasinya.

```

// import MobileNet dengan settingnya
class Mobilenet {
  constructor(options, callback) {
    this.mobilenet = null;
    this.topKPredictions = 10;
  }
}

```

```

// Cut bottleneck
this.hasAnyTrainedClass = false;
this.customModel = null;
this.jointModel = null;

// Konfigurasi
this.config = {
  epochs: options.epochs || DEFAULTS.epochs,
  version: options.version || DEFAULTS.version,
  hiddenUnits: options.hiddenUnits || DEFAULTS.hiddenUnits,
  numLabels: options.numLabels || DEFAULTS.numLabels,
  learningRate: options.learningRate || DEFAULTS.learningRate,
  batchSize: options.batchSize || DEFAULTS.batchSize,
  layer: options.layer || DEFAULTS.layer,
  alpha: options.alpha || DEFAULTS.alpha
};

// check if a mobilenet URL is given
this.mobilenetURL =
  options.mobilenetURL ||
  `${BASE_URL}${this.config.version}_${this.config.alpha}_${IM
AGE_SIZE}/model.json`;
this.graphModelURL = options.graphModelURL || this.url;

this.isPredicting = false;
this.mapStringToIndex = [];

this.usageType = null;
this.ready = callCallback(this.loadModel(), callback);
}

```

Pada arsitektur CNN mobilenet convolution di pisah menjadi 3 bagian yaitu:

1) *Standard Convolution [3x3]*

Hanya di lakukan di layer awal *convolution*

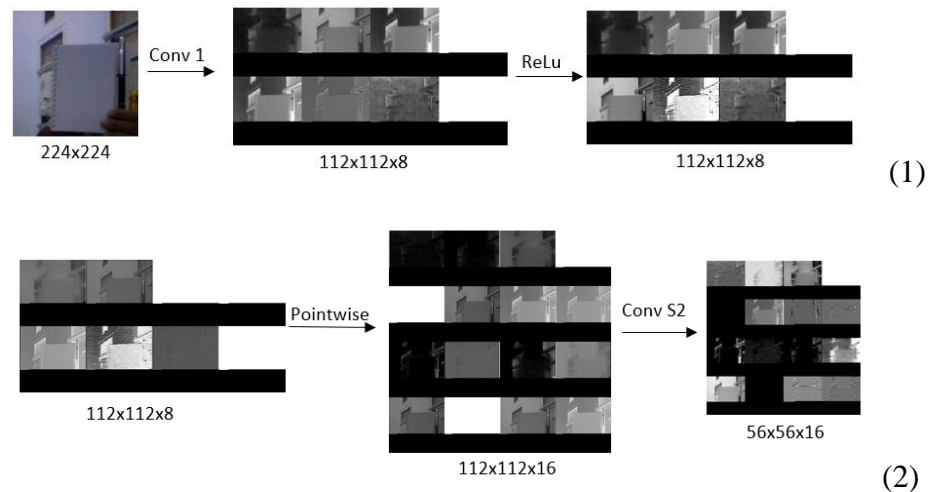
2) *Depthwise Convolution [3x3] Stride 1/Stride 2*

Memiliki 2 tipe yaitu Stride 1 dan Stride 2

3) *Pointwise Convolution [1x1]*

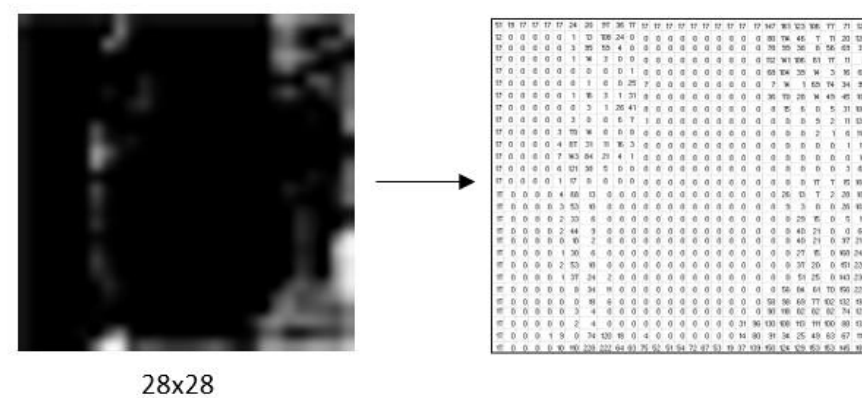
Di lakukan untuk memperbanyak fitur pada gambar.

Dan setiap convolution akan memiliki konfigurasinya masing-masing seperti padding, filter, channel, dan stride.



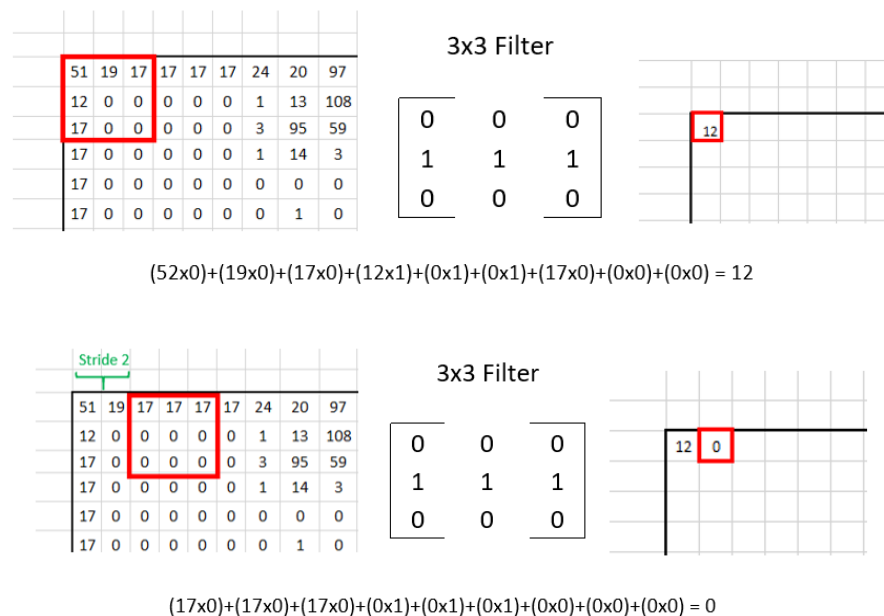
Gambar 4.8 (1) Hasil *Standard Convolution*. 1 (2) Hasil *Depthwise* dan *Pointwise Convolution*

Dari gambar pertama di atas dilakukan Convolution pertama pada gambar dan menghasilkan gambar dengan dimensi 112x112px dan 8 fitur. Lalu dari hasil convolution pertama akan diterapkan ReLu. Selanjutnya di gambar kedua merupakan rangkaian *Depthwise Convolution*, yang meliputi *Pointwise Convolution* yang akan menghasilkan 16 fitur pada gambar dan selanjutnya diterapkan *Convolution* dengan Stride 2 yang akan memperkecil dimensi gambar. Karena dimensi gambar yang cukup besar peneliti akan mencoba mendemonstrasikan salah-satu tahapan di atas dengan dimensi gambar yang lebih kecil.



Gambar 4.9 Matriks pada gambar

Setelah matriks pada gambar di dapatkan maka akan di lakukan *convolution*. *Regular convolution* pada MobileNet memiliki 3x3 filter dengan padding 1 pada pada prosesnya. Padding sendiri berfungsi agar gambar tidak mengecil pada saat di lakukan convolution. Untuk contoh peneliti akan menghitung convolution menggunakan Stride 2 tanpa padding yang akan menghasilkan matriks lebih kecil pada outputnya.



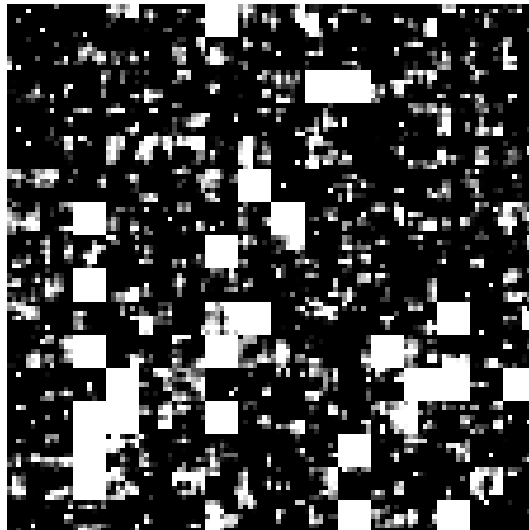
Gambar 4.10 Convolution dengan Stride 2

12	0	1	122	132	0	0	0	0	0	194	60	38	0
17	0	1	18	3	0	0	0	0	0	253	208	89	0
17	0	0	0	0	45	52	0	0	0	38	0	36	0
17	0	0	0	0	0	0	0	0	0	89	0	60	0
17	0	0	0	0	0	0	0	0	0	0	194	35	0
17	0	0	0	0	0	0	0	0	0	0	253	60	0
17	0	0	0	0	0	0	0	0	0	0	2	208	0
17	0	0	0	235	122	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	1	0
17	0	0	0	22	0	0	0	0	0	0	0	3	0
17	0	3	0	2	0	0	0	0	0	0	0	5	0
17	0	1	31	9	0	0	0	0	0	0	0	194	0
17	0	0	0	0	0	0	0	0	56	232	229	253	0
17	0	0	10	55	0	0	0	89	96	22	54	154	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0

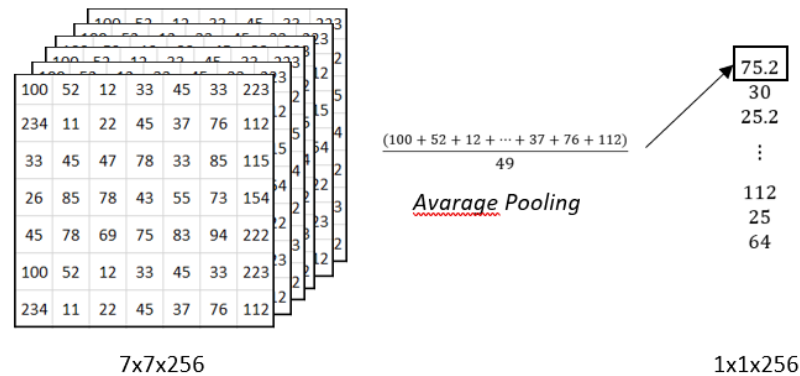
14x14

Gambar 4.11 Contoh hasil *Convolution*

Setelah semua proses convolution dengan arsitektur MobileNet akan di dapatkan hasil akhir dengan dimensi gambar 7x7 dengan 256 fitur.

Gambar 4.12 Hasil *Convolution* ke 13 pada contoh gambar dalam MobileNet

Selanjutnya di lakukan *flattening* dengan Average Pool menggunakan filter 7x7 sehingga di dapatkan matriks 1x256.

Gambar 4.13 penerapan *flattening*

#### 4. Train Model

Pada tahap training model lapisan pada Pre-Trained Model akan menggunakan pengaturan baru sesuai pembuatnya. Pengaturan seperti *batch size*, *epoch*, *learning rate*, dan *hidden layer* akan di butuhkan pada tahap ini. Untuk itu peneliti memberikan user kebebasan untuk mengatur sendiri pengaturan yang di inginkan. Pada penelitian ini peneliti menggunakan adam sebagai *optimizer*, ReLu sebagai *activation dense layer* pertama, dan Softmax sebagai *activation dense layer* terakhir yang merupakan *node output*. Berikut adalah kode pada saat neural network melakukan training pada model pada *fully connected layer*.

```

async train(onProgress) {
  if (!this.hasAnyTrainedClass) {
    throw new Error("Add some examples before training!");
  }
  this.isPredicting = false;
  if (this.usageType === "classifier") {
    this.loss = "categoricalCrossentropy";
    this.customModel = tf.sequential({
      layers: [
        tf.layers.flatten({ inputShape: [7, 7, 256] }),
        tf.layers.dense({
          units: this.config.hiddenUnits,
          activation: "relu",
          kernelInitializer: "varianceScaling",
          useBias: true
        }),
        tf.layers.dense({
          units: this.config.numLabels,
          kernelInitializer: "varianceScaling",
          useBias: false,
          activation: "softmax"
        })
      ]
    });
  }
}

```



```

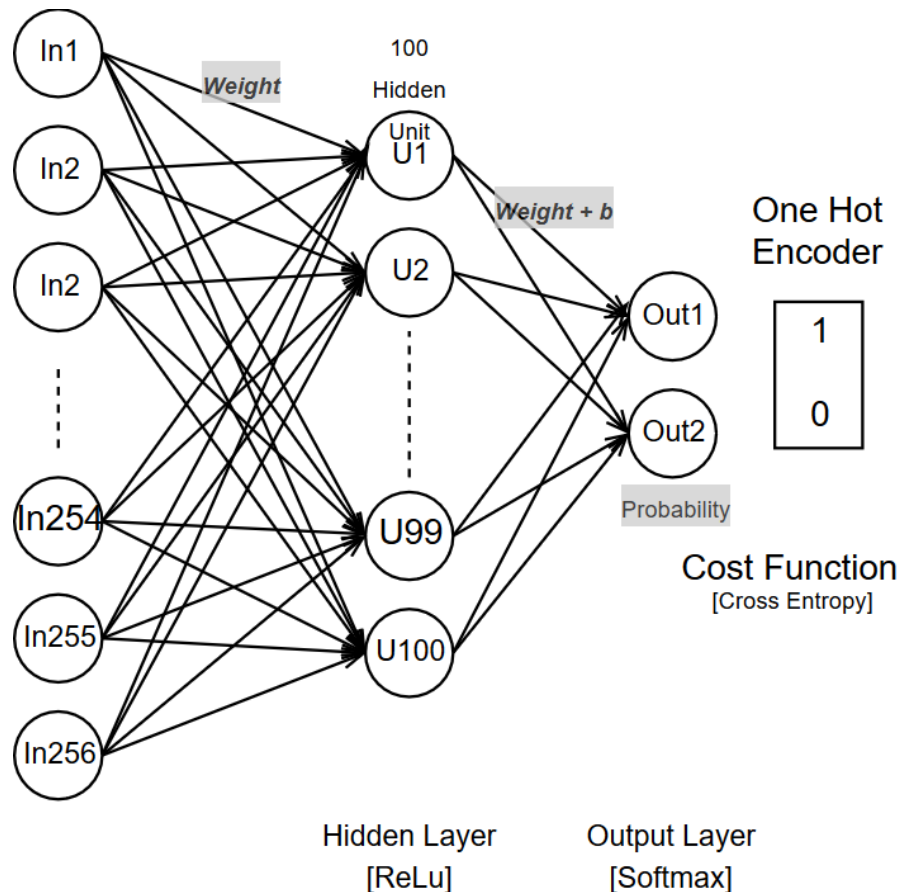
    })
  ]
});
}
this.jointModel = tf.sequential();
this.jointModel.add(this.mobilenetFeatures); // mobilenet
this.jointModel.add(this.customModel); // transfer layer

const optimizer = tf.train.adam(this.config.learningRate);
this.customModel.compile({ optimizer, loss: this.loss });
const batchSize = Math.floor(this.xs.shape[0] * this.config
.batchSize);
if (!(batchSize > 0)) {
  throw new Error(
    "Batch size is 0 or NaN. Please choose a fill the batch
size."
  );
}

return this.customModel.fit(this.xs, this.ys, {
  batchSize,
  epochs: this.config.epochs,
  callbacks: {
    onBatchEnd: async (batch, logs) => {
      onProgress(logs.loss.toFixed(5));
      await tf.nextFrame();
    },
    onTrainEnd: () => onProgress(null)
  }
});
}
}

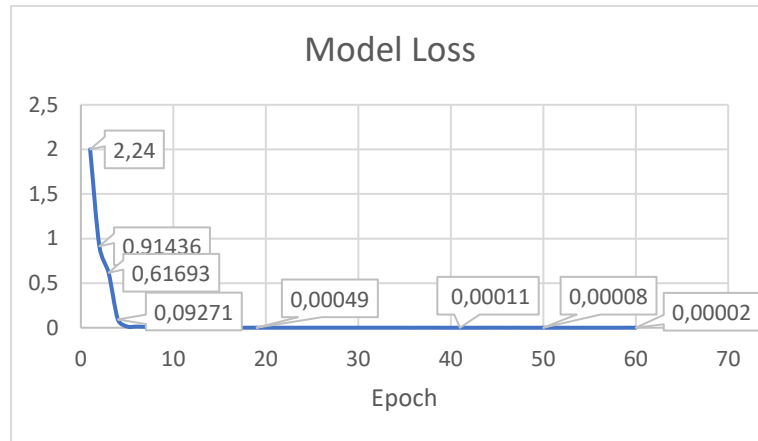
```

Fully connected layer pada transfer learning hanya membutuhkan 3 node yaitu Input (dari convolution), dense layer dengan ReLu, dan dense layer dengan Softmax.



Gambar 4.14 Rancangan *Fully Connected layer* pada penelitian

Pada gambar 4.14 akan lakukan proses training dan *backpropagation* pada model yang dibuat dengan tujuan mengurangi *loss / cost function*, dan pada lapisan terakhir akan di gunakan fungsi aktivasi *Softmax* yang akan mengubah beban / logit menjadi probabilitas untuk mengklasifikasikan gambar pada labelnya. Setelah mendapatkan probabilitas maka akan di gunakan *one hot encoder* untuk mengaktifkan label pada neural network. Output dari proses ini adalah *loss function* menggunakan *optimizer cross entropy* yang merupakan besarnya error per *epoch* pada model yang telah di training.



Grafik 4.1 *Loss Function* pada percobaan sistem

Pada percobaan ini peneliti menggunakan 2 *label*, 16 *batch size*, 60 *epoch*, 100 *hidden unit*, dan 0.0001 *learning rate*, sehingga menghasilkan nilai loss terakhir 0.00002. Setelah model klasifikasi selesai di buat maka hal terakhir yang di lakukan adalah memprediksi hasil dari klasifikasi yang akan di lakukan di proses selanjutnya.

### 5. *Classify Model*

Setelah model klasifikasi di buat maka prediksi akan di lakukan di proses ini. Menggunakan fungsi *classify()* pada ml5 model klasifikasi akan di gunakan untuk memprediksi gambar pada kamera webcam secara langsung. Pada proses ini tidak akan di berikan probabilitas karena prediksi di lakukan secara langsung. Berikut adalah code yang di gunakan pada saat melakukan klasifikasi pada model.

```

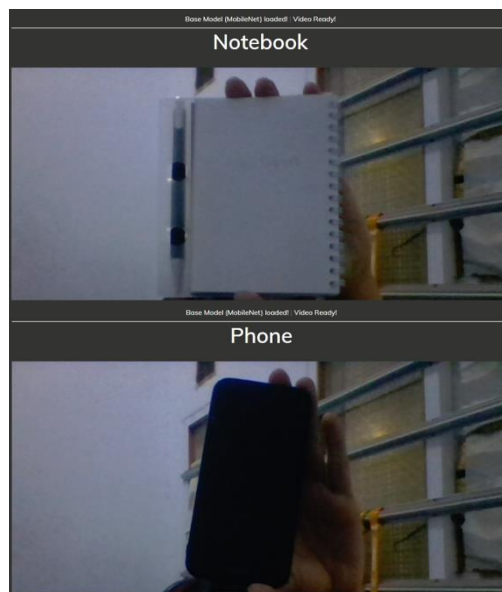
async classifyInternal(imgToPredict) {
  if (this.usageType !== "classifier") {
    throw new Error(
      "Mobilenet Feature Extraction has not been set to be a c
lassifier."
    );
  }
  await tf.nextFrame();
  this.isPredicting = true;
  const predictedClasses = tf.tidy(() => {
    const imageResize =
      imgToPredict === this.video ? null : [IMAGE_SIZE, IMAGE_
SIZE];
    const processedImg = imgToTensor(imgToPredict, imageResize
  );
    const predictions = this.jointModel.predict(processedImg);
    return Array.from(predictions.as1D().dataSync());
  });
}

```

```

const results = await predictedClasses.map((confidence, index) => {
  const label =
    this.mapStringToIndex.length > 0 && this.mapStringToIndex
    x[index]
      ? this.mapStringToIndex[index]
      : index;
  return {
    label
  };
});
return results;
}

```



Gambar 4.15 Hasil prediksi model klasifikasi

#### 6. *Save Model*

Setelah model di buat maka model bisa di ekspor dan di gunakan kembali. Model yang di keluar berupa 2 file yaitu model.json dan model.weight.bin yang masing-masing merupakan spesifikasi dari model klasifikasi dan beban pada model klasifikasi. Berikut ini adalah kode yang digunakan untuk melakukan ekspor pada model.

```

async save(callback, name) {
  if (!this.jointModel) {
    throw new Error("No model found.");
  }
  this.jointModel.save(
    tf.io.withSaveHandler(async data => {
      let modelName = "model";
      if (name) modelName = name;



      this.weightsManifest = {
        modelTopology: data.modelTopology,

```

```

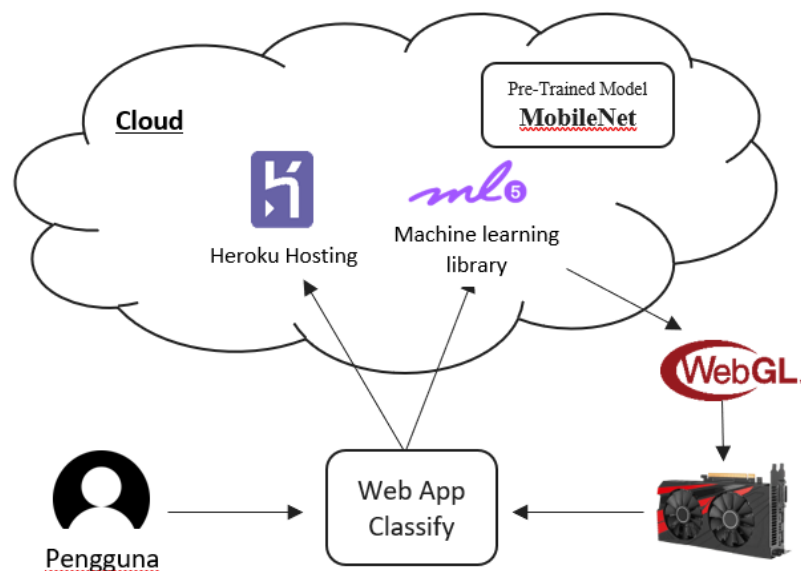
weightsManifest: [
  {
    paths: [`${modelName}.weights.bin`],
    weights: data.weightSpecs
  }
],
ml5Specs: {
  mapStringToIndex: this.mapStringToIndex
}
};
await saveBlob(
  data.weightData,
  `${modelName}.weights.bin`,
  "application/octet-stream"
);
await saveBlob(
  JSON.stringify(this.weightsManifest),
  `${modelName}.json`,
  "text/plain"
);
if (callback) {
  callback();
}
})
);

```

 model.json	Date modified: 03/12/2019 8:15
Type: JSON File	Size: 1,36 KB
 model.weights.bin	Date modified: 03/12/2019 8:15
Type: BIN File	Size: 4,78 MB

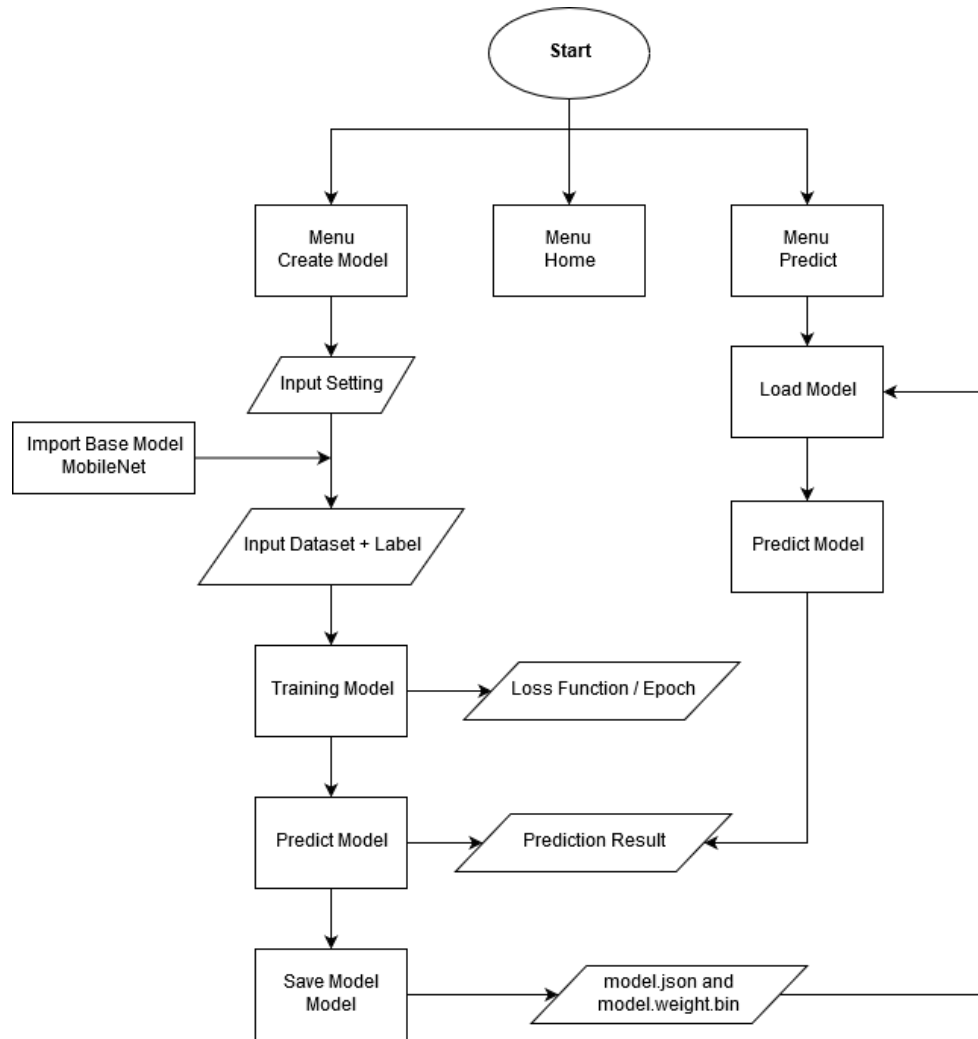
Gambar 4.16 File Hasil Ekspor Model Klasifikasi

#### 4.2.2 Perancangan Arsitektus Sistem



Gambar 4.17 Arsitektur sistem

Dalam arsitektur sistem, ml5 berperan besar untuk melakukan *client-side machine learning*. Dengan ml5 yang merupakan *library* dari tensorflow.js dapat mempermudah pengkodean machine learning di web. Dari pemrosesan gambar, penentuan layer, dan training model dilakukan oleh ml5. ML5 juga menghubungkan kode *machine learning* ke WebGL yang berfungsi untuk melakukan penghitungan dan pengolahan gambar menggunakan GPU yang memungkinkan *client-side machine learning* untuk bekerja dengan cepat. Sementara Pre-trained model MobileNet berperan dalam mempersingkat training model, sehingga sumber daya komputasi yang dibutuhkan lebih efisien dan cepat, karena dengan menggunakan pre-trained model pengguna hanya perlu memasukan sedikit dataset gambar yang akan di latih dan menggunakan transfer learning untuk membuat model baru untuk melakukan klasifikasi. Heroku *hosting* berfungsi untuk *hosting* aplikasi agar bisa di akses di manapun dengan secara *online*. Berikut adalah *flowchart* sistem aplikasi.



Gambar 4.18 Flowchart Aplikasi

#### 4.2.3 Perancangan UML

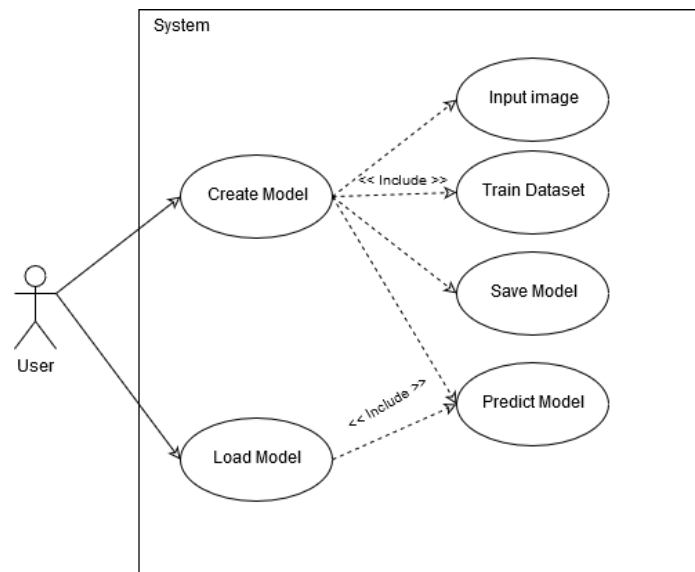
##### 4.2.2.1 Identifikasi Aktor

Pada sistem yang peneliti rancang terdapat 1 aktor, yaitu user

Tabel 4.2 Tabel Aktor

No.	Aktor	Deskripsi
1.	User	User adalah pengguna aplikasi yang akan memasukan dataset, melatih, dan memprediksi hasil dari klasifikasi gambar yang dibuatnya sendiri

#### 4.2.2.2 Usecase Diagram



Gambar 4.19 Usecase Diagram

Gambar 4.19 adalah usecase diagram yang berisi aktor dan aktivitas di dalam aplikasi yang peneliti buat.

#### 4.2.2.3 Usecase Scenario

*Usecase* scenario merupakan penjelasan lebih terperinci mengenai masing-masing *usecase* yang ada di *usecase* diagram pada suatu sistem. *Usecase* scenario terdiri dari:

- a. *Usecase Name*: nama usecase
- b. *Actor*: Actor yang terlibat
- c. *Description*: deskripsi usecase
- d. *Pre-Condition*: syarat penting bagi usecase untuk memulai
- e. *Alternative course*: kegiatan jika usecase gagal
- f. *Action*: kegiatan yang dilakukan oleh usecase
- g. *Post Condition*: kegiatan setelah usecase dilakukan

Berikut adalah *usecase* scenario dari *usecase* diagram pada gambar 4.19:



## 4.2.2.4 Usecase Create Model

Tabel 4.3 *Usecase Scenario Create Model*

<i>Usecase Name</i>	<i>Create Model</i>	
<i>Usecase Id</i>	1	
<i>Actor</i>	<i>User</i>	
<i>Description</i>	<i>Usecase</i> digunakan untuk membuat model klasifikasi gambar	
<i>Pre-Condition</i>	<i>Actor</i> memilih menu <i>Create Model</i>	
<i>Alternative Course</i>		
<i>Typical Course of Event</i>	<i>Actor Action</i>	<i>System Response</i>
	2) <i>Actor</i> mengizinkan sistem menggunakan kamera	1) Sistem menampilkan halaman <i>Create Model</i> dan meminta izin user untuk menggunakan kamera 3) sistem menampilkan tampilan <i>create model</i> beserta perekam kamera.

<i>Post Condition</i>	Jika usecase berhasil maka keterangan di atas box kamera akan berwarna putih dan kamera akan siap di gunakan
-----------------------	--

#### 4.2.3 Usecase Input image

Tabel 4.4 *Usecase Scenario Input Image*

<i>Usecase Name</i>	<i>Input Image</i>	
<i>Usecase Id</i>	2	
<i>Actor</i>	<i>User</i>	
<i>Description</i>	<i>Usecase</i> digunakan untuk memasukan gambar sebagai dataset untuk pembuatan model klasifikasi	
<i>Pre-Condition</i>	<i>Actor</i> memilih menu <i>Create Model</i>	
<i>Alternative Course</i>		
<i>Typical Course of Event</i>	<i>Actor Action</i>	<i>System Response</i>
	1) <i>Actor</i> memasukan nama label gambar pada text box yang di tentukan, dan menekan tombol “add” 3) <i>Actor</i> akan menekan tombol yang di buat oleh sistem untuk menambahkan gambar ke dalam dataset sesuai dengan label yang user masukan	2) Sistem akan menampilkan tombol untuk mengambil gambar, dengan sesuai nama label yang di masukan user. 4) sistem akan menyimpan satu gambar setiap user menekan

		tombol satu kali dan memasukkannya ke dalam dataset sesuai dengan labelnya
<i>Post Condition</i>	Jika usecase berhasil semua gambar sesuai dengan datasetnya akan tersimpan di dalam sistem dan siap di proses	

#### 4.2.4 Usecase Train Dataset

Tabel 4.5 *Usecase Train Dataset*

<i>Usecase Name</i>	Train Dataset	
<i>Usecase Id</i>	3	
<i>Actor</i>	<i>User</i>	
<i>Description</i>	<i>Usecase</i> digunakan untuk melakukan training pada dataset yang telah di masukan oleh user	
<i>Pre-Condition</i>	<i>Actor</i> memilih menu <i>Create Model</i>	
<i>Alternative Course</i>		
<i>Typical Course of Event</i>	<i>Actor Action</i>	<i>System Response</i>
	1) <i>Actor</i> menekan tombol “train model”	2) Sistem akan melakukan training dan <i>processing</i> pada dataset

		yang user masukan.
<i>Post Condition</i>	Jika usecase berhasil maka sistem akan mengeluarkan log berupa loss function dari proses training tersebut dan akan menghasilkan model yang siap di pakai	

#### 4.2.5 Usecase Predict

Tabel 4.6 *Usecase Predict*

<i>Usecase Name</i>	Predict	
<i>Usecase Id</i>	4	
<i>Actor</i>	User	
<i>Description</i>	Usecase digunakan untuk percobaan prediksi dengan model yang di buat user	
<i>Pre-Condition</i>	Actor memilih menu <i>Create Model</i>	
<i>Alternative Course</i>	Actor memilih menu <i>Predict</i>	
<i>Typical Course of Event</i>	<i>Actor Action</i>	<i>System Response</i>
	1) Actor menekan tombol “predict”	2) Sistem akan mengeluarkan hasil prediksi dari model yang telah di training berdasarkan objek yang di tangkap kamera dan

		model yang di buat user
<i>Post Condition</i>	Jika usecase berhasil maka sistem akan mengeluarkan prediksi dari model yang di buat user di atas box kamera	

#### 4.2.6 Usecase Save Model

Tabel 4.7 *Usecase Save Model*

<i>Usecase Name</i>	Save Model	
<i>Usecase Id</i>	5	
<i>Actor</i>	<i>User</i>	
<i>Description</i>	<i>Usecase</i> digunakan untuk menyimpan model yang telah di buat user untuk di gunakan kembali	
<i>Pre-Condition</i>	<i>Actor</i> memilih menu <i>Create Model</i> dan telah melakukan usecase Train Dataset	
<i>Alternative Course</i>		
<i>Typical Course of Event</i>	<i>Actor Action</i>	<i>System Response</i>
	1) <i>Actor</i> menekan tombol “save model” 3) <i>Actor</i> melakukan download pada file yang di berikan oleh sistem	2) Sistem akan mengeluarkan 1 file json dan bin yang merupakan model yang sudah di latih oleh user

<i>Post Condition</i>	Jika usecase berhasil maka sistem mengeluarkan file model berformat json dan bin yang merupakan keterangan model dan beban pada model
-----------------------	---

#### 4.2.7 Usecase Load Model

Tabel 4.8 *Usecase Load Model*

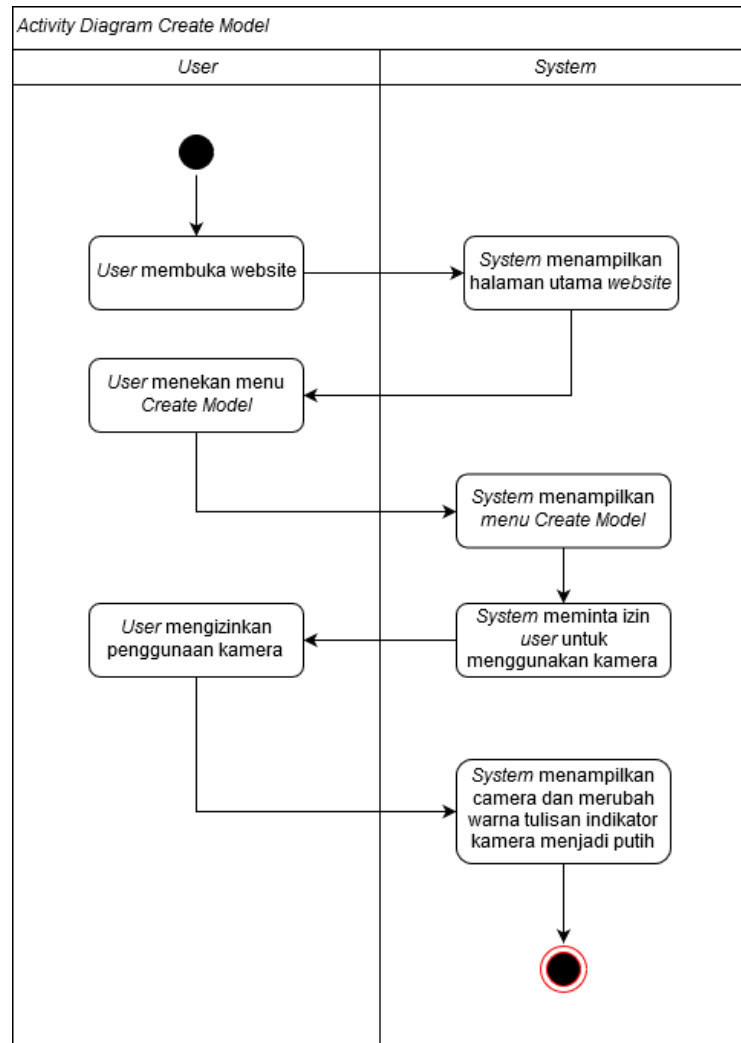
<i>Usecase Name</i>	Load Model	
<i>Usecase Id</i>	6	
<i>Actor</i>	<i>User</i>	
<i>Description</i>	<i>Usecase</i> digunakan untuk melakukan klasifikasi ulang pada model yang sudah jadi dan di simpan oleh user	
<i>Pre-Condition</i>	<i>Actor</i> memilih menu <i>Predict</i> dan telah melakukan usecase <i>Save Model</i>	
<i>Alternative Course</i>		
<i>Typical Course of Event</i>	<i>Actor Action</i>	<i>System Response</i>
	1) <i>Actor</i> menekan tombol “load model” 3) <i>Actor</i> memasukan file model yang berformat json dan bin pada sistem	2) Sistem akan mengeluarkan pop-up untuk user memasukan file model 3) Sistem akan meload model untuk di lakukannya klasifikasi

<i>Post Condition</i>	Jika usecase berhasil maka tulisan keterangan <i>costume model</i> akan berubah warna menjadi putih yang menandakan model siap digunakan
-----------------------	--

#### 4.2.7.1 Activity Diagram

Activity Diagram mengandalkan alur kerja (*workflow*) sebuah urutan aktivitas pada suatu proses. Diagram ini sangat mirip dengan flow chart karena kita dapat memodelkan logika, proses bisnis, dan alur kerja. Perbedaan utamanya adalah flowchart dibuat untuk menggambarkan alur kerja dari sistem, sedangkan activity diagram dibuat untuk menggambarkan aktivitas *actor*. Berikut ini adalah activity diagram yang menggambarkan aktivitas-aktivitas dalam sistem:

#### 4.2.7.2 Activity Diagram Create Model

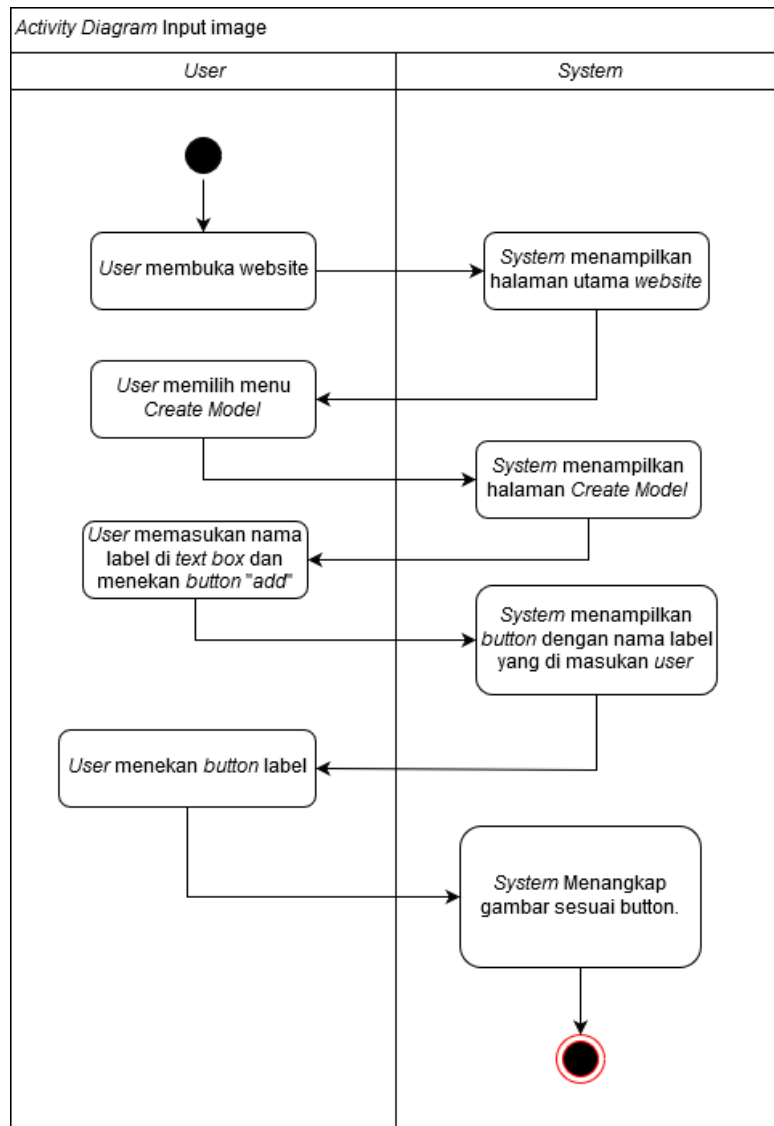


Gambar 4.20 Activity Diagram Create Model

Pada gambar 4.20 di atas, *activity diagram* tersebut menjelaskan tentang aktivitas yang harus dilakukan oleh *user* untuk dapat menjalankan menu *create model*. *User* hanya harus memberikan sistem izin untuk menggunakan kamera yang ada di komputer atau laptop untuk digunakan di dalam halaman ini. Jika *user* mengizinkan *system* untuk menggunakan kamera maka tulisan indikator di atas box kamera akan berubah warna menjadi putih menandakan kamera bisa digunakan.



#### 4.2.7.3 Activity Diagram Input image

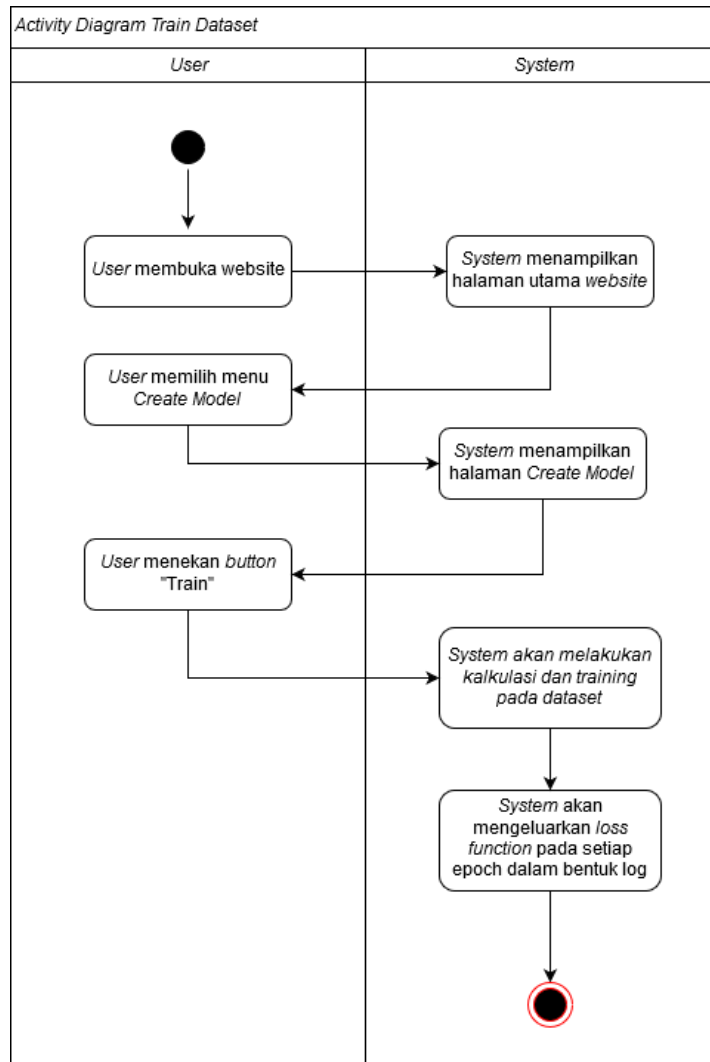


Gambar 4.21 Activity Diagram *Input Image*

Pada gambar 4.21 di atas, *activity diagram* tersebut menjelaskan tentang aktivitas yang harus dilakukan oleh *user* untuk melakukan input dataset dengan menambahkan label dataset dan foto pada setiap label yang dibuat. Setiap label yang di buat maka sistem akan menambahkan tombol dengan nama label tersebut sehingga user dapat menekan untuk menambahkan gambar pada dataset dengan label tersebut. Setiap klik yang di lakukan user akan menambahkan satu

gambar yang di tangkap oleh kamera ke dalam dataset dengan label tombol tersebut untuk di masukan ke dataset.

#### 4.2.7.4 Activity Diagram Train Dataset

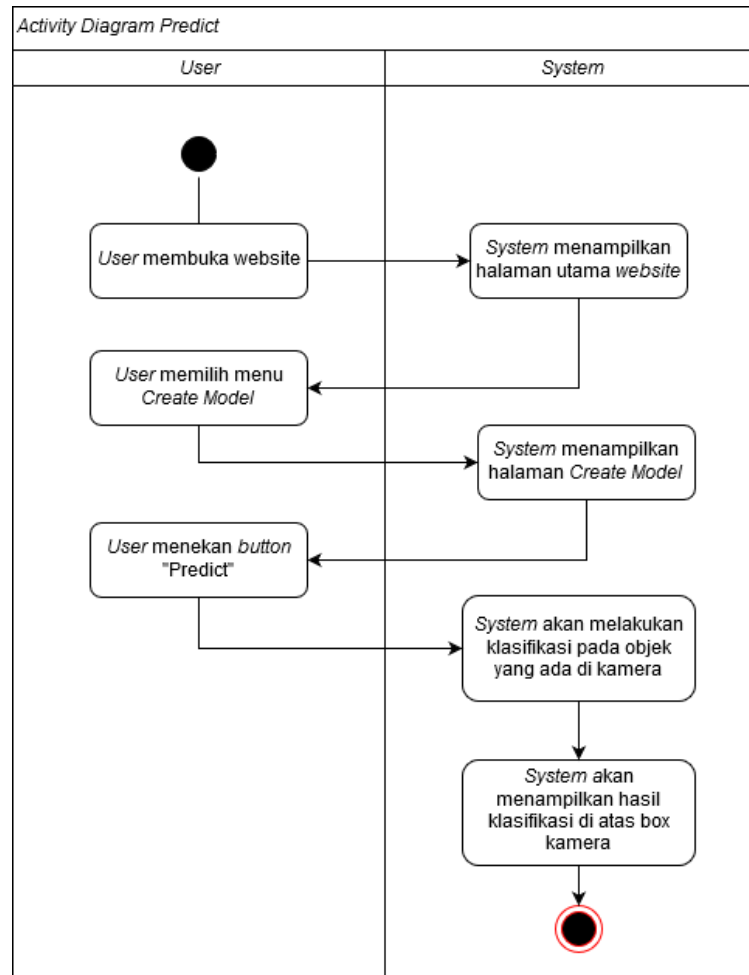


Gambar 4.22 Activity Diagram Train Dataset

Pada gambar 4.22 di atas, *activity diagram* tersebut menjelaskan tentang aktivitas yang di lakukan oleh *user* untuk dapat menjalankan training pada dataset yang telah di kumpulkan oleh user di diagram sebelumnya. Dengan menekan tombol “Train Model” maka sistem akan memulai untuk melakukan training terhadap dataset dan akan menghasilkan model klasifikasi sesuai lebel yang user berikan.

Pada tahap ini sistem akan mengeluarkan “loss function” akhir dan log pada setiap epoch yang di lakukannya.

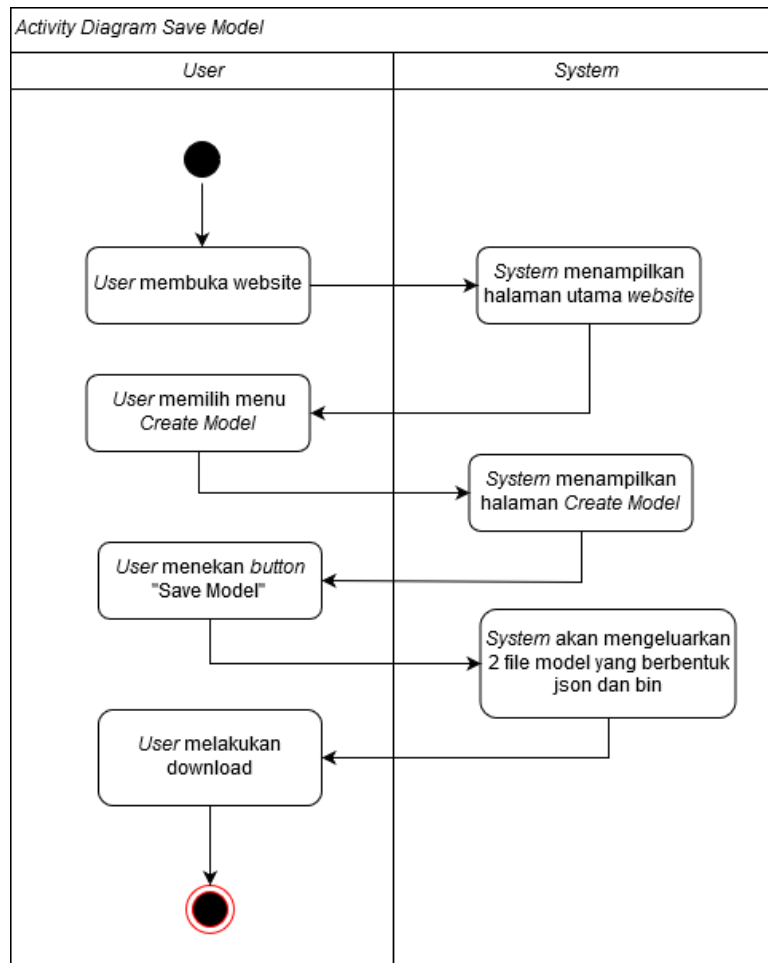
#### 4.2.7.5 Activity Diagram Predict



Gambar 4.23 Activity Diagram Predict

Pada gambar 4.23 di atas, *activity diagram* tersebut menjelaskan tentang aktivitas yang di lakukan oleh *user* untuk dapat menjalankan prediksi pada model klasifikasi yang sudah di *training* pada diagram sebelumnya. Pada tahap ini user menekan tombol “Predict” yang ada pada menu “Create Model” atau menu “Predict” maka sistem akan melakukan prediksi berbentuk label pada model yang sudah di *load* kedalam sistem dan mengeluarkan prediksi terhadap objek yang tertangkap di kamera.

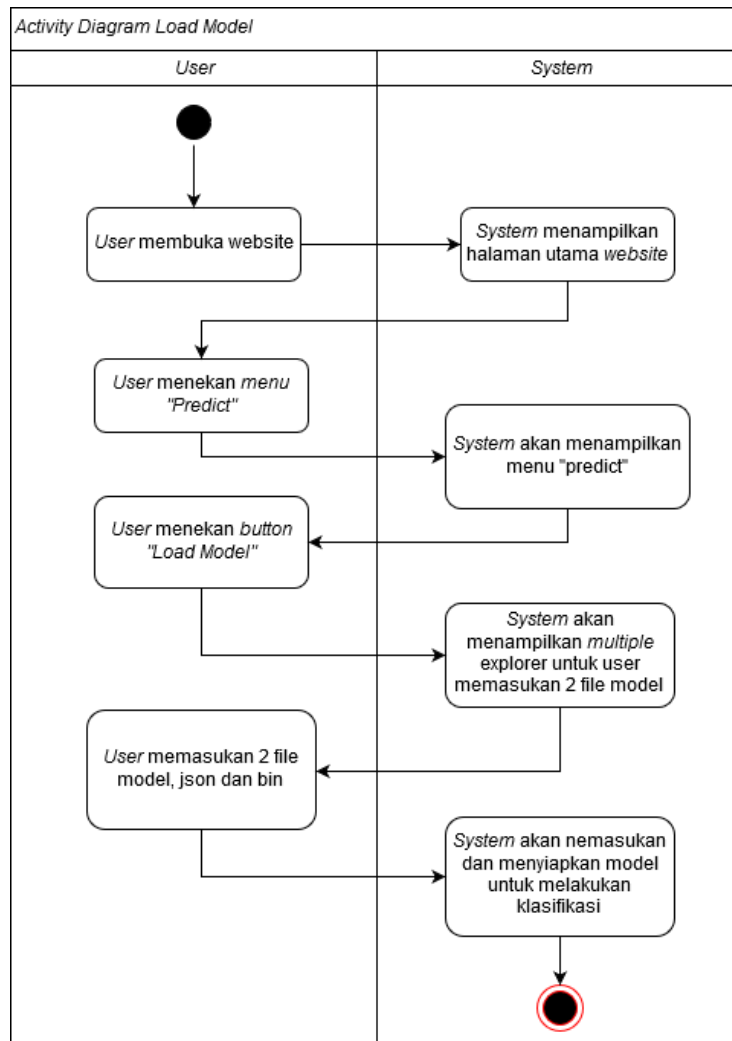
#### 4.2.7.6 Activity Diagram Save Model



Gambar 4.24 Activity Diagram Save Model

Pada gambar 4.24 di atas, *activity diagram* tersebut menjelaskan tentang aktivitas yang dilakukan oleh *user* untuk dapat menjalankan prediksi pada model klasifikasi yang sudah di *training* pada diagram sebelumnya. Pada tahap ini user menekan tombol “Predict” yang ada pada menu “Create Model” atau menu “Predict” maka sistem akan melakukan prediksi berbentuk label pada model yang sudah di *load* kedalam sistem dan mengeluarkan prediksi terhadap objek yang tertangkap di kamera.

#### 4.2.7.7 Activity Diagram Load Model

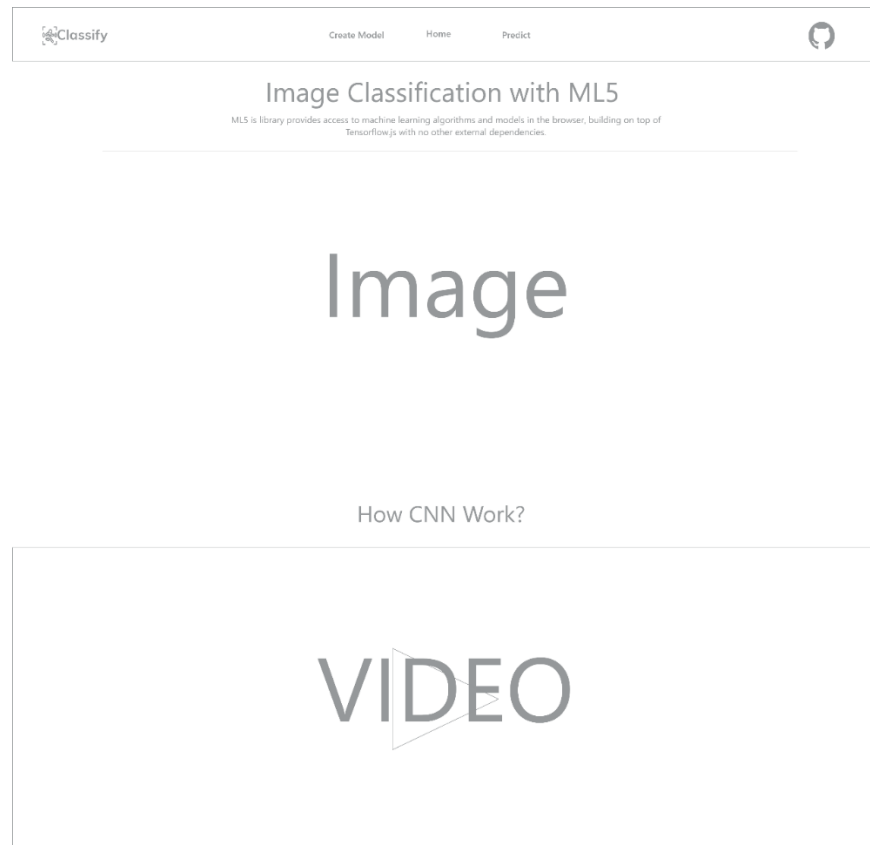


Gambar 4.25 Activity Diagram Load Model

Pada gambar 4.25 di atas, *activity diagram* tersebut menjelaskan tentang aktivitas yang dilakukan oleh *user* untuk dapat melakukan Load Model dengan model yang telah disimpan sebelumnya. Load Model dilakukan pada menu *predict* dengan menekan tombol “load model” dan memilih 2 file model yang di download di *activity* Safe Model dengan ekstensi .Json dan .bin

#### 4.2.4 Perancangan User Interface

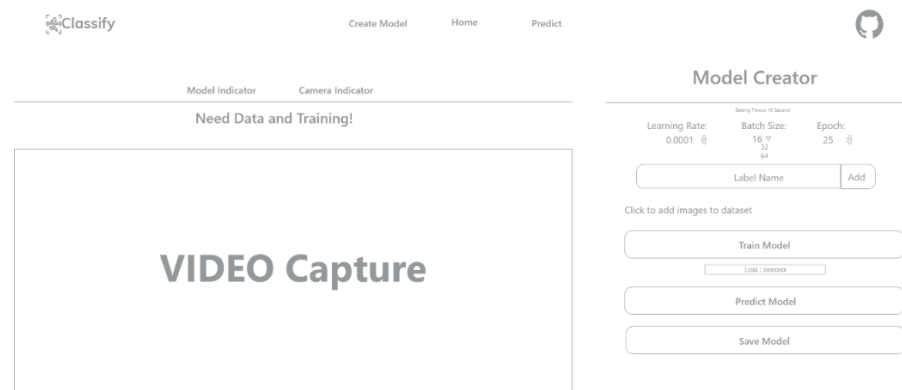
##### 4.2.4.1 Desain Interface Halaman Home



Gambar 4.26 Desain *Interface* Halaman *Home*

Pada gambar 4.26 merupakan rancangan tampilan halaman *home*, halaman ini berisi menu dasar aplikasi pada bagian atas dan keterangan berupa gambar dan video mengenai aplikasi dan cara kerjanya. Gambar menjelaskan bagaimana cara menggunakan aplikasi sementara video menjelaskan proses klasifikasi menggunakan metode yang di gunakan di aplikasi.

#### 4.2.4.2 Desain *Interface* Halaman *Create Model*



Gambar 4.27 Desain *Interface* Halaman *Create Model*

Pada gambar 4.27 merupakan rancangan tampilan halaman *Create Model*, halaman ini berisi menu dasar aplikasi pada bagian atas, 5 tombol, 1 text box, 4 indikator text dan 1 kontainer kamera. Text box berfungsi untuk menampung nama lebel yang akan di buat *user* dan tombol *add* untuk menambahkan label ke dalam sistem. Tombol *Train Model* digunakan untuk melakukan *training* pada *dataset* yang user buat, dengan berjalannya *training* pada *dataset* maka *loss function* indikator di bawahnya akan terus berubah hingga epoch terakhir. Tombol *Predict Model* digunakan untuk melakukan prediksi klasifikasi pada model yang telah di latih sebelumnya, sementara kamera berfungsi untuk menangkap gambar untuk melakukan kegiatan klasifikasi ataupun pengumpulan data set. Tombol *Save Model* digunakan untuk menyimpan model yang telah berhasil user buat. Sementara 3 indikator lainnya adalah model indikator, kamera indikator dan predict result yang masing masing berfungsi untuk melihat kondisi *pre-trained model*, kamera, dan hasil prediksi dari klasifikasi gambar.

#### 4.2.4.3 Desain *Interface* Halaman *Create Model - Input Image*

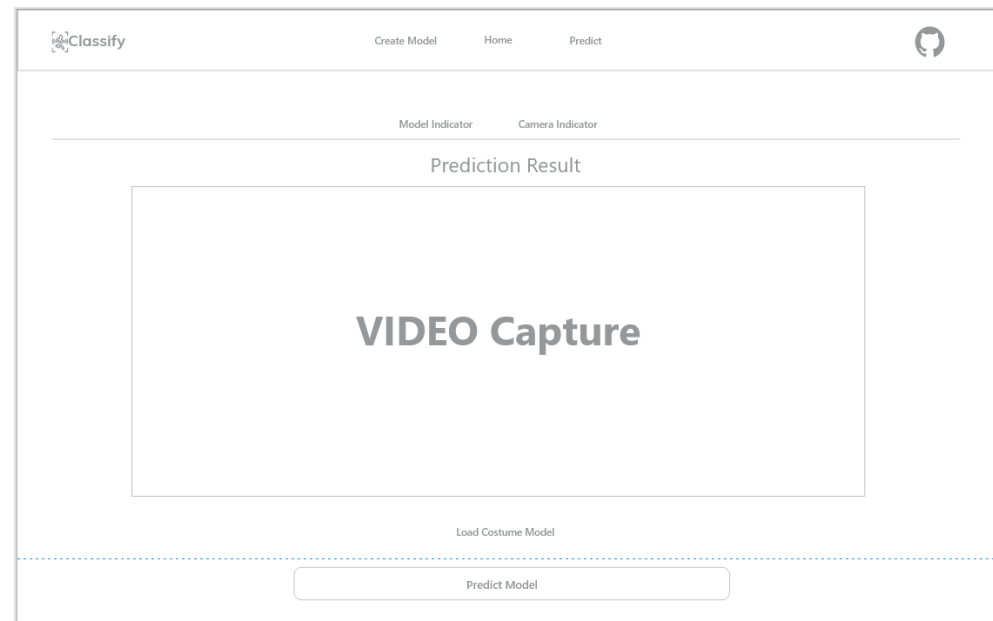
The interface is titled 'Classify' and includes navigation links for 'Create Model', 'Home', and 'Predict'. The 'Model Creator' section is titled 'Model Creator' and includes a 'Learning Rate' of 0.0001, a 'Batch Size' of 16, and an 'Epoch' of 25. Below these are four 'Label Name' input fields, each with a value of '20'. At the bottom of the 'Model Creator' section are buttons for 'Train Model', 'Predict Model', and 'Save Model'.

Gambar 4.28 Desain Interface Halaman Create Model - Input Image

Pada gambar 4.28 merupakan rancangan tampilan halaman *Create Model* yang sedang melakukan *input image* atau mengumpulkan *dataset* pada halaman ini setiap label yang di masukan *user* akan menghasilkan tombol baru sesuai dengan nama lebel yang di masukan oleh user. Setiap tombol tersebut mewakili lebelnya masing masing dan ketika di tekan kamera akan menangkap gambar dan menyimpannya sebagai dataset dari tombol yang user tekan, sementara di bawah nama lebel dari tombo adalah angka dari banyaknya gambar yang telah *user* ambil setiap label. Desain *Interface* Halaman *Predict*



#### 4.2.4.4 Desain *Interface* Halaman *Predict*



Gambar 4.29 Desain *Interface* Halaman *Predict*

Pada gambar 4.29 merupakan rancangan tampilan halaman *Predict*, halaman ini berisi menu dasar aplikasi pada bagian atas, 1 kontainer kamera, 3 indikator, dan 2 tombol. 3 indikator antara lain indikator model yang menampilkan status dari model yang akan di laod, kamera indikator, dan hasil prediksi. Tombol *Load Costume Model* berfungsi untuk melakukan *multiple load file* pada sistem *user* untuk mengupload model yang *user* buat sebelumnya. Tombol *Predict Model* berfungsi untuk menjalankan fungsi klasifikasi pada model yang telah *user upload*.

#### 4.2.5 Pengkodean

Setelah fase *design system* maka akan di lakukan *build system* atau pengkodean pada sistem. Pada tahap ini pengkodean menggunakan bahasa pemograman JavaScript dan terbagi menjadi 2 modul penting dalam sistem ini yaitu *Transfer.js* yang berperan dalam membuat model pada halaman *Create Model* dan *Predict.js* yang berperan untuk melakukan load model dan prediksi model pada halaman *Predict*. Berikut adalah gambar contoh pengkodean aplikasi yang dibuat oleh penulis:

- Penulisan code untuk Transfer.js

```

let featureExtractor;
let classifier;
let loss;
let addBtn;
let train;
let predict;
let input;
let video = document.querySelector("#videoStream");

let option = {
  learningRate: 0.0001,
  hiddenUnits: 100,
  epochs: 20,
  numClasses: 2,
  batchSize: 16
  // BatchSize 16,32,64,128,256
};

// Function Object for Setting
function lRate(val) {
  option.learningRate = parseFloat(val);
  console.log(option);
}
function epoch(val) {
  option.epochs = parseInt(val);
  console.log(option);
}
function batchSize(val) {
  option.batchSize = parseInt(val);
  console.log(option);
}
function numClass(val) {
  option.numClasses = parseInt(val);
  console.log(option);
}

// function classCount(count) {
//   var count = 0;
//   document.getElementById("class").innerHTML = count++;
// }

document.getElementById("lr").value = option.learningRate;
document.getElementById("epoch").value = option.epochs;
document.getElementById("batch").value = option.batchSize;
document.getElementById("classMax").value = option.numClasses;

var timeleft = 10;
var downloadTimer = setInterval(function() {
  document.getElementById("countdown").innerHTML =
    "<span style='color:red; font-size:15px;'>" + timeleft + "</span> seconds";
  timeleft -= 1;
  if (timeleft <= 0) {
    clearInterval(downloadTimer);
    document.getElementById("set").innerHTML =
      "<span style='color:#6e6e6e'>Setting</span>";
    document.getElementById("countdown").innerHTML =
      "<span style='color:#6e6e6e'>has been Setup and <span sty
le='color:red; font-size: 12px;'> can't be change </span> anymore</span>";

```

```

    }
  }, 1000);
function setup() {
  noCanvas();
  // Timeout for setting change
  setTimeout(() => {
    // Extract PreTrain features from MobileNet
    featureExtractor = ml5.featureExtractor("MobileNet", option
, modelReady);
    // Create a new classifier using those features and give th
e video we want to use
    classifier = featureExtractor.classification(video, vidRead
y);
  }, 10000);

  setupButtons();
}

//Element STATUS
// model
function modelReady() {
  select("#modelStatus").html("Base Model (MobileNet) loaded!")
;
  select("#modelStatus").style("color", "#fff");
}

// camera
function vidReady() {
  select("#videoStatus").html("Video Ready!");
  select("#videoStatus").style("color", "#fff");
}

// Classify the Trained Model
function classify() {
  classifier.classify(gotResults);
  select("#result").style("font-size", "3em");
}

// BUTTON FUNCTION for All Utilitys
function setupButtons() {
  let info = select("#info");
  let img = 0;
  addBtn = select("#addBtn");
  addBtn.mouseClicked(function() {
    let num = 0;
    if (select("#className").value() == "" || img >= option.num
Classes) {
      alert("You Reach the Maximum Class!");
    } else {
      info.html("Click to add images!");
      let className = select("#className").value();
      let imgItems = select(".imgItems");
      let imgItem = createElement("div").addClass("imgItem nose
lect");
      imgItem.parent(imgItems);
      let itemName = createElement("h3", className).addClass("i
temName");
      itemName.parent(imgItem);
      let imgNum = createElement("span", num).addClass("num");
      imgNum.parent(imgItem);
      img++;
      select("#className").value("");
      imgItem.parent(imgItems).mouseClicked(function() {

```

```

        num++;
        classifier.addImage(className);
        imgNum.html(num);
    });
}
});

// Train Button
train = select("#train");
train.mousePressed(function() {
    let ls = 0;
    classifier.train(function(lossValue) {
        if (lossValue) {
            loss = lossValue;
            select("#loss").html("Loss: " + loss);
            console.log("E " + ls + ". " + loss);
            ls++;
        } else {
            select("#loss").html(
                "Done Training! Final Loss: <span style='color: red;b
background-color: #3e3f42'>" +
                loss +
                "</span>"
            );
            select("#more").html(
                "Press <span style='color: red'>F12</span> to see los
s / epoch"
            );
        }
    });
});

// Predict Button
buttonPredict = select("#predict");
buttonPredict.mousePressed(classify);

// Save model
saveBtn = select("#save");
saveBtn.mousePressed(function() {
    classifier.save();
});
}

// Show the results
function gotResults(err, result) {
    if (err) {
        console.error(err);
    }
    if (result) {
        select("#result").html(result);
        classify();
    }
}
}

```

- Penulisan code untuk Predict.js

```

let mobilenet;
let classifier;

// Camera Setting
let video = document.querySelector("#videoStream");

```

```

function setup() {
  noCanvas();

  // Extract PreTrain features from MobileNet
  mobilenet = ml5.featureExtractor("MobileNet", modelReady);

  // Create a new classifier using those features and give the
  video we want to use
  classifier = mobilenet.classification(video, videoReady);

  // Button Setting
  setupButtons();
}

// Classify Train Test
function classify() {
  classifier.classify(gotResults);
  select("#result").style("font-size", "3em");
}

// A util function to create UI buttons
function setupButtons() {
  // Load Button
  loadBtn = select("#loadBtn");
  loadBtn = loadBtn.changed(function() {
    select("#modelStatus").html("Custom Model Loaded!");
    select("#modelStatus").style("color", "fff");
    select(".labelBtn").html("Click Here to Load Another Model.
  ..");
    classifier.load(loadBtn.elt.files, function() {
      select("#modelStatus").html("Custom Model Loaded!");
      console.log("loaded");
    });
  });

  // Predict Button
  buttonPredict = select("#predict");
  buttonPredict.mousePressed(classify);
}

// Model Ready
function modelReady() {
  select("#modelStatus").html("Waiting for Model");
}

// camera
function videoReady() {
  select("#videoStatus").html("Video ready!");
  select("#videoStatus").style("color", "fff");
}

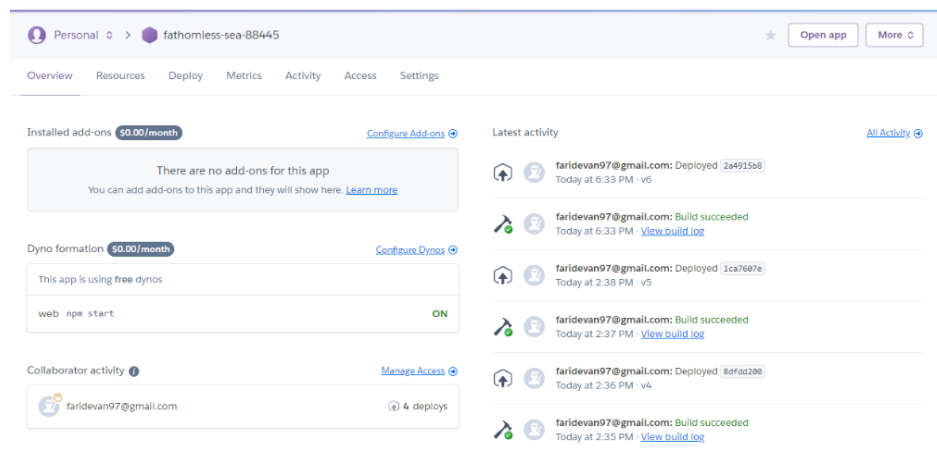
// Show the results
function gotResults(err, result) {
  if (err) {
    console.error(err);
    alert("Please Load The Model!");
  }
  if (result) {
    select("#result").html(result);
    classify();
  }
}

```

## 4.3 Fase Implementasi

### 4.3.1 Hosting

*Hosting* aplikasi di lakukan di Heroku.com karena merupakan layanan *cloud* yang gratis. Setelah di *hosting* aplikasi mendapatkan domain <https://fathomless-sea-88445.herokuapp.com/> dan peneliti singkat menggunakan bit.ly menjadi <https://bit.ly/clsfy>



Gambar 4.30 Dashboard Heroku

### 4.3.2 Testing

Pada tahap ini dilakukan pengujian aplikasi. Pengujian ini dilakukan untuk memastikan bahwa program dapat berjalan dengan baik saat digunakan. Dari setiap tes yang dilakukan tidak menutup kemungkinan terdapat kesalahan dari aplikasi yang sudah dites, namun dengan dilakukannya pengujian ini setidaknya dapat meminimalisir kesalahan yang ada pada aplikasi. Pengujian alpha di lakukan oleh peneliti sendiri dengan hasil sebagai berikut:

Tabel 4.9 Tabel Hasil Pengujian

No.	Link	Harapan	Hasil
1	Menu Home	User dapat melakukan navigasi ke Menu Home dengan baik	Sesuai

2	Menu <i>Create Model</i>	User dapat melakukan navigasi ke Menu Create Model dengan baik	Sesuai
3	Menu <i>Predict</i>	User dapat melakukan navigasi ke Menu Predict dengan baik	Sesuai
4	Tombol <i>Add Label</i>	Sistem berhasil menciptakan tombol Add Image sesuai dengan label yang di masukan user di text box	Sesuai
5	Tombol <i>Add Image</i>	Sistem berhasil menambahkan gambar sesuai dengan label dan indikator jumlah gambar pada tombol bertambah	Sesuai
6	Tombol <i>Train Model</i>	Sistem berhasil melakukan training pada dataset gambar yang user masukan dan indikator <i>loss function</i> berfungsi	Sesuai
7	Tombol <i>Predict Model</i>	Sistem berhasil melakukan prediksi pada model klasifikasi gambar yang user buat dengan menampilkan hasil prediksi di berupa String di atas kontainer kamera	Sesuai
8	Tombol <i>Save Model</i>	Sistem berhasil membungkus model dan	Sesuai

		memberikan user 2 file model untuk di download	
9	Tombol <i>Load Costume Model</i>	Sistem berhasil menerima dan memuat 2 file model yang user punya pada sistem untuk selanjutnya di lakukan prediksi pada gambar	Sesuai

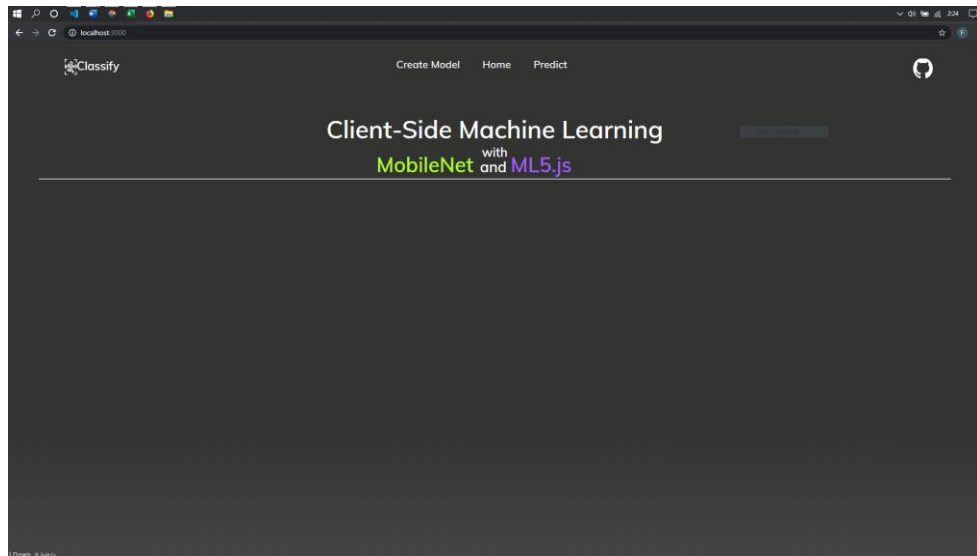


## BAB 5

### HASIL DAN PEMBAHASAN

#### 5.1 Hasil Tampilan *User Interface*

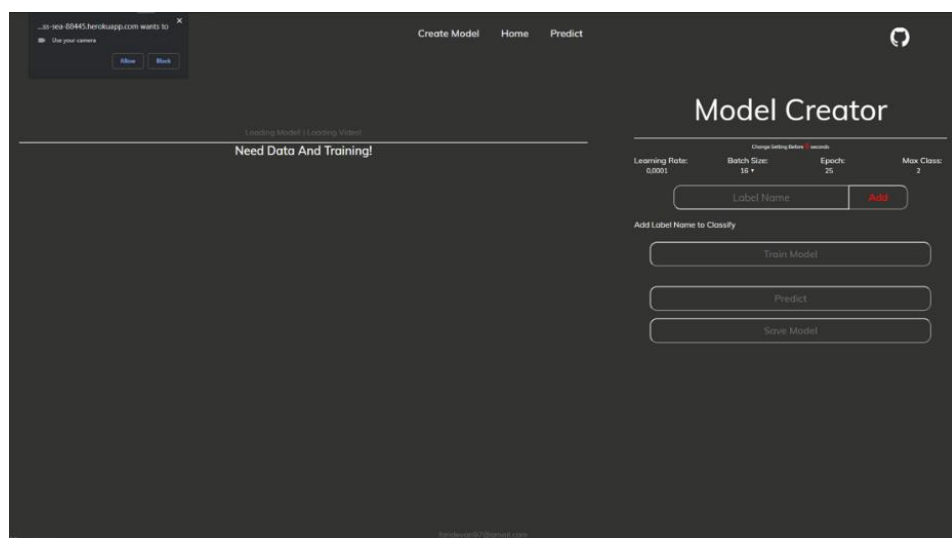
##### 5.1.1 Tampilan Halaman *Home*



Gambar 5.1 Tampilan Halaman *Home*

Gambar 5.1 merupakan tampilan halaman home yang digunakan untuk menyampaikan informasi tentang kegunaan website dan teori machine learning di dalamnya.

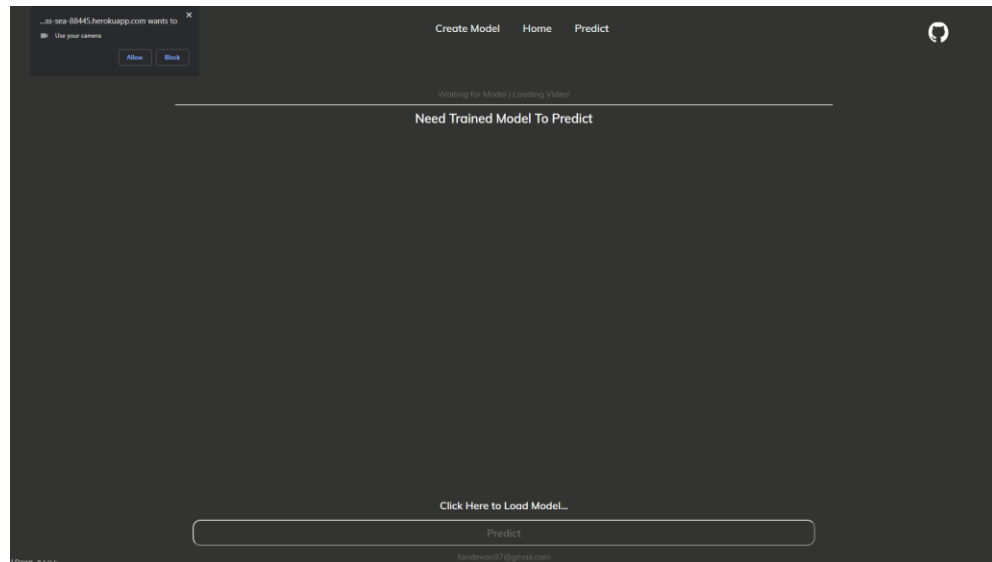
##### 5.1.2 Tampilan Halaman *Create Model*



Gambar 5.2 Tampilan Halaman *Create Model*

Gambar 5.2 Merupakan tampilan halaman *create model* yang di gunakan untuk membuat model klasifikasi, mencoba prediksi model klasifikasi, dan menyimpan model klasifikasi yang sudah di buat.

### 5.1.3 Tampilan Halaman *Predict*



Gambar 5.3 Tampilan Halaman *Predict*

Gambar 5.3 Merupakan tampilan halaman *predic* yang di gunakan untuk melakukan prediksi pada model klasifikasi yang sudah di buat sebelumnya.

## 5.2 Pengujian Sistem Dengan Skenario

Pengujian di lakukan dengan 3 skenario terhadap besar akhir *loss function* yang di hasilkan oleh sistem. pada pengujian ini pengaturan dasar sistem adalah *learning rate* 0.0001, *epoch* 20, dan 20 gambar tiap labelnya. Tiap scenario akan ada 2 variasi yaitu 2 label dan 4 label. Tujuan dari pengujian ini adalah mengetahui parameter apa yang menjadi faktor utama dari nilai *loss function* dari sistem klasifikasi yang peneliti buat.

Table 5.1 Data Percobaan

Parameter	Skenario 1	Skenario 2	Skenario 3
Epoch	20	40	60
Learning Rate	0.0001	0.001	0.01
Image	30	60	90

Setiap scenario akan ada 2 variasi yaitu 2 dan 4 label gambar benda yang di ambil langsung dari kamera webcam laptop. Benda yang di gunakan antara lain mouse, *smart phone*, jam tangan, dan *notebook*. Berikut adalah contoh gambar yang di ambil untuk skenario yang di buat.

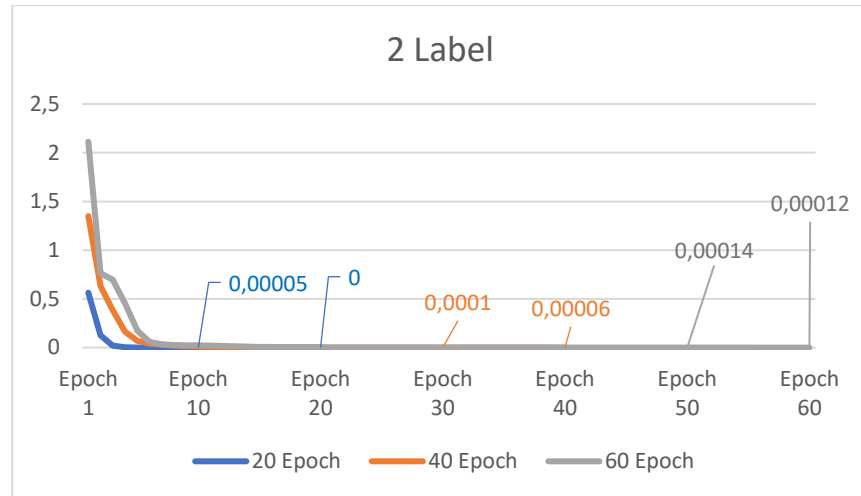


Gambar 5.4 Gambar pada skenario

#### 1. Pengujian terhadap *epoch*

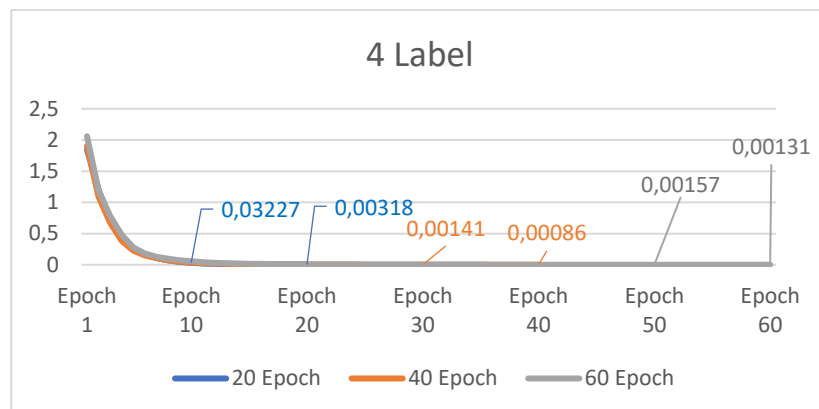
Pengujian terhadap epoch di lakukan dengan 2 label dan 4 label berikut adalah hasil dari masing-masing kategori.

## a. 2 Label



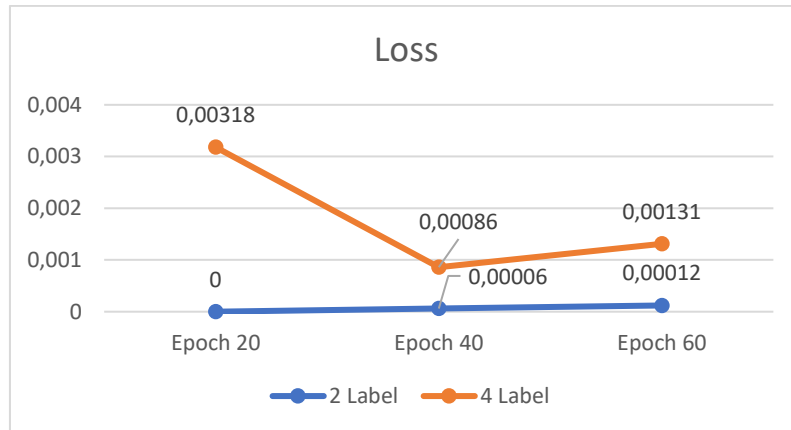
Grafik 5.1 Skenario pada 2 label

## b. 4 Label



Grafik 5.2 Skenario pada 4 label

Di lihat dari data di atas epoch cukup mempengaruhi loss function ketika label yang di training adalah 4 label. Sehingga dapat di simpulkan menggunakan epoch besar cukup berpengaruh dengan banyak label dalam menurunkan *loss function*.

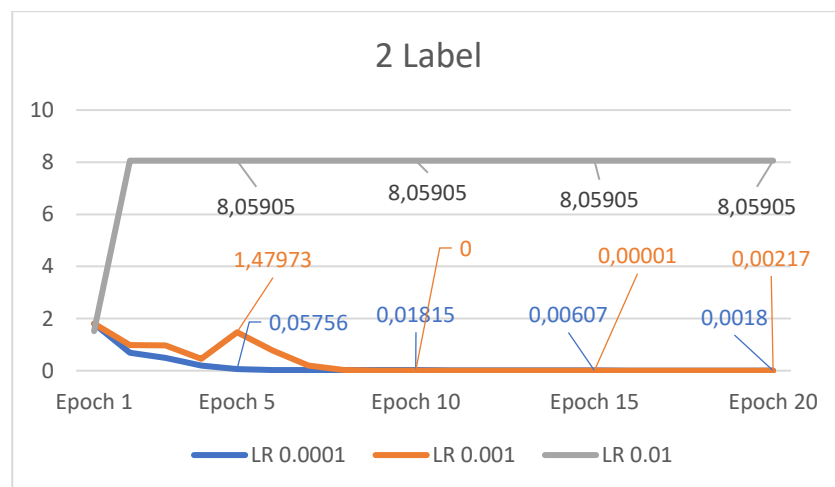


Grafik 5.3 Pengaruh *epoch* terhadap *loss function*

## 2. Pengujian terhadap *learning rate*

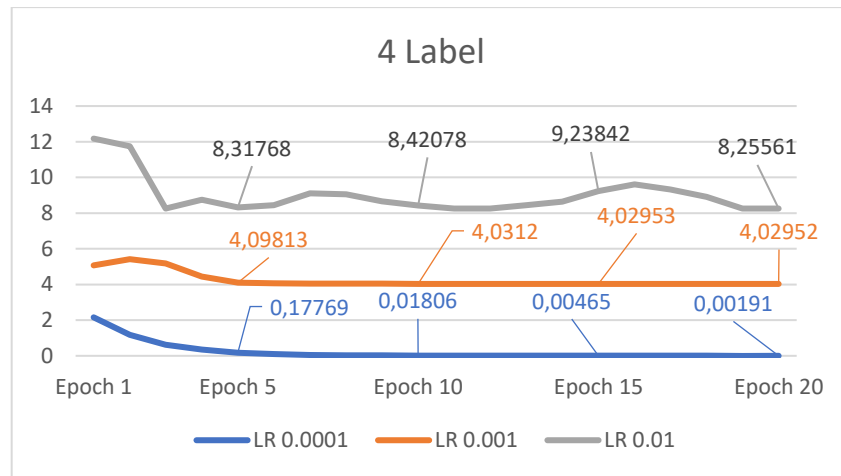
Pengujian terhadap *learning rate* di lakukan dengan 2 label dan 4 label berikut adalah hasil dari masing-masing kategori.

### a. 2 Label



Grafik 5.4 Skenario pada 2 label

## b. 4 Label



Grafik 5.5 Skenario pada 4 label

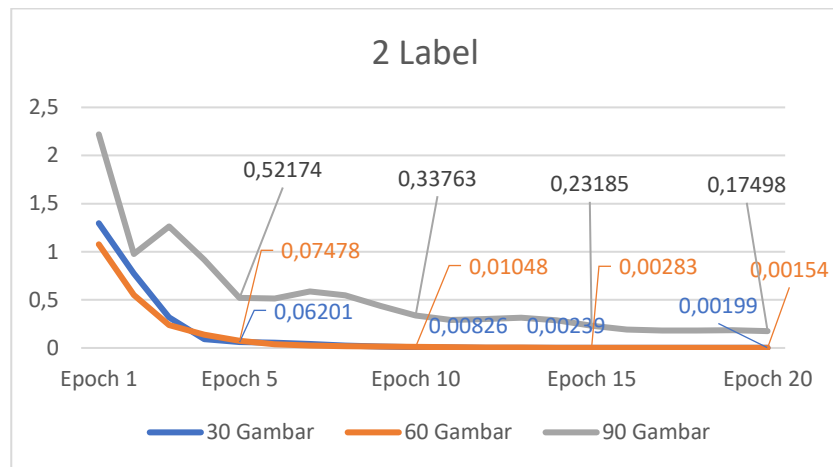
Dari data pengujian di atas dapat di ketahui semakin besar nilai *learning rate* maka akan semakin besar *loss function*, sehingga menggunakan *learning rate* terlalu besar tidak baik bagi model klasifikasi menggunakan MobileNet.

Grafik 5.6 Pengaruh *learning rate* terhadap *loss function*

## 3. Pengujian terhadap jumlah gambar tiap label

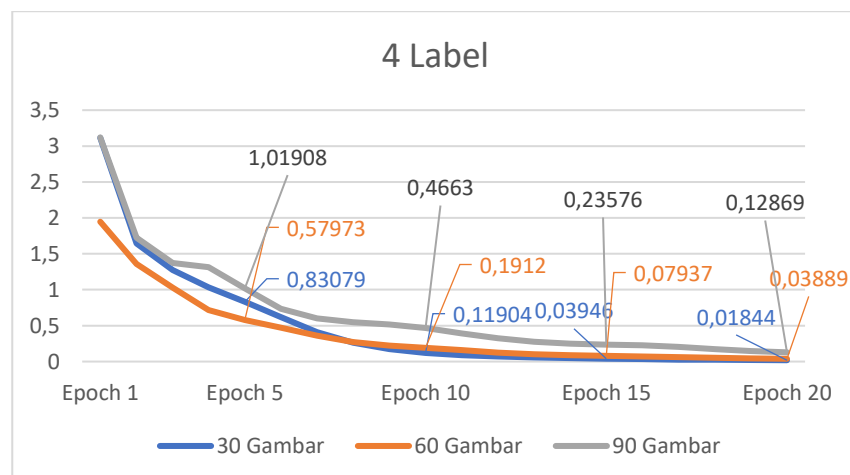
Pengujian terhadap jumlah gambar di lakukan dengan 2 label dan 4 label berikut adalah hasil dari masing-masing kategori.

## a. 2 Label



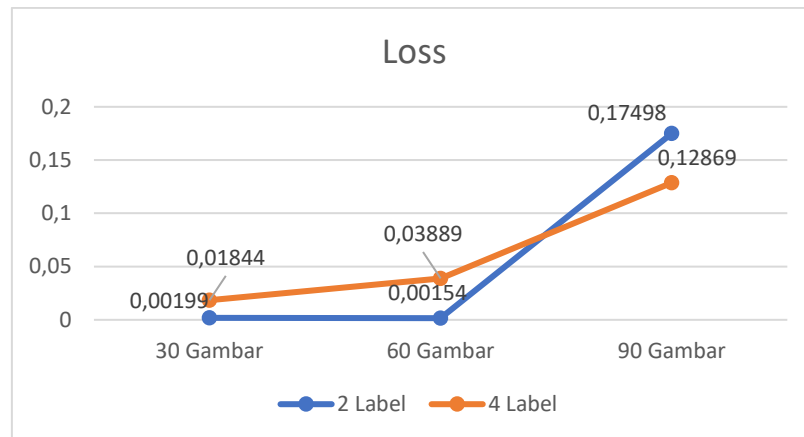
Grafik 5.7 Skenario pada 2 label

## b. 4 Label



Grafik 5.8 Skenario pada 4 label

Dari data di atas semakin banyak gambar dapat menaikkan *loss function* dan membuat angka error semakin besar pada model MobileNet.



Grafik 5.9 Pengaruh banyak gambar terhadap *loss function*

Dari skenario di atas dapat diketahui *Learning rate* menjadi faktor paling besar dalam melakukan training pada *client-side machine learning*, karena dapat mempengaruhi *loss function* secara besar sehingga model akan memiliki banyak *error* dalam melakukan klasifikasi jika *learning rate* tidak di konfigurasi secara tepat. Dari skenario di atas di dapatkan konfigurasi yang memiliki error terendah adalah *epoch* 20, *learning rate* 0.0001, dan banyak *dataset* 60 gambar.

### 5.3 Perbandingan Client-Side dan Server-side Machine Learning

Sebelum perbandingan di mulai, peneliti harus menentukan parameter yang seimbang. Antara lain keduanya menggunakan Pre-Trained Model MobileNet, pengaturan *neural network* yang sama, dan *dataset* yang sama. Perbandingan ini hanya di lakukan di dalam console pada server-side, dan console-log dalam client-side tanpa ada *user interface*. Pertama peneliti harus membuat model klasifikasi gambar menggunakan server-side, langkah-langkahnya adalah sebagai berikut.

Klasifikasi yang di lakukan terbagi menjadi 3 kategori, yaitu 2 label, 3 label, dan 4 label yang semuanya di ambil dari webcam laptop peneliti. Dataset menggunakan 20 gambar pada tiap labelnya dan *training* data menggunakan pengaturan yang di dapatkan dari skenario sebelumnya yaitu *epoch* 20, *learning rate* 0.0001, dan *batch size* 16. Pembuatan klasifikasi dengan server-side di lakukan pada Jupyter Notebook pada laptop yang sama. Berikut adalah sample gambar yang digunakan dalam dataset pada perbandingan.





Gambar 5.5 Contoh Dataset

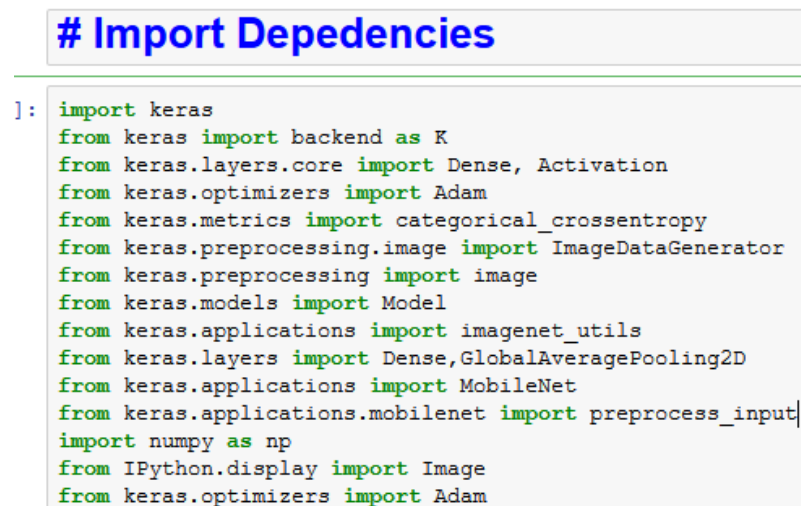
Dataset diambil menggunakan webcam laptop peneliti dan di ambil dalam keadaan ruangan dengan pencahayaan yang baik. Pembuatan klasifikasi yang di lakukan di Server-Side membutuhkan beberapa dependensi yang harus di download dan siapkan untuk mengolah data yang di berikan. Pada percobaan di bawah ini di butuhkan **Jupyter Notebook, Tensorflow, Keras, Python 3.6 dan Pillow**.

✓ keras	○ Deep learning library for theano and tensorflow
✓ keras-applications	○ Applications module of the keras deep learning library.
✓ keras-base	○
✓ tensorflow	○ Tensorflow is a machine learning library.
✓ tensorflow-base	○ Tensorflow is a machine learning library, base package contains only tensorflow.
✓ pillow	○ Pillow is the friendly pil fork by alex clark and contributors

Gambar 5.6 Dependensi yang di perlukan dalam Server-Side

Masing adalah Python versi 3.6 yang digunakan sebagai bahasa pemrograman dengan versi 3.6 yang cocok dengan Tensorflow dan Keras, Jupyter Notebook sebagai alat untuk mengolah data dan program, Tensorflow dan Keras sebagai library Machine Learning yang memudahkan peneliti dalam membuat model klasifikasi, Pillow digunakan sebagai library python yang memudahkan untuk melakukan *import* dan manipulasi berbagai jenis tipe gambar untuk di gunakan sebagai dataset dalam penelitian. Setelah semua dependensi dipenuhi maka akan di lakukan pemrograman sebagai berikut:

1. *Import Dependencies*



```

# Import Dependencies

]: import keras
   from keras import backend as K
   from keras.layers.core import Dense, Activation
   from keras.optimizers import Adam
   from keras.metrics import categorical_crossentropy
   from keras.preprocessing.image import ImageDataGenerator
   from keras.preprocessing import image
   from keras.models import Model
   from keras.applications import imagenet_utils
   from keras.layers import Dense, GlobalAveragePooling2D
   from keras.applications import MobileNet
   from keras.applications.mobilenet import preprocess_input
   import numpy as np
   from IPython.display import Image
   from keras.optimizers import Adam
  
```

Gambar 5.6 *import dependencies*

Sistem menggunakan keras sebagai dasar fungsi dari machine learning. Gambar 5.6 merupakan alat-alat yang di butuhkan saat menggunakan server-side machine learning untuk melakukan pembuatan model klasifikasi.

## 2. Import MobileNet

```
## Import MobileNet and insert Layer - Activation

In [106]: base_model=MobileNet(weights='imagenet',include_top=False) #imports the mobilenet model
x=base_model.output
x=GlobalAveragePooling2D()(x)
x=Dense(1024,activation='relu')(x) #dense layer 2
x=Dense(512,activation='relu')(x) #dense layer 3
preds=Dense(2,activation='softmax')(x) #final layer with softmax activation

In [107]: model=Model(inputs=base_model.input,outputs=preds)

Arsitektur MobileNet

In [108]: model.summary()
```

Layer (type)	Output Shape	Param #
input_4 (InputLayer)	(None, None, None, 3)	0
conv1_pad (ZeroPadding2D)	(None, None, None, 3)	0
conv1 (Conv2D)	(None, None, None, 32)	864
conv1_bn (BatchNormalization)	(None, None, None, 32)	128
conv1_relu (ReLU)	(None, None, None, 32)	0
conv_dw_1 (DepthwiseConv2D)	(None, None, None, 32)	288
conv_dw_1_bn (BatchNormaliza)	(None, None, None, 32)	128
conv_dw_1_relu (ReLU)	(None, None, None, 32)	0

Gambar 5.7 import MobileNet

Dilakukan download dan import pre-trained model MobileNet. Dalam proses ini di ambil *weight* / beban dari model dan untuk layer *neural network* di rancang agar sesuai dengan kebutuhan klasifikasi yaitu 2 *dense layer* dan 1 *classification layer*.

## 3. Mempersiapkan Model

### ## Setting Layer

```
|: for layer in model.layers:
    layer.trainable=False
    # or if we want to set the first 20 layers of the network
    for layer in model.layers[:20]:
        layer.trainable=False
    for layer in model.layers[20:]:
        layer.trainable=True
```

```

]: train_datagen=ImageDataGenerator(preprocessing_function=preprocess_input) #included in o

train_generator=train_datagen.flow_from_directory('c:/Users/farvn/Downloads/Dataset',
                                                target_size=(224,224),
                                                color_mode='rgb',
                                                batch_size=16,
                                                class_mode='categorical',
                                                shuffle=True)

Found 44 images belonging to 2 classes.

## Compile Model

]: model.compile(optimizer='Adam',loss='categorical_crossentropy',metrics=['accuracy'])
# optimizer = adam
# loss function = categorical cross entropy
# evaluation metric = accuracy

]: step_size_train=train_generator.n//train_generator.batch_size

```

Gambar 5.8 menyiapkan model dan pengaturan

Pada tahap ini dilakukan pengaturan terhadap pre-trained model dan pengambilan direktori dataset yang akan di gunakan, sehingga pengaturan neural network yang terdiri dari dimensi gambar, batch size, optimasi, dan loss function yang akan di terapkan pada model klasifikasi.

#### 4. Training Model

```

: step_size_train=train_generator.n//train_generator.batch_size

## Train Model

: process = model.fit_generator(generator=train_generator,
                                steps_per_epoch=step_size_train,
                                epochs=20)

```

Gambar 5.9 Melakukan training pada model.

Model di compile dan di lakukan training menggunakan pengaturan sebelumnya. Pada model ini di lakukan 20 epoch. Setelah di lakukan training di dapatkan hasil per epoch sebagai berikut.

```

Epoch 1/20
2/2 [=====] - 5s 2s/step - loss: 2.2911e-07 - acc: 1.0000
Epoch 2/20
2/2 [=====] - 5s 3s/step - loss: 1.1921e-07 - acc: 1.0000
Epoch 3/20
2/2 [=====] - 3s 2s/step - loss: 3.9328e-05 - acc: 1.0000
Epoch 4/20
2/2 [=====] - 4s 2s/step - loss: 9.8907e-07 - acc: 1.0000
Epoch 5/20
2/2 [=====] - 4s 2s/step - loss: 1.1921e-07 - acc: 1.0000
Epoch 6/20
2/2 [=====] - 4s 2s/step - loss: 9.4364e-05 - acc: 1.0000
Epoch 7/20
2/2 [=====] - 4s 2s/step - loss: 1.3039e-07 - acc: 1.0000
Epoch 8/20
2/2 [=====] - 3s 2s/step - loss: 2.1341e-07 - acc: 1.0000
Epoch 9/20
2/2 [=====] - 3s 2s/step - loss: 1.2701e-07 - acc: 1.0000
Epoch 10/20
2/2 [=====] - 4s 2s/step - loss: 1.9912e-06 - acc: 1.0000
Epoch 11/20
2/2 [=====] - 4s 2s/step - loss: 6.5193e-07 - acc: 1.0000
Epoch 12/20
2/2 [=====] - 4s 2s/step - loss: 4.8346e-07 - acc: 1.0000
Epoch 13/20
2/2 [=====] - 4s 2s/step - loss: 1.1921e-07 - acc: 1.0000
Epoch 14/20
2/2 [=====] - 4s 2s/step - loss: 1.9579e-06 - acc: 1.0000
Epoch 15/20
2/2 [=====] - 3s 2s/step - loss: 8.5714e-06 - acc: 1.0000
Epoch 16/20
2/2 [=====] - 4s 2s/step - loss: 1.3597e-07 - acc: 1.0000
Epoch 17/20
2/2 [=====] - 3s 2s/step - loss: 1.1921e-07 - acc: 1.0000
Epoch 18/20
2/2 [=====] - 4s 2s/step - loss: 1.5824e-07 - acc: 1.0000
Epoch 19/20
2/2 [=====] - 4s 2s/step - loss: 3.0175e-07 - acc: 1.0000
Epoch 20/20
2/2 [=====] - 4s 2s/step - loss: 2.6586e-07 - acc: 1.0000

```

Gambar 5.10 Hasil *training* per epoch yang di hasilkan Server-Side

Setelah model berhasil di latih dan menghasilkan akurasi latih dan loss functionnya, model harus di conver jika ingin melakukan prediksi secara dinamik menggunakan user interface dan prediksi secara langsung. Namun pada penilitian ini hanya di lakukan di notebook saja karena hanya di butuhkan untuk perbandingan loss dan waktu training.

Setelah di lakukan training model pada 3 kategori yang masing-masing di lakukan 5 kali pengulangan pada setiap kategorinya menggunakan pengaturan dan laptop yang sama, di dapatkan hasil perbandingan sebagai berikut.

Tabel 5.1 Hasil Perbandingan *loss function***Nilai Loss**

Pengujian	4 Label 20 epoch		Pengujian	3 Label 20 epoch		Pengujian	2 Label 20 epoch	
	Server	Client		Server	Client		Server	Client
1	0,0637	0,00598	1	0,0156	0,00053	1	1,3E-07	0,00345
2	0,0245	0,00912	2	0,000128	0,0014	2	0,0259	0,00001
3	0,1091	0,00232	3	0,000117	0,0006	3	1,3E-07	0,00074
4	0,000136	0,0031	4	0	0,00092	4	1,2E-07	0,00004
5	0,000153	0,0076	5	0,0023	0,00061	5	2,6E-07	0,00003
Rata-rata	<b>0,039518</b>	<b>0,005624</b>	Rata-rata	<b>0,003629</b>	<b>0,000812</b>	Rata-rata	<b>0,00518</b>	<b>0,000854</b>

Tabel 5.2 Hasil Perbandinagn Waktu (detik)

**Training Time (s)**

Pengujian	4 Label 20 epoch		Pengujian	3 Label 20 epoch		Pengujian	2 Label 20 epoch	
	Server	Client		Server	Client		Server	Client
1	191	2,28	1	176	2,03	1	84	2,63
2	186	1,62	2	164	1,65	2	80	2,33
3	188	1,77	3	163	1,44	3	80	1,77
4	183	1,62	4	160	1,42	4	88	1,96
5	185	1,65	5	160	1,53	5	77	1,17
Rata-rata	<b>186,6</b>	<b>1,788</b>	Rata-rata	<b>164,6</b>	<b>1,614</b>	Rata-rata	<b>81,8</b>	<b>1,972</b>

Dapat di simpulkan dari percobaan yang di lakukan peneliti dalam 3 skenario dan 5 kali percobaan tiap skenarionya, rata-rata *loss funcrtion* yang di hasilkan oleh *Server-Side Machine Learning* adalah 0,00243 sementara *Client-Side* 0,016109, dan rata-rata kecepatan training pada *Server-Side* adalah 144 detik dan *Clien-Side* 1.79 detik. Dengan demikian *Server-Side Machine Learning* lebih unggul sedikit pada hasil error yang di hasilkan sedangkan pada waktu training *Server-Side* jauh lebih unggul dari *Server-Side*.

## **BAB 6**

### **KESIMPULAN DAN SARAN**

#### **6.1 Kesimpulan**

Hasil penelitian aplikasi klasifikasi gambar menggunakan MobileNet dan Client-side machine learning berbasis *website* ini memiliki poin-poin kesimpulan sebagai berikut:

- a. Aplikasi berhasil dikembangkan dengan menerapkan metode pengembangan sistem *Rapid Application Development* dan menggunakan algoritma CNN dari Pre-Trained Model MobileNet.
- b. Aplikasi menggunakan metode *client-side* machine learning yang memungkinkan pembuatan model klasifikasi di dalam web browser tanpa harus mendownload *dependencie* dan melakukan *convert* model jika di lakukan di *server-side*, dan karena hal ini menggunakan *Client-Side Machine Learning* pengguna dapat melakukan klasifikasi secara *dynamic*.
- c. Client-side machine learning melakukan training model jauh lebih cepat dari server-side dengan selisih waktu 142 detik lebih cepat atau hanya memerlukan 1,2% waktu yang di lakukan di Server.
- d. *Server-Side* lebih unggul dalam nilai *loss function* / nilai *error* yang di keluarkan dalam perbandingan yang di lakukan dengan selisih rata-rata 0,013679.
- e. Aplikasi mengeluarkan nilai *loss function* / nilai *error* yang dapat di optimasi dengan menyesuaikan jumlah label, epoch, learning rate, dan jumlah gambar yang di masukan per label. Dalam aplikasi ini learning rate sangat berpengaruh dengan loss function pada model sehingga nilai yang paling tepat adalah 0.0001, dan jumlah epoch berpengaruh jika jumlah label yang di klasifikasi lebih dari 2.

## 6.2 Saran

Dalam penelitian selanjutnya, penulis merasa perlu dilakukan beberapa perbaikan dan pengembangan aplikasi ini dalam beberapa aspek sebagai berikut:

- a. Penelitian berikutnya dapat mengembangkan versi mobile dalam aplikasi dan menerapkan metode *Web Progressive App* agar bisa di jalankan offline.
- b. Dapat mengeluarkan visualisasi gambar saat di lakukan ekstraksi gambar.
- c. Diharapkan penelitian selanjutnya dapat menerima gambar melalui *file explorer* sehingga user tidak harus menggunakan kamera dalam mengumpulkan dataset.
- d. Penelitian selanjutnya dapat bereksperimen dengan beberapa model pretrained model lainnya untuk dapat menemukan model paling baik akurasi dalam klasifikasi gambar pada *client-side machine learning*.



## DAFTAR PUSTAKA

- Agarap, A. F. (2018). *Deep Learning using Rectified Linear Units (ReLU)*. (1), 2–8. <https://doi.org/10.1249/01.MSS.0000031317.33690.78>
- Arfian. (2018). *Implementasi Convolutional Neural Network Terhadap Transportasi Tradisional Menggunakan Keras (Studi Kasus : Data Citra Transportasi Tradisional Andong, Becak dan Pedati)* (Universitas Islam Indonesia). Retrieved from <http://e-journal.uajy.ac.id/14649/1/JURNAL.pdf>
- Aurélien, G. (2017). *Hands-On Machine Learning with Scikit-Learn*. <https://doi.org/10.3389/fninf.2014.00014>
- Budi, R. (2011a). *Belajar Otodidak Pemrograman Web dengan PHP+Oracle*. Bandung: Informatika.
- Budi, R. (2011b). *Belajar Pemrograman Web*. Bandung: Modula.
- Carol Britton, J. D. (2001). *Object-oriented System Development*. McGraw-Hill Publishing Co.
- ConvNetJS. (2018). Deep Learning in Your Browser. Retrieved March 11, 2019, from <https://cs.stanford.edu/people/karpathy/convnetjs/>
- Dangeti, P. (2017). *Statistics for Machine Learning* (D. Pawar, Ed.). MUMBAI: Packt Publishing Ltd.
- Danukusumo, K. P. (2017). *Implementasi Deep Learning Menggunakan Convolutional Neural Network Untuk Klasifikasi Citra Candi Berbasis GPU*. (Universitas Atma Jaya Yogyakarta). Retrieved from <http://e-journal.uajy.ac.id/12425/>
- Dipanjan, S., Bali, R., & Ghosh, T. (2018). *Hands-On Transfer Learning with Python* (1st ed.; Sunith Shetty, Ed.). Birmingham: Packt Publishing Ltd.
- Flasiński, M. (2016). *Introduction to Artificial Intelligence*. <https://doi.org/10.1007/978-3-319-40022-8>
- Gartner, & Panetta, K. (2018). Gartner Top 10 Strategic Technology Trends for 2019. Retrieved March 5, 2019, from

- <https://www.gartner.com/smarterwithgartner/gartner-top-10-strategic-technology-trends-for-2019/>
- GitHub. (2018). The state of the octoverse 2018. Retrieved March 5, 2019, from <https://octoverse.github.com/>
- Goodfellow, I., Bengio, Y., & Courville, A. C. (2016). *Deep learning* (Adaptive C). Retrieved from <https://lccn.loc.gov/2016022992>
- Hermawan, A. (2006). *Jaringan Syaraf Tiruan Teori dan Aplikasi*. Yogyakarta: C.V Andi Offset.
- Herron, D. (2018). *Node.js Web Development Fourth Edition* (4th ed.; A. Banerjee, Ed.). Birmingham: Packt Publishing Ltd.
- Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., ... Adam, H. (2017). *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*. [https://doi.org/10.1016/S1507-1367\(10\)60022-3](https://doi.org/10.1016/S1507-1367(10)60022-3)
- IdHost. (2018). Pengertian Website Secara Lengkap. Retrieved November 12, 2018, from <https://idwebhost.com/blog/pengertian-website-secara-lengkap/>
- Kendall, K. E., & Kendall, J. E. (2010). *System Analysis And Design* (8th ed.; S. Yagan, Ed.). Pearson/Prentice Hal.
- Khalilulah, H. A. (2016). *Implementasi Speech Recognition pada Aplikasi Penerjemah Idiom Bahasa Inggris ke Bahasa Indonesia Berbasis Android*. 1–6.
- Liang, Y., Tu, Z., Huang, L., & Lin, J. (2018). *CNNs for NLP in the Browser : Client-Side Deployment and Visualization Opportunities*. 61–65.
- Ma, Y., Xiang, D., Zheng, S., Tian, D., & Liu, X. (2019). *Moving Deep Learning into Web Browser: How Far Can We Go?* 1–17. Retrieved from <http://arxiv.org/abs/1901.09388>
- Michele, A., Colin, V., & Santika, D. D. (2019). MobileNet Convolutional Neural Networks and Support Vector Machines for Palmprint Recognition. *Procedia Computer Science*, 157, 110–117. <https://doi.org/10.1016/j.procs.2019.08.147>

NYU. (2018). ML5.js.

Pressman, R. S., & Maxim, B. R. (2015). *Software Engineering: A Practitioners Approach* (8th ed.; V. Bradshaw, Ed.). New York: McGraw-Hill Education.

Richard E. Neapolitan, X. J. (2018). Artificial Intelligence with an introduction to machine learning. In *Taylor & Francis Group, LLC*.

Rismiyati. (2016). *Implementasi Convolutional Neural Network Untuk Sortasi Mutu Salak Ekspor Berbasis Citra Digital*. Universitas Gajah Mada`.

Rungta, B. K. (2016). *Learn NodeJS in 1 Day*.

Sathya, R., & Abraham, A. (2013). Comparison of Supervised and Unsupervised Learning Algorithms for Pattern Classification. *International Journal of Advanced Research in Artificial Intelligence*, 2(2), 34–38. <https://doi.org/10.14569/ijarai.2013.020206>

Smilkov, D., Thorat, N., Assogba, Y., Yuan, A., Kreeger, N., Yu, P., ... Wattenberg, M. (2019). *TensorFlow.js: Machine Learning for the Web and Beyond*. Retrieved from <http://arxiv.org/abs/1901.05350>

Stack Overflow. (2018). Stack overflow developer survey 2018. Retrieved March 3, 2019, from <https://insights.stackoverflow.com/survey/2018/>

Sudarsono, A. (2016). Jaringan Syaraf Tiruan Untuk Memprediksi Laju Pertumbuhan Penduduk Menggunakan Metode Bacpropagation (Studi Kasus Di Kota Bengkulu). *Teknik Informatika*, 12(1), 61–69.

Uniqtech. (2018). Understand the Softmax Function in Minutes. Retrieved March 25, 2019, from Medium.com website: <https://medium.com/data-science-bootcamp/understand-the-softmax-function-in-minutes-f3a59641e86d>

Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). How transferable are features in deep neural networks? *Advances in Neural Information Processing Systems* 27, 2, 3320–3328. <https://doi.org/10.1002/celc.201500375>