**RANCANG BANGUN SISTEM PENGENALAN WAJAH BERBASIS *RESIDUAL NETWORK***

Tugas Akhir

Untuk memenuhi sebagian persyaratan

mencapai derajat Sarjana S-1 Jurusan Teknik Elektro



**Disusun Oleh:**

**Muhammad Sya’roni Mujahidin**

**F1B014065**

**JURUSAN TEKNIK ELEKTRO**

**FAKULTAS TEKNIK**

**UNIVERSITAS MATARAM**

**2019**

# Tugas Akhir

**`**

**RANCANG BANGUN SISTEM PENGENALAN WAJAH BERBASIS *RESIDUAL NETWORK***

Oleh:

**Muhammad Sya’roni Mujahidin**

**F1B 014 065**

Telah diperiksa dan disetujui oleh Tim Pembimbing:

|  |  |
| --- | --- |
| 1. Pembingbing Utama   Dr. Misbahuddin, ST., MT.  NIP. 19681005 199703 1 001 | Tanggal: Oktober 2019 |
| 1. Pembimbing Pendamping   Bulkis Kanata, ST., MT.  NIP: 19730723 199903 2 001 | Tanggal: Oktober 2019 |

Mengetahui

Ketua Jurusan Teknik Elektro

Fakultas Teknik

Universitas Mataram

Muhamad Syamsu Iqbal, ST., MT., Ph.D.

NIP. 19720222 199903 1 002

# Tugas Akhir

**RANCANG BANGUN SISTEM PENGENALAN WAJAH BERBASIS *RESIDUAL NETWORK***

Oleh:

**Muhammad Sya’roni Mujahidin**

**F1B 114 030**

Telah dipertahankan di depan Dewan Penguji

Pada tanggal 30 Oktober 2019

dan dinyatakan telah memenuhi syarat mencapai derajat Sarjana S-1

Jurusan Teknik Elektro

Susunan Tim Penguji

1. Penguji 1

Tanggal:

I Made Budi Suksmadana, ST., MT.

NIP: 19710426 199903 1 002

1. Penguji 2

Tanggal:

A. Sjamsjiar Rachman, ST., MT.

NIP: 19711124 199903 1 004

1. Penguji 3

Tanggal:

Lalu A. Syamsul Irfan Akbar, ST., MT.

NIP: 19830310 200912 1 004

Mengetahui,

Dekan Fakultas Teknik

Universitas Mataram

Akmaluddin, ST., MSc(Eng)., Ph.D.

NIP: 19681231 199412 1 001

# SURAT PERNYATAAN KEASLIAN

Dengan ini saya menyatakan bahwa dalam Tugas Akhir ini tidak terdapat karya yang pernah diajukan untuk memperoleh Gelar Kesarjanaan di suatu Perguruan Tinggi, dan sepanjang pengetahuan saya, tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis diacu dalam naskah ini dan disebutkan dalam daftar acuan.

Demikian surat pernyataan ini saya buat tanpa tekanan dari pihak manapun dan dengan kesadaran penuh terhadap tanggung jawab dan konsekuensi serta menyatakan bersedia menerima sanksi terhadap pelanggaran dari pernyataan tersebut.

|  |
| --- |
| Mataram, 7 Oktober 2019  **Muhammad Sya’roni Mujahidin**  **F1B 014 065** |

# KATA PENGANTAR

Puji Syukur peneliti haturkan kehadirat Allah SWT atas segenap kasih sayang dan anugerah yang telah diberikan kepada penulis dan selawat dan salam semoga senantiasa tercurahkan kepada Baginda Rasulullah SAW sebagai soko guru peradaban, selanjutnya peneliti mengucapkan banyak terima kasih kepada semua pihak yang ikut membantu sehingga dapat menyelesaikan tugas akhir dengan judul “**Rancang Bangun Sistem Pengenalan Wajah Berbasis *Residual Network*”.**

Tugas akhir ini dikhususkan untuk menentukan peminatan akademik bagi siswa SMA dengan mempertimbangkan bakat, kemampuan dan minat siswa, namun tidak menutup kemungkinan untuk dikembangkan ke arah yang lebih luas lagi dengan menambah parameter kriteria peminatan akademik agar lebih baik lagi.

Kekurangan dan kealpaan adalah hal lumrah yang terdapat dalam diri segenap manusia, karenanya peneliti mohon maaf jika terdapat hal-hal tersebut tadi di dalam penyusunan tugas akhir ini. Saran dan kritik sangat diharapkan demi pembelajaran dan membangun semangat peneliti untuk dapat menjadi lebih baik lagi.

Akhirnya peneliti ucapkan terima kasih kepada semua pihak atas bantuanya selama menyelesaikan tugas akhir ini. Semoga tugas akhir ini dapat bermanfaat bagi khalayak banyak.

Mataram, 7 Oktober 2019

Peneliti

# UCAPAN TERIMA KASIH

Tugas Akhir ini dapat diselesaikan berkat bimbingan dan dukungan ilmiah maupun materil dari berbagai pihak, oleh karena itu pada kesempatan ini penulis menyampaikan ucapan terima kasih yang setulus-tulusnya kepada :

1. Allah S.W.T. yang telah memberikan nikmat kesehatan dan kelancaran sehingga tugas akhir ini dapat diselesaikan
2. Bapak Akmaluddin, ST., M.Sc (Eng)., Ph.D., Selaku Dekan Fakultas Teknik Unram.
3. Bapak Muhamad Syamsu Iqbal, S.T., M.T., Ph.D., Selaku Ketua Jurusan Teknik Elektro Fakultas Teknik Unram sekaligus sebagai dosen penguji.
4. Bapak Dr. Misbahuddin, S.T., M.T, selaku dosen pembimbing pertama yang telah memberikan bimbingan dan arahan kepada penulis selama penyusunan Tugas Akhir ini, sehingga dapat terselesaikan dengan baik.
5. Ibu Bulkis Kananta, S.T., M.T, selaku dosen pembimbing pendamping yang telah memberikan bimbingan dan arahan selama menyusun Tugas Akhir ini, sehingga dapat terselesaikan dengan baik.
6. Bapak I Made Budi Suksmadana, S.T., M.T, , Bapak Sjamsiar Rahman, S.T., M.T, dan Bapak Lalu A. Syamsul Irfan Akbar, S.T., M.T., selaku dosen penguji yang telah memberikan masukan-masukan selama proses penyusunan Tugas Akhir ini.
7. Kedua Orang Tua penulis Bapak Abd Hakim S.Pdi dan Ibu Suryani serta Saudari tersayang Annisa Dwi Rahmawati yang selalu memberikan dukungan moril maupun materil, motivasi dan Do’a yang tiada henti-hentinya selama penulis menempuh dan sampai menyelesaikan pendidikan di Universitas Mataram. Terima kasih yang sebesar-besarnya.
8. Rizki Amalia Nuraini yang selalu memberikan support dan semangat serta doa ketika menyelesaikan Tugas Akhir ini.
9. Sahabat Lumbung Inovasi Bang fandi, Toufan, Zul, Bang Hayi, Bang Lian, Darmawan, Jihad, Anwar, Aan dan Januarman yang selalu menemani dalam suka maupun duka selama menyusun Tugas Akhir ini.
10. Sahabat dan saudara penulis khususnya Teknik Elektro angkatan 2014.
11. Keluarga besar Jurusan Teknik Elektro Fakultas Teknik Universitas Mataram.

DAFTAR ISI

[Tugas Akhir ii](#_Toc23125777)

[Tugas Akhir iii](#_Toc23125778)

[SURAT PERNYATAAN KEASLIAN iv](#_Toc23125779)

[KATA PENGANTAR v](#_Toc23125780)

[UCAPAN TERIMA KASIH vi](#_Toc23125781)

[DAFTAR GAMBAR ix](#_Toc23125782)

[DAFTAR TABEL ix](#_Toc23125783)

[DAFTAR LAMPIRAN ix](#_Toc23125784)

[BAB I PENDAHULUAN 1](#_Toc23125785)

[1.1. Latar Belakang 1](#_Toc23125786)

[1.2. Rumusan Masalah 2](#_Toc23125787)

[1.3. Batasan Masalah 2](#_Toc23125788)

[6.1. Tujuan Penelitian 2](#_Toc23125789)

[6.2. Manfaat Penelitian 2](#_Toc23125790)

[1.6 Sistematika Penulisan 3](#_Toc23125791)

[BAB II TINJAUAN PUSTAKA 4](#_Toc23125792)

[2.1. Penelitian yang terkait pengenalan wajah 4](#_Toc23125793)

[2.2. Landasan Teori 5](#_Toc23125794)

[2.2.1. Convolulutional Neural Network (*CNN*) 5](#_Toc23125795)

[2.2.2. *Python* 11](#_Toc23125796)

[2.2.3. *Deep Learning* 13](#_Toc23125797)

[2.2.4. *Training* 18](#_Toc23125798)

[2.2.5. *Learning Rate* 18](#_Toc23125799)

[2.2.6. *AI Framewk* 19](#_Toc23125800)

[BAB III METODOLOGI PENELITIAN 21](#_Toc23125801)

[3.1. Tahapan Penelitian 21](#_Toc23125802)

[3.2. Studi Literatur 22](#_Toc23125803)

[3.3. Analisa Kebutuhan 22](#_Toc23125804)

[3.3.1. Kebutuhan Perangkat *Keras* 22](#_Toc23125805)

[3.3.2. Kebutuhan Perangkat Lunak 22](#_Toc23125806)

[3.4. Perancangan Sistem 23](#_Toc23125807)

[3.4.1. Perancangan *Resnet* 24](#_Toc23125808)

[3.5. Implementasi 26](#_Toc23125809)

[3.5.1. Perancangan Sistem 27](#_Toc23125810)

[3.6. Pengujian Sistem 28](#_Toc23125811)

[BAB IV METODOLOGI PENELITIAN 29](#_Toc23125812)

[4.1. Pelatihan 29](#_Toc23125813)

[4.1.1. Lanju Pembelajaran Terhadap Waktu Komputasi 30](#_Toc23125814)

[4.1.2. Laju Pembelajaran Terhadap Kesalahan (%) **Error! Bookmark not defined.**](#_Toc23125815)

[4.1.3. Laju Pembelajaran Terhadap Akurasi (%) 31](#_Toc23125816)

[4.2. Validasi (*Testing)* 32](#_Toc23125817)

[BAB V KESIMPULAN DAN SARAN 35](#_Toc23125818)

[5.1. Kesimpulan 36](#_Toc23125819)

[5.2. Saran 36](#_Toc23125820)

[DAFTAR PUSTAKA 37](#_Toc23125821)

[LAMPIRAN 41](#_Toc23125822)

# DAFTAR GAMBAR

[Gambar 2.1 Proses pemintasan blok](#_Toc513303971) 6

[Gambar 2.2 Pemintasan *Resnet*](#_Toc513303972) 7

[Gambar 2.3 layer *resnet*](#_Toc513303973) 7

[Gambar 2.4 konvolusi](#_Toc513303974) 8

[Gambar 2.5 *Deep Learning*](#_Toc513303975) 9

[Gambar 2.6 Image Pre-processing](#_Toc513303976) 11

[Gambar 2.7 *Max Pool*ing](#_Toc513303977) 14

[Gambar 2.8 Flatten](#_Toc513303978) 14

[Gambar 3.1 Tahapan Penelitian](#_Toc513303979) 16

[Gambar 3.2 Perancangan Sistem](#_Toc513303980) 18

[Gambar 3.3 Blok *Resnet*](#_Toc513303981) 19

[Gambar 3.4 Skip Connection *resnet*](#_Toc513303982) 20

[Gambar 3.5 Lapisan *Resnet*](#_Toc513303983) 20

[Gambar 3.6 Flow Chart *Training*](#_Toc513303984) 22

[Gambar 4.1. Diagram laju pembelajaran terhadap waktu komputasil](#_Toc513303985) 25

[Gambar 4.2. Diagram laju pembelajaran terhadap kesalahan](#_Toc513303986) 26

[Gambar 4.3. Diagram Laju pembelajaran terhadap akurasi](#_Toc513303987) 27

# DAFTAR TABEL

[Tabel 4.1 Hasil pengujian *Algoritma* pada folder *training*](#_Toc527920439) 24

[Tabel 4.2. Validation (Testing)](#_Toc527920440) 28

# DAFTAR LAMPIRAN

[Tabel *Learning Rate*](#_Toc527920439) 34

[*Source Code*](#_Toc527920440) 40

**ABSTRAK**

*Deep Learning* adalah sebuah bidang keilmuan baru dalam bidang *Machine Learning* yang akhir-akhir ini berkembang karena perkembangan teknologi *GPU accelaration*. *Deep Learning* memiliki kemampuan yang sangat baik dalam visi komputer. Salah satunya adalah pada kasus pengenalan objek pada citra. Dengan mengimplementasikan salah satu metode *Machine Learning* yang dapat digunakan untuk pengenalan citra objek yaitu *CNN*. *Convolutional Neural Networks* (*CNN*) merupakan*) yang saat ini diklaim sebagai model terbaik* yang saat ini diklaim sebagai model terbaik untuk memecahkan masalah *object recognition* dan *detection*. Dalam implementasinya, *Algoritma* *CNN* menggunakan input data berupa sampel *training* menggunakan gambar yang diberi label berupa teks. Pada penelitian ini peneliti merancang sistem pengenalan wajah dengan menggunalan *Algoritma CNN ResNet*. Hasil dari penelitian didapatkan hasil untuk meningkatkan akurasi pengenalan wajah yaitu 92% .

Kata Kunci: *Residual Network, CNN,* Pengenalan Wajah.

***ABSTRACT***

*Deep Learning is a new scientific field in the field of Machine Learning which has recently developed due to the development of GPU acceleration technology. Deep Learning has excellent skills in computer vision. One of them is when promoting attractions in the image. By implementing one of the Machine Learning methods that can be used to introduce object images, namely CNN. Convolutional Neural Networks (CNN) is a special case of artificial neural networks (ANN) which is currently claimed as the best model for solving the problem of object recognition and detection. In its implementation, the CNN algorithm uses input data in the form of training samples using images that are labeled containing text. In this study, researchers examined the facial recognition system using the CNN ResNet algorithm. The results of the study obtained results to improve facial recognition that is 92%.*

*Keywords: Residual Network, CNN, Face Recognition.*

# BAB I PENDAHULUAN

## Latar Belakang

Pengenalan wajah adalah salah satu metode *biometerik* yang cukup popular. Wajah lebih sulit untuk ditiru, dimodifikasi, atau dicuri jika dibandingkan dengan kunci atau *password* pada keamanan *non-biometrik*. Pada umumnya, metode *biometric* membutuhkan perangkat khusus untuk mengumpulkan data. Misalnya, *fingerprint* untuk mendeksi sidik jari*, scanner* untuk memindai sebuah bentuk dan *palmprint scanner* untuk mendeksi jari di handphone namun pengguna harusmenyentuh alat tersebut secara fisik untuk mendapatkan data. Pada pengenalan wajah, wajah akan dideteksi secara otomatis tanpa memerlukan sentuhan wajah pada perangkat pendeteksinya

Sistem pengenalan wajah menjadi topik yang sering dipelajari dibidang *computer vision* dalam beberapa dekade ini. Sistem ini telah diaplikasikan dalambeberapa bidang, contohnya pada *smartphone* untuk *facelock,* imigrasi, dan juga di media sosial untuk mengatasi *face tagging*. Pengenalan wajah sendiri terdiri dari tahap deteksi dan klasifikasi. Kedua tahap tersebut begitu cepat dilakukan oleh manusia tetapi butuh waktu yang lama bagi komputer. Kemampuan manusia itulah yang ingin diduplikasi oleh para peneliti dalam beberapa tahun belakangan ini sebagai teknologi biometrik dalam bidang *computer vision* degan tujuan membentuk suatu model untuk pengenalan citra wajah pada komputer.

Penelitian yang pernah dilakukan sebelumnya oleh Abhirawa (2017) merancang sistem pengenalan wajah dengan menerapkan *Convolutional Neural Network (CNN)* dengan membuat beberapa layer *conv2D.* Akan tetapi, penelitian tersebut memiliki akurasi pengenalan wajah yang rendah, selain itu *Convolutional Neural Network (CNN)* yang digunakan hanya membuat beberapa layer saja, sehingga jaringan syaraf tiruan yang dibuat tidak dalam.

Oleh karena itu pada penelitian ini akan dibuat sistem pengenalan wajah dengan menggunakan metode *Convolutional Neural Network (CNN)* dengan mengaplikasikan *Residual network (Resnet)* untuk meningkatkan akurasi pada sistem pengenalan wajah.

## Rumusan Masalah

*Algoritma*  yang ada pada saat ini masih relatif rendah untuk meningkatkan akurasi prediksi pada pengenalan wajah. Hal tersebut mengakibatkan kesalahan dalam prediksi pengenalan wajah masih dapat terjadi. Penelitian sebelumnya dilakukan abirawa merancang sistem pengenalan menggunakan *convolution neural network* hanya mendapatkan hasil pengujian pada data testing dengan hasil akurasi diperoleh yaitu 75,79% . Oleh karena itu poin masalah yang dapat diangkat pada penelitian ini adalah *Algoritma* *Residual Network* yang dapat meningkatkan akurasi pengenalan wajah lebih baik.

## Batasan Masalah

Batasan masalah yang digunakan pada tugas akhir ini adalah sebagai berikut:

1. Meningkatkan akurasi pengenalan wajah
2. Menggunakan dataset Yale Extended B
3. Ukuran gambar 32x32 pixel
4. Menggunakan *library Keras*
5. Format gambar .jpg

## Tujuan Penelitian

Tujuan penelitian ini adalah adalah meningkatkan akurasi pengenalan wajah menggunakan *Residual Netwok (Resnet)*

## Manfaat Penelitian

Dengan adanya rancangan sistem ini, diharapkan dapat memberikan manfaat antara lain :

1. Dapat dijadikan sebagai rancangan dan paten untuk diterapkan di rumah, kantor, maumpun diinstansi pemerintahan untuk meningkatkan keamanan
2. Meningkatkan wawasan mengenai penerapan neural network dan *Machine Learning* dalam kehidupan nyata dalam hal sistem keamanan

## 1.6 Sistematika Penulisan

Laporan perencanaan Tugas Akhir ini dibagi menjadi beberapa bab dan sub-sub bab sebagai berikut :

|  |  |
| --- | --- |
| BAB I | PENDAHULUAN |
|  | Pada bab ini, menguraikan tentang latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat serta sistematika penulisan |
| BAB II | TINJAUAN PUSTAKA DAN LANDASAN TEORI |
|  | Pada bab ini berisi beberapa rujukan penelitian yang berkaitan dengan topik yang dibahas, dan teori-teori dasar yang mendukung topik ini. |
| BAB III | METODE PENELITIAN |
|  | Dalam bab ini, menjelaskan tentang waktu dan tempat penelitian, bahan penelitian serta langkah-langkah penelitian yang akan ditempuh. |
| BAB IV | HASIL DAN PEMBAHASAN |
|  | Pada bab ini juga berisi tentang analisa hasil implementasi *Algoritma*, performa kinerja model dan evaluasi terhadap sistem yang telah dibuat. |
| BAB V | KESIMPULAN DAN SARAN |
|  | Bab ini menjelaskan kesimpulan dari hasil yang didapatkan dalam analisa yang telah dilakukan dan menberikan saran dari penelitian yang telah dilakukan. |

# 

# BAB II TINJAUAN PUSTAKA

## Penelitian yang terkait pengenalan wajah

Penelitian ini menggunakan sumber acuan yang berasal dari jurnal-jurnal penelitian yang membahas tentang Sistem Pengenalan Wajah. Sumber acuan yang digunakan sebagian besar menggunakan metode yang berbeda dalam pengenalan wajah.

Sehman (2015) merancang sistem yang dapat menangkap pencuri yang berada di dalam mobil yang telah terpasang sebuah kamera. Penangkapan dilakukan dengan mengunci dan mematikan mesin mobil ketika wajah tidak dikenali. Metode yang digunakan yaitu *eigenface* dan *haarcascade* untuk melakukan proses pengenalan wajah. Tingkat keamanan pada sistem yang dibuat ini sudah sangat efektif untuk digunakan pada setiap mobil.

Indra (2012) merancang sistem absensi yang dapat mencatat data persentase kehadiran para karyawan. Untuk Membuat sistem ini digunakan metode *eigenface* untuk melakukan pengenalan wajah. Tingkat keberhasilan yang didapatkan dari sistem pengenalan ini masih rendah. dikarenakan banyak factor yang dapat mempengaruhi proses pengenalan. Beberapa factor yang dimaksud yaitu intensitas cahaya, sudut wajah, jarak wajah, dan aksesoris yang digunakan

Aris (2016) merancang sistem pengenalan wajah yang menggunakan ekstraksi fitur berbasis *Principal* *Component Analysis* (PCA). Teknik ini melibatkan pengambilan komponen utamadari database wajah. Untuk mengetahui keakuratan sistem pengenalan wajah manusia yang dirancang pada tugas akhir ini, telah dilakukan uji coba sistem dengan menggunakan input sebanyak 60 *Citra* wajah dari database. Dari hasil pengujian sistem ini, didapatkan hasil performansi sistem adalah 80% dalam mengenali *Citra* input dengan benar.

Abhirawa (2017) merancang *Convolutional Neural Nerwor*k untuk membuat sistem pengenalan wajah dengan menggunakan *Dataset The Extended Yale Face Database B*, yang berupa dataset foto wajah. Menggunakan proses *dropout* diperoleh akurasi pengenalan wajah 75.79%, namun metode yang digunakan saat ini akurasinya masih sangat rendah untuk mengenali wajah.

## Landasan Teori

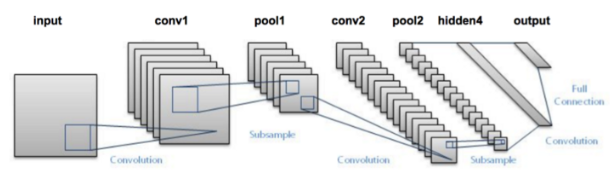
## Convolulutional Neural Network (*CNN*)

*Convolutional Neural Network* (*CNN*) adalah pengembangan dari Multilayer Perceptron (MLP) yang didesain untuk mengolah data dua dimensi. *CNN* termasuk dalam jenis Deep Neural Network karena kedalaman jaringan yang tinggi dan banyak diaplikasikan pada data citra.

Macam-macam *CNN* arsitektur yaitu

* 1. LeNet-5(1998)

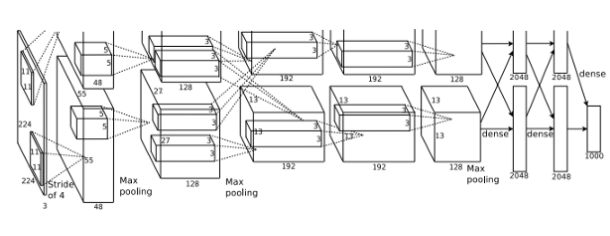
LeNet-5, jaringan konvolusional 7-level perintis oleh LeCun et al pada tahun 1998, yang mengklasifikasikan digit, diterapkan oleh beberapa bank untuk mengenali nomor tulisan tangan pada cek (cek) yang didigitalkan dalam input input skala abu-abu 32x32 pixel. Kemampuan untuk memproses gambar dengan resolusi lebih tinggi membutuhkan lapisan yang lebih besar dan lebih convolutional, sehingga teknik ini dibatasi oleh ketersediaan sumber daya komputasi.



Gambar 2.1. LeNet-5 arsitektur

* 1. AlexNet (2012)

[AlexNet](https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf) secara signifikan mengungguli semua pesaing sebelumnya dan memenangkan tantangan dengan mengurangi kesalahan top-5 dari 26% menjadi 15,3%. Tingkat kesalahan top-5 tempat kedua, yang bukan variasi *CNN*, adalah sekitar 26,2%.

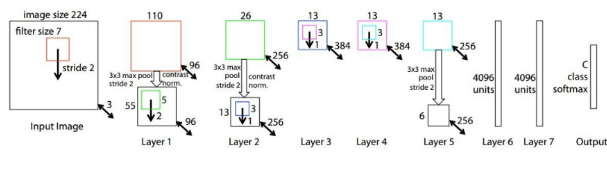


Gambar 2.2. Alexnet arsitektur

Jaringan ini memiliki arsitektur yang sangat mirip dengan [LeNet](http://yann.lecun.com/exdb/publis/pdf/lecun-98.pdf) oleh Yann LeCun et al tetapi lebih dalam, dengan lebih banyak filter per lapisan, dan dengan lapisan convolutional yang bertumpuk. Itu terdiri dari 11x11, 5x5,3x3, konvolusi, *Max Pool*ing, dropout, augmentasi data, aktivasi ReLU, SGD dengan momentum. Ini melekat aktivasi ReLU setelah setiap lapisan konvolusional dan terhubung sepenuhnya. AlexNet dilatih selama 6 hari secara bersamaan pada dua GPU Nvidia Geforce GTX 580 yang merupakan alasan mengapa jaringan mereka dipecah menjadi dua jaringan pipa. AlexNet dirancang oleh grup SuperVision, yang terdiri dari Alex Krizhevsky, Geoffrey Hinton, dan Ilya Sutskever.

* 1. ZFNet (2013)

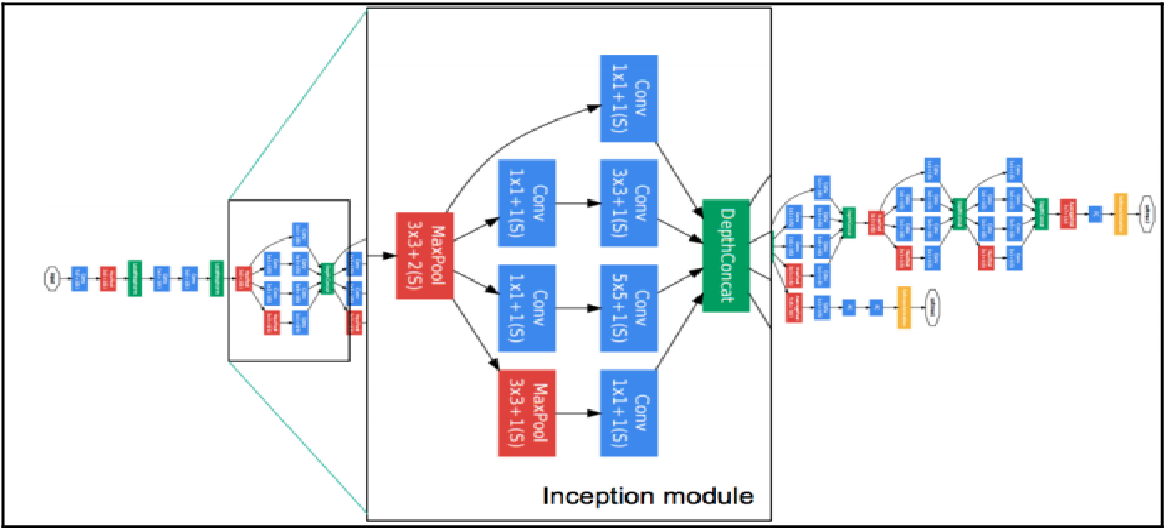
pemenang ILSVRC 2013 juga merupakan *CNN* yang kemudian dikenal sebagai ZFNet. Ini mencapai tingkat kesalahan top-5 14,8% yang sekarang sudah setengah dari tingkat kesalahan non-saraf yang disebutkan sebelumnya. Itu sebagian besar merupakan sebuah pencapaian dengan men-tweak parameter-hyper dari AlexNet sambil mempertahankan struktur yang sama dengan elemen-elemen Belajar Dalam tambahan



Gambar 2.3. ZFNet arsitektur

* 1. GoogLeNet/Inception (2014)

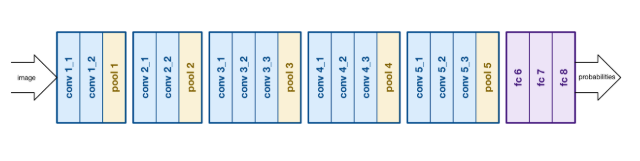
Pemenang kompetisi ILSVRC 2014 adalah GoogLeNet (alias Inception V1) dari Google. Itu mencapai tingkat kesalahan top-5 dari 6,67%!  sangat dekat dengan kinerja tingkat manusia yang sekarang dipaksa oleh panitia tantangan untuk dievaluasi. Ternyata, ini sebenarnya agak sulit dilakukan dan membutuhkan beberapa pelatihan manusia untuk mengalahkan ketepatan GoogLeNets. Setelah beberapa hari pelatihan, pakar manusia (Andrej Karpathy) mampu mencapai tingkat kesalahan top-5 sebesar 5,1% (model tunggal) dan 3,6% (ansambel). Jaringan menggunakan *CNN* yang terinspirasi oleh LeNet tetapi menerapkan elemen novel yang dijuluki modul awal. Ini digunakan normalisasi batch, distorsi gambar dan RMSprop. Modul ini didasarkan pada beberapa konvolusi yang sangat kecil untuk mengurangi jumlah parameter secara drastis.



Gambar 2.4. GoogLeNet arsitektur

* 1. VGGNet(2014)

Runner-up di kompetisi ILSVRC 2014 dijuluki VGGNet oleh komunitas dan dikembangkan oleh Simonyan dan Zisserman. VGGNet terdiri dari 16 lapisan konvolusional dan sangat menarik karena arsitekturnya yang sangat seragam.Mirip dengan AlexNet, hanya konvolusi 3x3, tetapi banyak filter. Dilatih dengan 4 GPU selama 2-3 minggu. Saat ini pilihan paling disukai di komunitas untuk mengekstraksi fitur dari gambar. Konfigurasi berat VGGNet tersedia untuk umum dan telah digunakan di banyak aplikasi dan tantangan lainnya sebagai ekstraktor fitur dasar. Namun, VGGNet terdiri dari 138 juta parameter, yang bisa sedikit menantang untuk ditangani



Gambar 2.5. VGGNet arsitektur

* 1. Residual Network (2015)

*Residual networks (Resnet)* Ini adalah arsitektur pemenang ILSVRC2015 dengan 152 lapisan. Kontribusi utamanya adalah menggunakan normalisasi batch dan koneksi loncatan khusus untuk melatih arsitektur yang lebih dalam. *Resnet* dengan 1000 lapisan dapat dilatih dengan teknik-teknik tersebut. Namun, secara empiris menemukan bahwa *Resnet* biasanya beroperasi pada blok kedalaman yang relatif rendah (20 - 30 lapisan), yang bertindak secara paralel, daripada secara serial mengalir ke seluruh panjang jaringan.

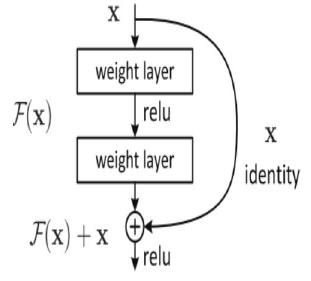
*Deep residual networks* (*Resnets*) terdiri dari banyak "*Residual Units"* yang ditumpuk. Setiap unit dapat dinyatakan dalam bentuk umum:

𝒚𝒍 = 𝒉( 𝒙𝒍) + 𝑭 (𝒙𝒍, 𝒘𝒍) , 𝒙𝒍 + 𝟏 = 𝒇(𝒚𝒍) (2.1)

xl dan xl + 1 adalah input dan output dari unit ke-l, dan F adalah fungsi residual. Dalam, h (xl) = xl adalah pemetaan identitas dan f adalah fungsi *Relu*. *wl* adalah satu **R** set bobot (dan bias) yang dikaitkan dengan *Unit Residual* ke-l. *Resnets* yang lebih dari 100 lapisan telah menunjukkan akurasi *state-of-the-art* untuk beberapa tugas pengenalan yang menantang *di ImageNet* dan *MS COCO* kompetisi.

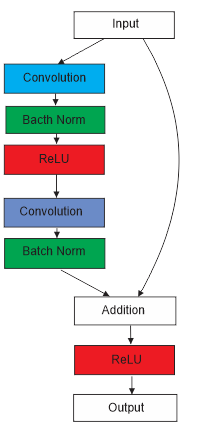
Gagasan utama *Resnets* adalah untuk mempelajari fungsi residual aditif F sehubungan dengan h (xl), dengan pilihan kunci menggunakan pemetaan identitas h (xl) = xl. Ini diwujudkan dengan melampirkan koneksi lewati identitas ("jalan pintas").

Ide inti dari *Resnet* adalah memperkenalkan apa yang disebut "koneksi pintas identitas" yang melompati satu atau lebih lapisan, seperti yang ditunjukkan pada gambar berikut:



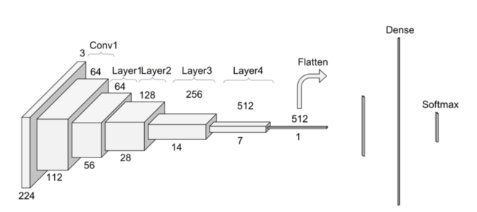
Gambar 2.6. Proses pemintasan blok

Para peneliti berpendapat bahwa susunan lapisan seharusnya tidak menurunkan kinerja jaringan, karena resnet dapat menumpuk pemetaan identitas (layer yang tidak melakukan apa-apa) pada jaringan dan arsitektur yang dihasilkan akan melakukan hal yang sama. Ini menunjukkan bahwa model yang lebih dalam seharusnya tidak menghasilkan kesalahan pelatihan yang lebih tinggi dari pada model yang lebih rendah.



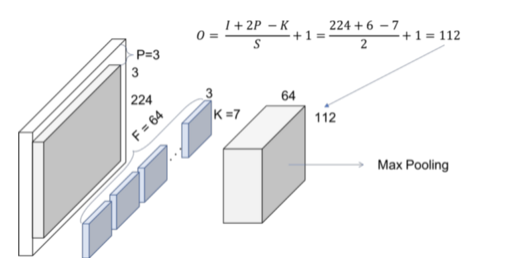
Gambar 2.7. Pemintasan Resnet

*Resnet* bekerja secara parallel yaitu pada saat diberi inputan jalur yang dilalui ada 2 yaitu pertama lurus ke proses *convolutional*, *Batch Norm, Fungsi Activation Relu*, dilanjutkan dengan *Convolution* lagi dan *Bacth Norm* dan yang kedua menuju addition pada proses ini dilakukan proses menggabungkan layer input yang dimasukan dan digabungkan dengan proses *convolutional* kemudian diteruskan kembali oleh *Relu* dan mengeluarkan hasil berupa *output*(faris, 2018).

**

Gambar 2.8. layer resnet

Langkah pertama pada *Resnet* sebelum memasuki perilaku lapisan umum adalah blok - disebut di sini Conv1 - terdiri dari konvolusi + normalisasi batch + operasi penyatuan maks Jadi, pertama ada operasi konvolusi. Pada Gambar 2.3 dapat dilihat bahwa *resnet* menggunakan ukuran kernel 7, dan ukuran peta fitur 64.



Gambar 2.9. konvolusi

Dapat disimpulkan bahwa *resnet* telah diisi dengan nol 3 kali pada setiap dimensi. Dengan mempertimbangkan hal ini, dapat dilihat pada Gambar 4 bahwa ukuran output dari operasi itu adalah volume (112x122). Karena setiap filter konvolusi (dari 64) menyediakan satu saluran dalam volume output, selanjutnya berakhir dengan volume output (112x112x64) - perhatikan ini bebas dari dimensi bets untuk menyederhanakan penjelasan.

## *Python*

*Python* adalah bahasa pemrograman model skrip (*scripting language*) yang berorientasi obyek. *Python* dapat digunakan untuk berbagai keperluan pengembangan perangkat lunak dan dapat berjalan di berbagai *platform* sistem operasi. *Python* merupakan bahasa pemrograman yang freeware atau perangkat bebas dalam arti sebenarnya, tidak ada batasan dalam penyalinannya atau mendistribusikannya, lengkap dengan *source* codenya, debugger dan profiler, antarmuka yang terkandung di dalamnya untuk pelayanan antarmuka, fungsi sistem, *GUI* (antarmuka pengguna grafis), dan basis datanya (Kumar, 2015).

Adapun *Library python* yang digunakan dalam penelitian ini adalah :

* 1. *Keras*

*Keras* adalah jaringan jaringan saraf tingkat tinggi, yang ditulis dengan *Python* dan mampu berjalan di atas *TensorFlow*  , CNTK , atau Theano. Ini dikembangkan dengan fokus pada memungkinkan eksperimen cepat. Mampu pergi dari ide ke hasil dengan penundaan yang paling mungkin adalah kunci untuk melakukan penelitian yang baik. Pada tahun 2017, tim *TensorFlow*  Google memutuskan untuk mendukung *Keras* di perpustakaan inti *TensorFlow* . Chollet menjelaskan bahwa *Keras* dipahami sebagai antarmuka daripada kerangka pembelajaran mesin mandiri. Ini menawarkan rangkaian abstraksi yang lebih tinggi dan lebih intuitif yang membuatnya mudah untuk mengembangkan model pembelajaran mendalam terlepas dari backend komputasi yang digunakan.

* 1. *NumPy (Numerical Python)*

*Numpy* adalah pustaka aljabar linier dalam *Python*. Ini adalah pustaka yang sangat penting tempat hampir setiap sains data atau mesin mempelajari paket *Python* seperti SciPy (Scientific *Python*), Mat − plotlib (plotting library), Scikit-learning, dll tergantung pada batas yang wajar. NumPy sangat berguna untuk melakukan operasi matematika dan logis pada Array. Ini menyediakan banyak fitur berguna untuk operasi pada n-array dan matriks dalam *Python*. (aigiomawu, 2018)

* 1. *Cv2* atau *OpenCV*

*Cv2* adalah sebuah *library* fungsi pemrograman yang ditujukan untuk *computer vision.* Awalnya dikembangkan oleh pusat penelitian Intel di Nizhny Novgorod (Rusia), kemudian didukung oleh Willow Garage dan sekarang dikelola oleh Itseez. *Library OpenCV* di bawah lisensi BSD *open-source* gratis dan cross-platform untuk digunakan. Didalamnya terdapat ratusan *Algoritma computer vision.*

* 1. *Matplotlib*

*Matplotlib* adalah pustaka penyusunan 2D untuk bahasa pemrograman *Python* yang menghasilkan angka kualitas publikasi dalam berbagai format hardcopy serta lingkungan interaktif di seluruh *platform. Matplotlib* mencoba untuk membuat hal-hal mudah mudah serta hal-hal yang sulit dan menyediakan API berorientasi objek untuk menanamkan plot ke dalam aplikasi menggunakan toolkit GUI tujuan umum seperti wx*Python*, Qt, atau GTK +. *Matplotlib* terutama ditulis oleh John D. Hunter, memiliki komunitas pengembangan aktif, dan didistribusikan di bawah lisensi gaya BSD (Kumar, 2015)

## *Deep Learning*



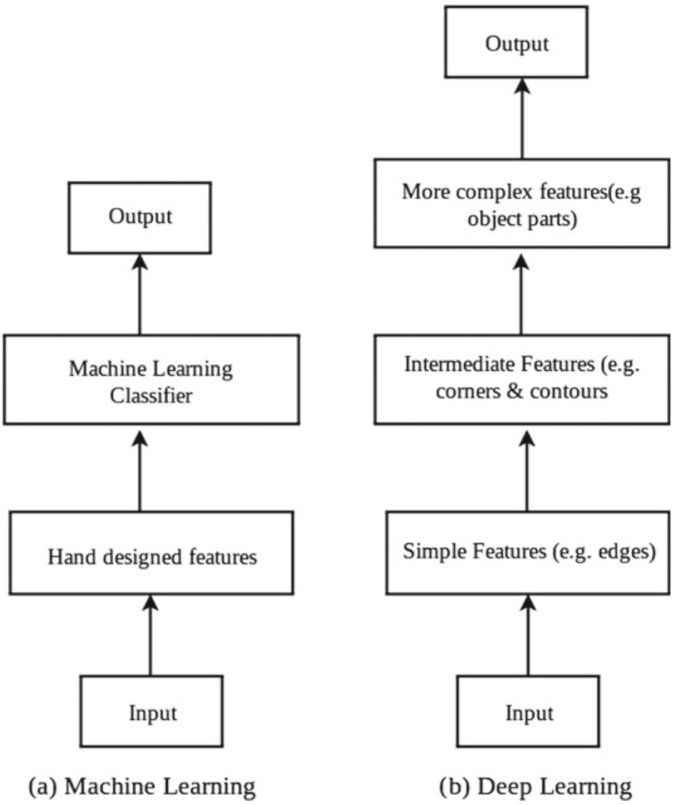
Gambar 2.10 *Deep Learning*

*Deep Learning* adalah area baru dari *machine learning* yang semakin populer pada dekade ini. Istilah *Deep Learning* merujuk pada arsitektur ANN yang terdiri dari beberapa *hidden layers* (*Deep Networks*) yang berfungsi untuk mengenali fitur-fitur yang berbeda pada *input*. Algoritma *Deep Learning* bertujuan untuk mencari struktur yang tidak diketahui untuk mendapatkan representasi yang baik dari *input* (Wani et al., 2018).

Metode *machine learning* yang konvensional terbatas dalam cara memproses data mentah. Dalam waktu yang lama untuk dapat mengenal pola menggunakan metode *machine learning* membutuhkan keahlian khusus dalam bidang-bidang tertentu dan membutuhkan keahlian rekayasa yang sangat hati-hati sesuai kasus yang dihadapi untuk dapat mengekstraksi fitur dari sebuah *input*.

Namun dengan *Deep Learning* proses pengenalan pola dan rekayasa secara manual yang membutuhkan keahlian dalam domain tertentu tersebut dapat dilakukan secara otomatis selama proses pembelajaran. Dalam proses pembelajaran (*training*) metode *Deep Learning* dapat mengenali fitur-fitur yang ada pada data mentah secara otomatis dengan hasil yang lebih baik dari cara manual (Wani et al., 2018).

Metode pembelajaran fitur pada Deep Learning dilakukan dengan membentuk layer hirarki dimana setiap layer dibangun diatas layer yang lain. Layer yang paling bawah dari model ini bertanggung jawab untuk mempelajari representasi dasar dari masalah, dan layer-layer diatasnya bertanggung jawab untuk membentuk konsep yang lebih kompleks dari data. Sebagai contoh dalam kasus pengenalan wajah pada gambar, setiap pixel yang ada pada gambar akan dimasukkan ke hirarki layer. Setiap hidden layer pada struktur hirarki ini kemudian akan mengekstrak fitur-fitur dari gambar input. Pada layer pertama hirarki ini akan mendeteksi tepi-tepi dari wajah, kemudian pada pada layer kedua akan dipelajari garis-garis pipi, alis, cekungan pada dagu, pada layer berikutnya garis-garis yang telah dipelajari akan membentuk gambaran yang lebih abstrak dari gambar. Semua pembelajaran yang dilakukan pada setiap layer ini dilakukan secara otomatis tanpa campur tangan manusia. Secara visual perbedaan tehnik *machine learning* konvesional dan *deep learning* bisa dilihat pada gambar 2.6 dibawah.

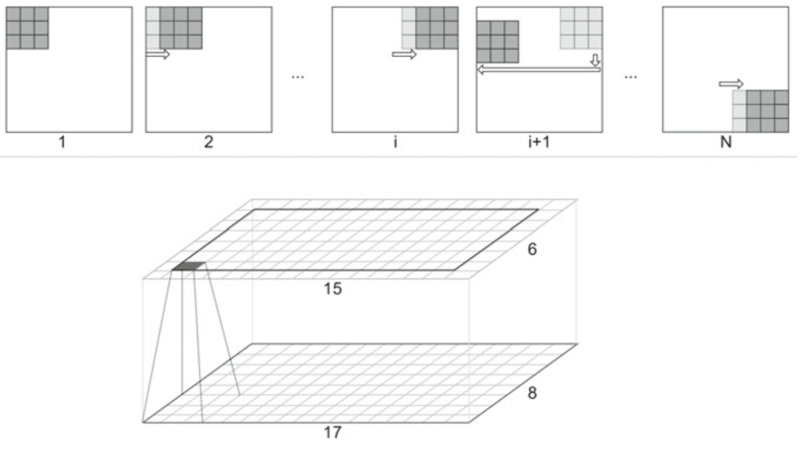


Gambar 2.11. Perbedaan *Machine Learning* dan *Deep Learning*

Adapun layer ekstraksi fitur yang bisa digunakan pada arsitektur *Deep Learning* adalah sebagai berikut :

* + - * 1. Layer Konvolusi

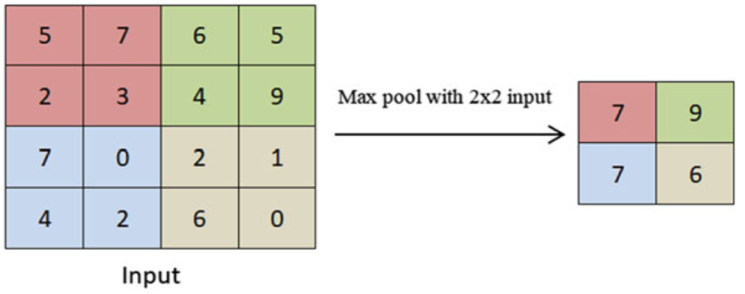
LayerKonvolusi (*Convolutional Layer*)adalah layer operasi konvolusi 2D, dimana dengan operasi konvolusi setiap blok wilayah pada gambar dapat dipelajari dan menjadi fitur dari gambar tersebut. Komponen dari operasi konvolusi adalah matriks persegi yang disebut filter/kernel umumnya berdimensi 3x3, filter/kernel pada operasi konvolusi berfungsi untuk mengekstrak fitur setiap wilayah yang ada pada gambar. Dalam proses *training* filter/kernel akan menjadi *learnable parameter* yaitu setiap nilai matriks kernel akan dipelajari pada saat *training*. Pada setiap layer konvolusijumlah kernel/filter bisa lebih dari satu sesuai kompleksitas dan jumlah fitur yang ingin dipelajari dari input gambar. Sebagai contoh jika pada sebuah gambar dengan satu kanal warna ingin dipelajari 64 jenis fitur maka pada layer *convolutional* bisa memiliki 64 jenis kernel/filter dengan dimensi sehingga hasil dari operasi layer *convolutional* adalah 64 gambar atau juga disebut 64 kanal. Ilustrasi proses konvolusi bisa dilihat pada gambar 2.7 dibawah.



Gambar 2.12. Ilustrasi Proses Konvolusi

* + - * 1. *Max Pool*

Layer *Max Pool* pada lapisan jaringan ANN berfungsi untuk mereduksi ukuran gambar dengan metode *down sampling.* Layer *Max Pool* akan menyimpulkan nilai tertinggi pada gambar dalam ukuran region tertentu (Wani et al., 2018). Sebagai contoh gambar 2.8 dibawah operasi *Max Pool* akan mengambil nilai tertinggi pada gambar dalam setiap 2x2 wilayah pada gambar.



Gambar 2.13. Ilustrasi Operasi *Max Pool*

* + - * 1. *Resnet*

Proses *Batch Normalization* (BN)pada lapisan ANN berfungsi untuk menormalisasi skala nilai *input*. Menurut Sergey Ioffe (2015) dalam proses *training* menambahkan lapisan BN dapat meningkatkan generalisasi pada model sehingga model dapat belajar dengan baik tanpa *overfit* (Ioffe & Szegedy, 2015)*.* Proses BN bisa dilihat dalam persamaan dibawah (Wani et al., 2018).

|  |  |
| --- | --- |
|  | (2.2) |

**Dimana:**

adalah hasil proses BN

adalah data yang akan dinormalisasi

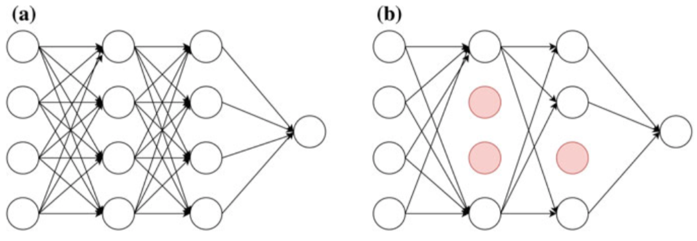
adalah nilai rata-rata pada *batch*

adalah nilai variansi dalam batch

Menumpuk layer Konvolusi dengan *Max Pool* menjadi beberapa layer disebut dengan *CNN* (*Convolutional Neural Network*).

* + - * 1. Dropout

Lapisan jaringan ANN yang terdiri dari beberapa lapis yang kompleks dapat membuat model mempelajari fitur yang rumit pada data, setiap fitur yang dipelajari akan tersambung ke sejumlah *neuron* atau disebut *Fully Connected Layer* (FC) untuk membuat kesimpulan dari fitur yang didapat.LayerFCakan saling terhubung dengan layer FC yang lain dan sangat mudah mengalami *overfit. Overfit* adalah keadaan dimana model mempelajari data *training* dengan sangat baik namun gagal dalam memprediksi kesimpulan dari data baru, untuk mengatasi hal ini layer *dropout* dapat ditambahkan pada lapisan ANN. Layer *dropout* akan memutus secara acak sambungan FC pada saat *training* sehingga hanya sambungan yang masih terhubung yang akan dilatih pada saat *training* (Wani et al., 2018)*.* Ilustrasi proses *dropout* bisa dilihat pada gambar 2.9 dibawah.



Gambar 2.14. Ilustrasi Proses *Dropout*

## *Training*

*Training* dalam *Deep Learning* adalah proses melatih model untuk mempelajari data untuk mendapatkan hasil yang diinginkan. Dalam proses *training* ada beberapa komponen yaitu:

* + - * 1. Epoch

*Epoch* dalam *training* merujuk pada berapa kali semua data *training* akan digunakan untuk belajar. Menggunakan data *training* dalam melatih model tidak cukup hanya sekali, dibutuhkan beberapa kali agar model dapat belajar dengan baik.

* + - * 1. Batch Size

*Batch Size* dalam *training* merujuk pada jumlah data yang akan diproses untuk latihan dalam satu waktu. Metode *deep learning* pada umumnya membutuhkan data dalam jumlah besar agar dapat belajar dengan baik. Dalam proses belajar, memasukkan keseluruhan data ke dalam jaringan dalam waktu bersamaan akan memakan banyak sumber daya memory dan operasi komputasi. Agar dapat diproses dengan baik data yang dalam jumlah banyak akan dipecah menjadi beberapa bagian yang disebut dengan *mini batch* dimana jumlah data setiap *mini batch* disebut dengan *Batch Size* dan pemrosesan setiap *batch* disebut dengan iterasi*.* Umumnya ukuran *Batch Size* yang digunakan adalah kelipatan 2 misal 32, 64, 128 jumlah data.

## *Learning Rate*

Tingkat belajar adalah salah satu parameter hiper paling penting ketika datang ke pelatihan jaringan saraf. Ini menentukan besarnya pembobotan (atau parameter)

pembaruan. Ini juga merupakan parameter tersulit yang harus ditetapkan karena dapat berdampak signifikan pada kinerja model. Semakin rendah nilainya, semakin lambat *resnet* melakukan pengecekan data. Nilainya berkisar antara 0 < lr < 1.

Jika tingkat pembelajaran disetel terlalu rendah, kemajuan pelatihan tidak efisien karena memakan waktu dengan pembaruan bobot yang kecil. Jika tingkat pembelajaran ditetapkan terlalu tinggi, itu dapat menyebabkan perilaku yang berbeda dalam *loss function* (Janocha, 2017)

## *AI Framewk*

*TensorFlow*

*TensorFlow* adalah platform *open source end to end*  untuk pembelajaran mesin. Ini memiliki ekosistem alat, perpustakaan, dan sumber daya komunitas komprehensif yang fleksibel, yang memungkinkan para peneliti mendorong teknologi mutakhir dalam ML dan pengembang dengan mudah membangun dan menggunakan aplikasi bertenaga ML.

Fungsi *Tensor Flow* adalah :

* Ini berguna untuk membuat dan bereksperimen dengan arsitektur pembelajaran yang mendalam, dan formulasinya nyaman untuk integrasi data seperti memasukkan grafik, tabel SQL, dan gambar secara bersamaan.
* itu didukung oleh Google yang menjamin itu akan tetap ada untuk sementara waktu, maka masuk akal untuk menginvestasikan waktu dan sumber daya untuk mempelajarinya.

TF beroperasi dengan **grafik perhitungan statis** . Yaitu, pertama-tama mendefinisikan grafik, selanjutnya menjalankan perhitungan dan, jika perlu membuat perubahan pada arsitektur dan melatih kembali model. Pendekatan seperti itu dipilih demi efisiensi, tetapi banyak alat jaringan saraf modern dapat memperhitungkan perbaikan dalam proses pembelajaran tanpa kehilangan yang signifikan dalam kecepatan belajar. Dalam hal ini, pesaing utama *TensorFlow* adalah ***PyTorch****.*

*Keras*

*Keras* adalah pustaka [neural-network](https://en.wikipedia.org/wiki/Artificial_neural_network)[open-source yang](https://en.wikipedia.org/wiki/Open-source_software) ditulis dengan [Python](https://en.wikipedia.org/wiki/Python_(programming_language)) . Itu mampu berjalan di atas [*TensorFlow*](https://en.wikipedia.org/wiki/TensorFlow) , [Microsoft Cognitive Toolkit](https://en.wikipedia.org/wiki/Microsoft_Cognitive_Toolkit),[Theano](https://en.wikipedia.org/wiki/Theano_(software)), atau [PlaidML](https://en.wikipedia.org/wiki/PlaidML). Dirancang untuk memungkinkan eksperimen cepat dengan [jaringan saraf yang dalam](https://en.wikipedia.org/wiki/Deep_learning) , *Keras* berfokus untuk menjadi ramah pengguna, modular, dan dapat dikembangkan. *Keras* berisi banyak implementasi dari blok-blok pembangun jaringan-saraf yang biasa digunakan seperti lapisan, [tujuan](https://en.wikipedia.org/wiki/Objective_function) , [fungsi aktivasi](https://en.wikipedia.org/wiki/Activation_function) , [pengoptimal](https://en.wikipedia.org/wiki/Mathematical_optimization) , dan sejumlah alat untuk membuat bekerja dengan data gambar dan teks lebih mudah. Selain jaringan saraf standar, *Keras* memiliki dukungan untuk [jaringan saraf](https://en.wikipedia.org/wiki/Recurrent_neural_networks)[konvolusional](https://en.wikipedia.org/wiki/Convolutional_neural_networks) dan [berulang](https://en.wikipedia.org/wiki/Recurrent_neural_networks) . Ini mendukung lapisan utilitas umum lainnya seperti dropout, normalisasi batch, dan pengumpulan.

*PyTorch*

*PyTorch* adalah [perpustakaan](https://en.wikipedia.org/wiki/Library_(computing))[pembelajaran mesin](https://en.wikipedia.org/wiki/Machine_learning)[sumber terbuka](https://en.wikipedia.org/wiki/Open_source) berdasarkan [perpustakaan](https://en.wikipedia.org/wiki/Library_(computing))[Torch](https://en.wikipedia.org/wiki/Torch_(machine_learning)) , digunakan untuk aplikasi seperti [visi komputer](https://en.wikipedia.org/wiki/Computer_vision) dan [pemrosesan bahasa alami](https://en.wikipedia.org/wiki/Natural_language_processing). Ini terutama dikembangkan oleh kelompok riset kecerdasan buatan [Facebook](https://en.wikipedia.org/wiki/Facebook) . Ini adalah [perangkat lunak bebas dan sumber terbuka yang](https://en.wikipedia.org/wiki/Free_and_open-source_software) dirilis di bawah [lisensi BSD yang Dimodifikasi](https://en.wikipedia.org/wiki/Modified_BSD_License) . Meskipun antarmuka [Python](https://en.wikipedia.org/wiki/Python_(programming_language)) lebih dipoles dan fokus utama pengembangan, *PyTorch* juga memiliki [antarmuka](https://en.wikipedia.org/wiki/Front_and_back_ends) C ++ .  Selanjutnya, perangkat lunak [bahasa pemrograman probabilistik](https://en.wikipedia.org/wiki/Probabilistic_programming_language)[Uber](https://en.wikipedia.org/wiki/Uber) Pyro menggunakan *PyTorch* sebagai backend. (Pytorch n.d, 2019)

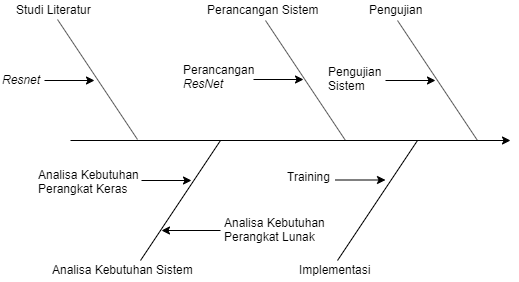
MXNet

MXNet adalah framework mendalam yang sangat skalabel dan dapat digunakan pada berbagai perangkat. Meskipun tampaknya tidak banyak digunakan dibandingkan dengan *TensorFlow*, pertumbuhan MXNet kemungkinan akan didorong dengan menjadi proyek Apache. memungkinkan untuk [pelatihan model](https://en.wikipedia.org/wiki/Training,_test,_and_validation_sets) cepat , dan mendukung [model pemrograman yang](https://en.wikipedia.org/wiki/Programming_model) fleksibel

# BAB III METODOLOGI PENELITIAN

## Tahapan Penelitian

Dari diagram fishbone pada gambar 3.1 dapat dilihat bahwa pembahasan pada bab ini pertama dilakukan studi literatur untuk mengetahui prinsip kerja dari *Residual network* (*Resnet*). Selanjutnya dilakukan analisa kebutuhan sistem yaitu analisa kebutuhan perangkat *keras* dan analisa kebutuhan perangkat lunak untuk mengetahui kebutuhan system dan menghindari hal yang tidak diinginkan selain itu dalam bab ini akan dibahas bagaimana model perancangan *Resnet* serta bagaimana implementasinya pada *training* data yang diharapkan dapat meningkatkan akurasi pengenalan wajah pada dataset Yale dan yang terakhir dilakukan pengujian system untuk mengetahui seberapa besar akurasi peninggalan wajah yang didapatkan.



Gambar 3.1 Tahapan Penelitian

## Studi Literatur

Studi literatur berfungsi untuk memahami dasar teori sebagai landasan ilmiah yang berkaitan dengan kebutuhan penelitian yang ada. Adapun teori-teori pendukung penelitian ini adalah *Residual network (Resnet)* untuk diterapkan dalam pengenalan wajah

## 3.3. Analisa Kebutuhan

Pada tahap analisa kebutuhan sistem akan dilakukan analisa terhadap perangkat *keras* dan perangkat lunak untuk mempermudah dalam penelitian ini

## 3.3.1. Kebutuhan Perangkat *Keras*

Penelitian ini menggunakan sebuah laptop untuk melakukan proses pengolahan data dengan spesifikasi yaitu Komputer PC, RAM 16 GB/MacBook Pro 13

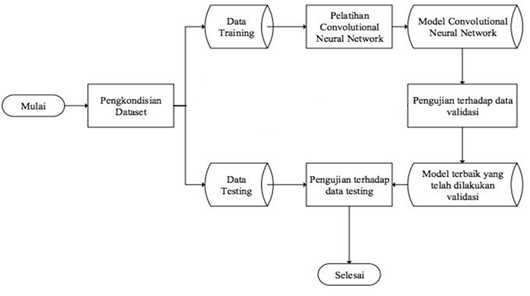
## 3.3.2. Kebutuhan Perangkat Lunak

Selain kebutuhan perangkat *keras*, penelitian ini membutuhkan perangkat lunak dan aplikasi yang mendukung pemrosesan data, perancangan dan evaluasi model pembelajaran sistem yang meliputi :

1. Aplikasi *jupyterlab* (berbasis *python* v.3.6) dan digunakan untuk keperluan *data processing*, konstruksi pohon dan evaluasi model (sudah termasuk dengan paket *library* pendukungnya).
2. *Web browser* digunakan untuk menjalankan aplikasi *jupyterlab* seperti *google* *chrome*.
3. *Microsoft* *Office* digunakan untuk membuat dokumentasi dan laporan hasil penelitian.

## 3.4. Perancangan Sistem

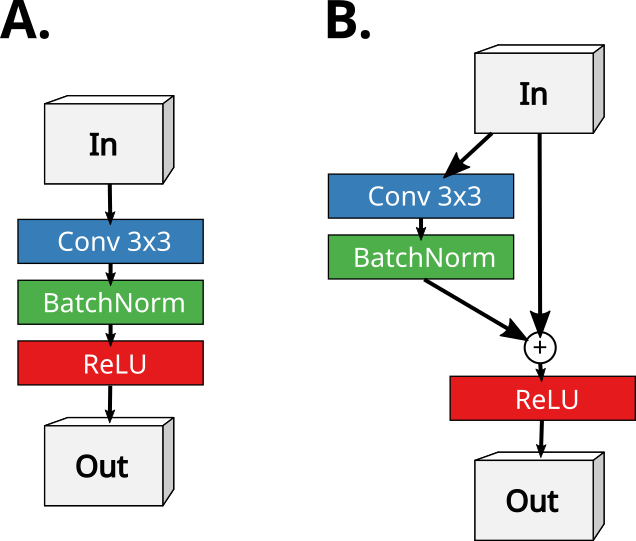
Dari gambar 3.2 peneliti melakukan pengkondisian *dataset* dengan mengecilkan gambar awal yaitu 256x256 menjadi ukuran 32x32 pixel. Selanjutnya dataset dibagi menjadidua bagian yaitu data *training* dan data *testing*. Data *training* adalah peroses pegujian *Algoritma* untuk melatih data sehingga dapat memahami informasi yang terdapat didalam data tersebut sedangkan data *testing* adalah data yang digunakan untuk memvalidasi hasil dari data *training.* Data yang sudah dibagi selanjutnya dilakukan pelatihan dengan *Algoritma* yang *CNN resnet* dengan inputan gambar yang untuk mencari model *convolutional neuratl network* terbaik. Bentuk model terbaik dicari dengan merubah parameter yaitu *learning rate* untuk mendapatkan akurasi terbaik sehingga selanjutnya dilakukanlah pengujian terhadap data *testing* dengan mengatur jumlah data mulai dari 15%,20%,25%,30%, sampai 50 %. Hal ini bertujuan untuk mendapatkan model terbaik sehingga didapatkanlah akurasi pengenalan wajah yang baik.



Gambar 3.2 Perancangan Sistem

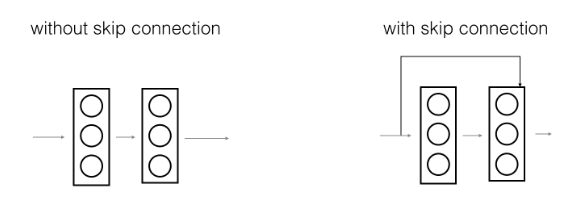
## 3.4.1. Perancangan *Resnet*

Pada gambar 3.3 adalah gambar dari pembentukan *residual network. Resnet* bekerja dengan melakukan lompatan layer. Setiap lapisan *Resnet* terdiri dari beberapa blok. Ini karena ketika *Resnet* masuk lebih dalam, *resnet* melakukannya dengan meningkatkan jumlah operasi di dalam blok, tetapi jumlah total *layer* tetap sama yaitu 4. Operasi di sini mengacu pada konvolusi normalisasi bets dan aktivasi ReLu ke input, kecuali operasi terakhir dari sebuah blok, yang tidak memiliki ReLu.

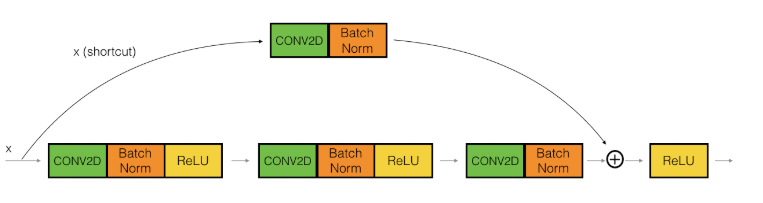
****

Gambar 3.3 Blok *Resnet*

*Resnet* memiliki layer yang di skip. Gambar 3.4 menggambarkan bentuk lompatan *resnet*. Pada gambar di sisi kiri adalah menyusun lapisan konvolusi bersama satu demi satu dan pada sisi kanan terbentuk layer yang penjumlahan layer dari input asal menjadi output pada bagian blok konvolusi.

Gambar 3.4 *Skip Connection resnet*

*Resnet* dibentuk dengan Setiap lapisan mengikuti pola yang sama yaitu *konvolusi* 3x3 dengan dimensi peta fitur tetap (F) yaitu pada kernel [64, 128, 256, 512] masing-masing, mem-bypass input setiap 2 *konvolusi*. Selain itu, dimensi lebar (W) dan tinggi (H) tetap konstan selama seluruh lapisan.



Gambar 3.5 Lapisan *Resnet*

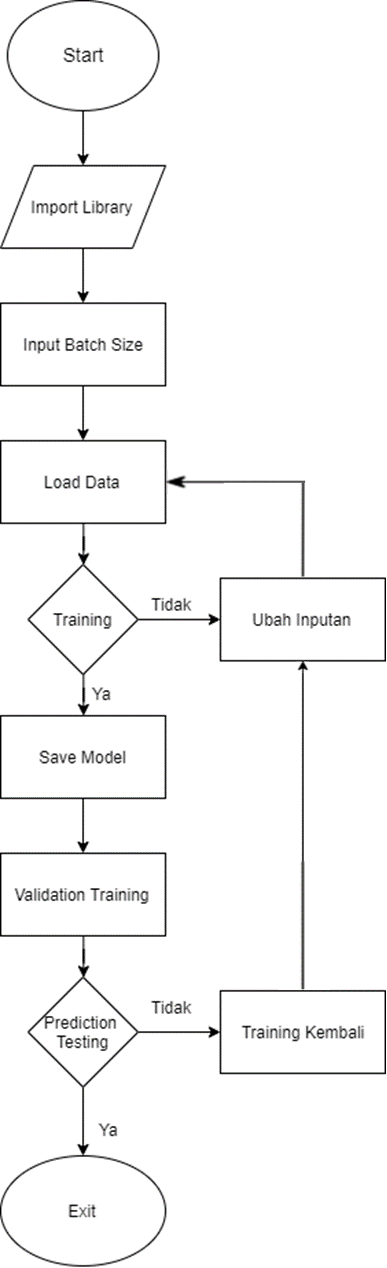
Gambar 3. Adalah proses *Algoritma resnet* bekerja sebelum memasuki layer. Layer yang dimaksud adalah blok - disebut di sini Conv1 - terdiri dari *konvolusi* + normalisasi batch + operasi penyatuan maks.Jadi, pertama ada operasi *konvolusi*. Pada Gambar diatas peneliti menggunakan ukuran kernel 7. Langkah selanjutnya adalah normalisasi bets, yang merupakan operasi normalisasi yang menjadikan nilai yang ada hanya berada pada rentan 0 dan 1

Tahap terakhir adalah melakukan *activation function*. *Activation function* befungsi untuk menentukan apakah neuron tersebut harus “aktif” atau tidak berdasarkan dari weighted sum dari input. *activation function* yang digunakan pada *resnet* adalah *ReLu.* Pada dasarnya *ReLu* melakukan “*treshold*” dari 0 hingga infinity dan selanjutnya *output* akan menjadi inputan selanjutnya. Hal ini bertujuan untuk mendapatkan hasil yang terbaik.

## 3.5. Implementasi

Dalam tahapan implentasi akan dilakukan proses *training.* Pelatihan (*Training)* adalah peroses pegujian *Algoritma* untuk melatih data sehingga dapat memahami informasi yang terdapat didalam data tersebut. Pelatihan dilakukan dengan membuat model *Algoritma* yang selanjutnya ada beberapa parameter yang di atur dengan tujuan untuk mendapatkan akurasi pengenalan wajah yang terbaik. Dalam tahapan implementasi ini dilakukan proses *training* yang berulang berulang yang nantinya akan merubah parameter *leraning rate.* Learning rate merupakan salah satu parameter *training* untuk menghitung nilai koreksi bobot pada waktu proses *training*. Nilai α ini berada pada range nol (0) sampai (1). Semakin besar nilai learning rate, maka proses *training* akan berjalan semakin cepat. Namun apabila nilai learning rate relatif terlalu besar, pada umumnya proses *training* dapat melampaui keadaan optimal yaitu pada saat dicapai nilai error yang paling minimal. Dengan kata lain, learning rate mempengaruhi ketelitian jaringan suatu sistem. Semakin besar learning rate, maka ketelitian jaringan akan semakin berkurang, tetapi berlaku sebaliknya, apabila learning rate-nya semakin kecil, maka ketelitian jaringan akan semakin besar atau bertambah dengan konsekuensi proses *training* akan memakan waktu yang semakin lama*.*

## 3.5.1. Perancangan Sistem



Gambar 3.6 *Flow Chart Training*

Dari *flow chart* diatas adalah proses *training* data. Dalam proses *training* ini dimulai dengan start, import *keras* model yg dimaksud adalah library *keras* yang dipakai untuk men*training* model sebagai wadahnya, input *Batch Size* adalah parameter untuk menload data *training*, semakin besar inputan *Batch Size* maka akan sebagus hasil *training*. H , w, d adalah height, weight dan dimension(channel) sebagai contoh 224x224x3. Epoch adalah iterasi ketika menjalankan *training* maka *training* akan selesai ketika iterasi terpenuhi. Resize adalah merubah ukuran pixel dari ukuran semula menjadi ukuran yang diinginkan. Augmentation yang dimaksud adalah menciptakan gambar *training* melalui berbagai cara pemrosesan atau kombinasi dari beberapa proses,seperti rotasi acak, pergeseran, geser dan membalik, dll. Load data *training* adalah proses *Dataset* di load kedalam proses pen-*training*an, termasuk juga load data *testing* untuk pembuktian probabilitas yang akan muncul. Import *resnet* adalah *Algoritma* *resnet* yang diimport dari library *keras* kemudian akan digunakan sebagai model *training*. *Training* adalah proses pembelajar mesin untuk mengenal *Dataset* yang digunakan. Save model adalah proses penyimpanan model hasil *training*. Validation adalah proses pemeriksaan probabilitas data kebenaran yang muncul berbentuk output citra / gambar. Prediksi adalah proses menebak data hasil validasi. Jika proses diatas telah selesai maka hasil *training* siap menyimpulkan hasil akurasi, validasi akurasi, loss, validasi loss.

## 3.6. Pengujian Sistem

Dalam *Dataset* terdapat folder *training* diimplementasikan untuk membangun model, sementara *testing* (atau validasi) yang ditetapkan untuk memvalidasi model yang dibangun. Pengujian *Dataset* dapat dipilih dengan menerapkan filter acak ke data, misalnya, pilih 20% dari poin secara acak untuk menghasilkan model dan uji terhadap sisa 80%. Parameter yang diuji dalam pengujian sistem ini adalah hasil akurasi dari perubahan laju pembelajaran (*learning rate)* yaitu laju pembelajaran terhadap waktu komputasi, laju pembelajaran terhadap kesalahan (*loss),* dan laju pembelajarann teradap akurasi. Sehingga dari perubahan parameter tersebut akhirnya akan didapatkan model terbaik dan akurasi yang terbaik.

# BAB IV HASIL DAN PEMBAHASAN

Pada penelitian ini peneliti malekukan pelatihan dan pengujian selanjutnya membagi data yang berjumlah 1279 dengan perbandingan 80:20 pada pelatihan (*Training)* dan pengujian (*Testing).* Terdapat 2  *Training* adalah adalah proses yang dilakukan pada folder *training* untuk menguji *Algoritma.* Pada proses *training* terdapat istilah akurasi (*acc)* merupakan hasil pengujian *Algoritma* pada folder *training* dan akurasi (*acc)* adalah bentuk validasi akurasi dari pengujian algroritma pada folder testing.Selanjutnya setelah dilakukan pelatihan (*training)* peneliti melakukan pengujian dengan yaitu melihat akurasi yang didapatkan pada setiap dataset. Data yang diambil secaraacak oleh program sehingga 20% dari 1279 data adalah 315 data pengujian.

## 4.1. Pelatihan

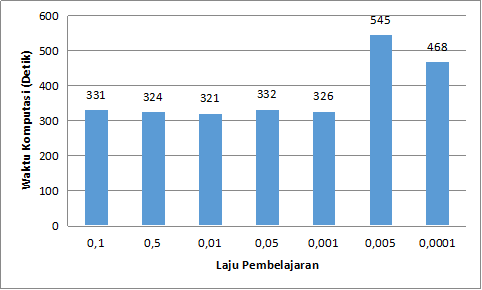
Pelatihan (*Training)* adalah peroses pegujian *Algoritma* untuk melatih data sehingga dapat memahami informasi yang terdapat didalam data tersebut. Pelatihan dilakukan dengan membuat model *Algoritma* yang selanjutnya ada beberapa parameter yang di atur dengan tujuan untuk mendapatkan akurasi pengenalan wajah yang terbaik.

Tabel 4.1 Hasil pengujian *Algoritma* pada folder *training*

|  |  |  |  |
| --- | --- | --- | --- |
| Laju Pembelajaran | Jumlah *iterasi* | Waktu Komputasi (Detik) | Akurasi (%) |
| 0.1 | 50 | 331 | 2 |
| 0.5 | 50 | 324 | 3 |
| 0.01 | 50 | 331 | 99 |
| 0.05 | 50 | 332 | 99 |
| 0.001 | 50 | 326 | 99 |
| 0.005 | 50 | 545 | 99 |
| 0.0001 | 50 | 564 | 92 |

Tabel 4.1 peneliti merubah laju pembelajara (*learning rate)* yaitu 0.1, 0.5, 0.01, 0.05, 0.001, 0.05 dan 0.0001. Dalam penerapan *resnet* peneliti melakukan pelatihan (*training)* data dengan jumlah belajar 50 *iterasi* (*epoch)* dan jumlah data latihan yang digunakan 1279 data. Langkah pertama yaitu menguji *Algoritma* pada data *training* yaitu dengan mengubah laju pembelajaran dan membandingkan hasil laju pembelajaran terhadap waku komputasi, laju pembelajaran terhadap Kesalahan (*loss),* dan laju pembelajaran terhadap akurasi (*acc).* Hal ini bertujuanuntuk mendapatkan laju pembelajaran yang tepat sehingga didapatkalah akurasi yang terbaik dengan tingkat persentase Kesalahan yang mendekati 0%. Untuk mendapatkan hasil pelatihan yang terbaik maka dalam pengujian dilakukan perubahan laju pembelajaran dengan jumlah *iterasi* yang sudah selanjutnya membandingkan hasil laju pembelajaran terhadap waktu komputasi, laju pembelajaran terhadap Kesalahan*,* dan laju pembelajaran terhadap akurasi. Hasil pelatihan yang didapatkanyaitu :

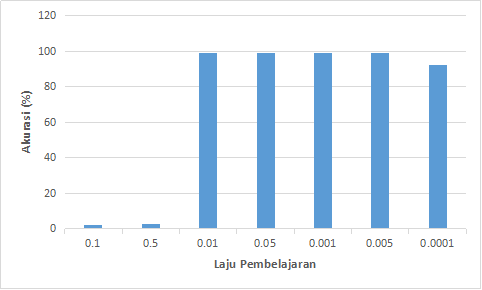
## Laju Pembelajaran Terhadap Waktu Komputasi



Gambar 4.1. Diagram laju pembelajaran terhadap waktu komputasi

Gambar 4.1 adalah diagaram perbandingan laju pembelajaran terhadap waktu komputasi. Dari diagram diatas dapat disipmulkan perubahan laju pembelajaran mempengaruhi waktu komputasi yang dapat dilihat dari laju pembelajara 0.0001 memerlukan waktu komputasi yang cukup lama dibandingkan dengan laju pembelajan 0.005. Waktu komputasi semakin cepat seiring dengan semakin melambatnya laju pembelajaran yaitu dari 0.005 ke laju pembelaran 0.01 tetapi pada laju pembelajaran 0.1 terjadi perubahan waktu komputasi yaitu 331 detik. hal ini disebabkan oleh data *training* yang cukup lama diperores oleh *Algoritma* karena nilai pixel yang diperoses oleh laju pembelajaran memperlambat pelatihan pada data *training.*

## Laju Pembelajaran Terhadap Akurasi (%)



Gambar 4.2. Laju pembelajaran terhadap akurasi

Gambar 4.3 adalah diagram perbandinan laju pembelajaran terhadap akurasi pada data *trainng*. Dari gambar diatas dapat disimpulkan bahwa dengan semakin melambatnya laju pembeajaran yaiu dimula dari laju pembelajaran 0.0001 menuju 0.005 didapatkan nilai akurasi pengujian *Algoritma* pada data *training* yaiu semakin meningkat dari 92% menuju 99% dan pada laju pembelajaran 0.005 menuju 0.01 didapatkan akurasi yang konstan sebesar 99% dapat disimpulkan bahwa akurasi pengenalan yang baik berada pada lerning 0.005-0.01 karena mendaatkan akurasi sebesar 99%. Dari hasil peruban learning rate diatas dapat disimpulkan bahwa pengujin *Algoritma* yang terbak berada pada laju pembelajaran 0.01 dengan persentase kesalahan yaitu 1.1% dan akurasiyaitu 99%. **Maka dapat disimpulkan dari keseluruhan diagram hasil terbaik adalah laju pembelajaran 0.001, waktu komputasi 326 detik dan *loss* 1.4%**

## Validasi (*Testing)*

Tahap selanjutnya adalah melakukan evaluasi terhadap performa *Algoritma* yang dibuat (*Testing*). Dalam tahap ini peneliti akan membagi dataset menjadi 8 pengujian dengan merubah jumlah perbandingan data *testing* terhadap data *training* yang selanjutkan dari hasil perubahan jumlah dataset akan diperloeh akurasi pengenalan wajah. Data *testing* ( validasi) yang ditetapkan untuk memvalidasi model yang dibangun. Pengujian *Dataset* dapat dipilih dengan menerapkan filter acak ke data, misalnya, pilih 20% dari poin secara acak untuk menghasilkan model dan uji terhadap sisa 80%. Data pengujian dirubah untuk melihat. Pengaruh perubahan data terhadap hasil akurasi.

Tabel 4.2. Tabel Validation (*Testing)*

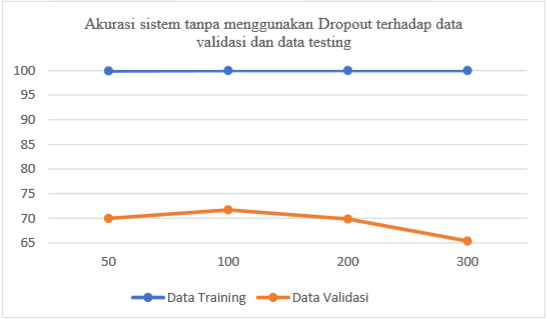
|  |  |  |  |
| --- | --- | --- | --- |
| No urut Pengujian | Data Pengujian | | Akurasi (%) |
| Jumlah | Persentase (%) |
| 1 | 240 | 15 | 87 |
| 2 | 315 | 20 | 92 |
| 3 | 400 | 25 | 86 |
| 4 | 480 | 30 | 84 |
| 5 | 560 | 35 | 84 |
| 6 | 640 | 40 | 80 |
| 7 | 720 | 45 | 78 |
| 8 | 800 | 50 | 76 |

Dari tabel 4.2 dilakukan 8 kali pengujian dengan parameter jumlah data yang dirubah mulai dari 15% sampai 50 %. Dari table diatas dapat disimpulkan jumlah datas jumlah data akan mempengaruhi akurasi pengenalan wajah yaitu pada nomor urut pengujian 2. dapat dilihat dari hasil pengujian yang dilakukan 8 kali. Pada saat *testing* berkurang secara otomatis data *training* akan meningkat. Peningkatan data *training* memberikan dampak akan semakin banyak data yang dilatih sehingga *resnet* dapat mempelajari data semakin baik. Akan tetapi pada saat data *testing* bertambah jumlahnya maka hasil akurasi yang didapatkan menurun. Hal ini dikarenakan data yang dilatih berkurang sehingga dapat dapat disimpulkan perubahan data pasti berpengaruh terhadap hasil akurasi pengenalan wajah dan akurasi pengenalan wajah pada data testing adalah 92%

## Analisis Penelitian Sebelumnya

Penelitian sebelumnya dilakukan oleh abirawa (2017) membahas tentang sistem pengenalan wajah menggunakan *convolutional neural network (CNN)*. Dataset yang digunakan adalah The Extended Yale Face Database B, yang berupa dataset foto wajah. Hasil pengujian yang dilakukan oleh abirawa adalah :

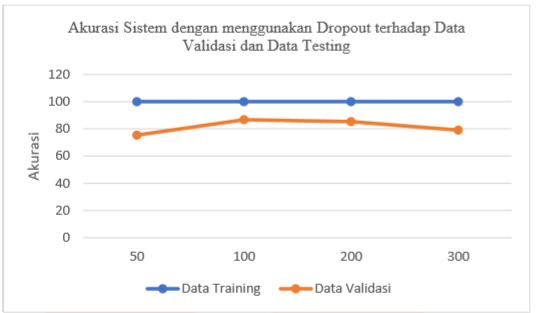
## Pengaruh Sistem Tanpa Dropout



Gambar 4.3. Pengaruh sistem tanpa menggunakan dropout

Berdasarkan Gambar 4.3, didapatkan bahwa sistem memiliki batas optimum pada epoch 100 yang memiliki nilai 100% pada data *training* dan 71.71% pada data validasi, yang dimana selisih akurasi diantara keduanya paling kecil yang memiliki nilai 28.29%. Setelah epoch 100 data menjadi overfit karena selisih akurasi antara jarak data *training* dan data testing bertambah seiring dengan bertambahnya epoch, yaitu 30.14% pada epoch 200 dan 34.61% pada epoch 300. Pada data *training*, sistem belum mempelajari ciri dengan baik pada epoch 50 karena belum memiliki akurasi sebesar 100%, namun sistem sudah mempelajari ciri dengan baik pada epoch 100 karena memiliki akurasi sebesar 100%.

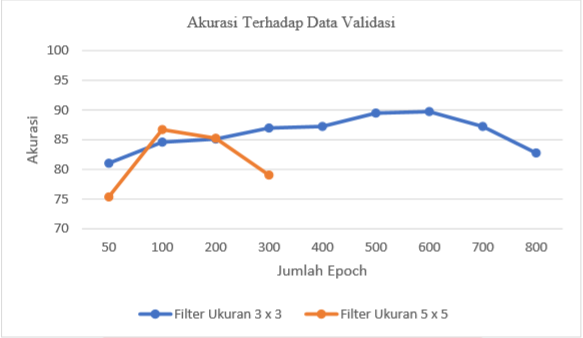
## Pengaruh Sistem menggunakan Dropout



Gambar 4.4. Pengaruh sistem dengan menggunakan Dropout

Berdasarkan Gambar 4.4, dapat dibuktikan juga bahwa dropout mampu mempercepat proses learning. Hal ini dibuktikan dengan akurasi 100% terhadap data *training* yang sudah terjadi pada epoch 50. Sistem yang menggunakan dropout memiliki batas optimum terhadap data validasi yang terjadi pada epoch 100. Batas optimum yang terjadi memiliki nilai akurasi sebesar 86.71% dan memiliki selisih paling sedikit diantara epoch lainnya terhadap data *training*. Pada data *training*, sistem sudah mempelajari ciri dengan baik pada epoch 50 karena memiliki akurasi sebesar 100%

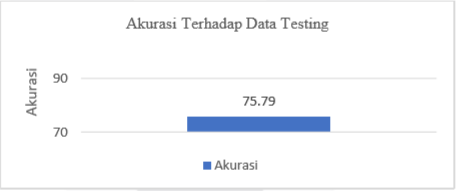
## Pengaruh Filter Terhadap Data Validasi



Gambar 4.5 Pengaruh Filter Terhadap Data Validasi

Berdasarkan gambar 4.5 sistem dengan filter 3 x 3 memiliki akurasi yang lebih tinggi daripada sistem dengan filter 5 x 5. Sistem dengan filter 5 x 5 memiliki tingkat akurasi optimal sebesar 86.71% yang terletak pada epoch 100. Sedangkan sistem dengan filter 3 x 3 memiliki tingkat akurasi optimal sebesar 89.73% yang terletak pada epoch 600.

## Validation



Gambar 4.6. Performansi Terhadap Data Sistem

pengujian pada gambar 4.4 dan 4.5 dilakukan bertujuan untuk mengetahui performansi sistem dengan parameter dan kombinasi terbaik hasil dari observasi skenario 1 dan 2 terhadap data baru, yaitu data testing. Didapatkan akurasi terhadap data testing sebesar 75,79%. Sehingga dapat disimpulkan algoritma *residual network* dapat meningkatkan akurasi pengenalan wajah sebesar 92%

BAB V  
KESIMPULAN DAN SARAN

## 5.1. Kesimpulan

1. Hasil perubahan laju pembelajaran terhadap parameter waktu komputasi adalah perubahan laju pembelajaran mempengaruhi waktu komputasi yang dapat dilihat dari laju pembelajara 0.0001 memerlukan waktu komputasi yang cukup lama dibandingkan dengan laju pembelajan 0.005. Waktu komputasi semakin cepat seiring dengan semakin melambatnya laju pembelajaran
2. Hasil validasi pengenalan wajah dengan merubah variasi jumlah data adalah jumlah data akan mempengaruhi akurasi pengenalan wajah yaitu pada nomor urut pengujian 2. dapat dilihat dari hasil pengujian yang dilakukan 8 kali. Pada saat *testing* berkurang secara otomatis data *training* akan meningkat. Peningkatan data *training* memberikan dampak akan semakin banyak data yang dilatih sehingga *resnet* dapat mempelajari data semakin baik.
3. Perbandingan hasil akurasi pada data testing. Penelitian sebelumnya menggunakan *Convolutionan neural network* mendapatkan akurasi pengenalan wajah sebesar 75,79 % dan menggunakan *Residual Network* mendapatkan akurasi pengenalan wajah sebesar 92% sehingga dapat disimpulkan *Algoritma Residual Network* dapat meningkatkan akurasi pengenalan wajah

## 5.2. Saran

1. Simulasi dari implementasi dapat dikembangkan menjadi sebuah aplikasi yang dapat digunakan di kehidupan nyata.
2. Kedalaman layer dapat dimaksimalkan sampe kedalaman 152 layer

# DAFTAR PUSTAKA

Abhirawan, H., Jondri, & Arifianto, A. (2017). Pengenalan Wajah Menggunakan *Convolutional Neural Networks* (*CNN*). *Universitas Telkom*, *4*(3), 4907–4916.

Sehman. (2015). Penerapan Face Recognition dengan Metode EigenFace pada Intelligent Car Security. *Seminar Nasional Inovasi dalam Desain dan Teknologi*, 342–348.

Indra. (2012). Sistem Pengenalan Wajah Dengan Metode Eigenface Untuk Absensi Pada PT Florindo Lestari. *Journal Pendidikan Teknologi-Informasi & Komunikasi Terapan*, *2012*(Semantik), 138.

Janocha, Katarzyna and Wojciech Marian Czarnecki. 2017. “On Loss Functions for Deep *Neural Network*s in Classification.” 1–10.

Faris (2018). *Implementasi Algoritma CNN resnet untuk pengenalan tanda tangan*. Tugas Akhir. Program Studi Teknik Informatika Universitas Bumigora Mataram.

Wani, M. A., Bhat, F. A., Afzal, S., & Khan, A. I. (2018). *Advances in Deep Learning on Graphs*.

Pytorch. (n.d.). torchvision.models — PyTorch master documentation. Retrieved July 19, 2019, from https://pytorch.org/docs/stable/torchvision/models.html?highlight=resnet

Qiong Cao, Shen, L., Weidi Xie, Parkhi, O. M., & Zisserman, A. (2018). VGGFace2: A dataset for recognising faces across pose and age, 826.

Ranjan, R., Castillo, C. D., & Chellappa, R. (2017). L 2 -constrained Softmax Loss for Discriminative Face Verification.

Schroff, F., Kalenichenko, D., & Philbin, J. (2015). FaceNet: A unified embedding for face recognition and clustering. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *07*-*12*-*June*, 815–823. https://doi.org/10.1109/CVPR.2015.7298682

Schroff, F., & Philbin, J. (n.d.). FaceNet : A Unified Embedding for Face Recognition and Clustering.

Sebastian Raschka. (2015). Single-Layer Neural Networks and Gradient Descent. Retrieved July 27, 2019, from https://sebastianraschka.com/Articles/2015\_singlelayer\_neurons.html

Skansi, S. (2018). *Introduction to deep learning: Part 1*. *Chemical Engineering Progress* (Vol. 114). https://doi.org/10.1007/978-3-319-73004-2

Soares, F., & Souza, A. M. F. (2016). *Neural network programming with Java : unleash the power of neural networks by implementing professional Java code*.

Udacity Course. (2018). Secure and Private AI Scholarship Challenge - Udacity. Retrieved July 27, 2019, from https://classroom.udacity.com/nanodegrees/nd185/parts/3fe1bb10-68d7-4d84-9c99-9539dedffad5/modules/28d685f0-0cb1-4f94-a8ea-2e16614ab421/lessons/d9869c40-de54-4395-9d5f-fa13c8254277/concepts/70526adf-40d3-4446-ac32-d3f798739745

Wani, M. A., Bhat, F. A., Afzal, S., & Khan, A. I. (2018). *Advances in Deep Learning on Graphs*.

Zhang, K., Zhang, Z., Li, Z., Member, S., Qiao, Y., & Member, S. (2016). Joint Face Detection and Alignment using Multi-ta, (1), 1–5. https://doi.org/10.1109/LSP.2016.2603342

## 

# LAMPIRAN

**Lampiran 1. Tabel *Learning Rate***

**Tabel learning rate 0.0001**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Akurasi** | **Validasi Akurasi** | **Kesalahan** | **Validasi Kesalahan** |
| 1 | 0.034 | 0.047 | 4.269 | 3.875 |
| 2 | 0.060 | 0.063 | 3.771 | 3.677 |
| 3 | 0.096 | 0.075 | 3.530 | 3.547 |
| 4 | 0.137 | 0.081 | 3.356 | 3.454 |
| 5 | 0.162 | 0.109 | 3.228 | 3.355 |
| 6 | 0.212 | 0.116 | 3.113 | 3.312 |
| 7 | 0.250 | 0.163 | 3.019 | 3.204 |
| 8 | 0.295 | 0.181 | 2.905 | 3.147 |
| 9 | 0.327 | 0.209 | 2.818 | 3.093 |
| 10 | 0.375 | 0.231 | 2.732 | 3.053 |
| 11 | 0.403 | 0.241 | 2.651 | 2.987 |
| 12 | 0.442 | 0.244 | 2.577 | 2.978 |
| 13 | 0.488 | 0.300 | 2.499 | 2.875 |
| 14 | 0.500 | 0.272 | 2.423 | 2.869 |
| 15 | 0.506 | 0.331 | 2.386 | 2.800 |
| 16 | 0.575 | 0.353 | 2.285 | 2.746 |
| 17 | 0.596 | 0.388 | 2.211 | 2.723 |
| 18 | 0.608 | 0.366 | 2.175 | 2.687 |
| 19 | 0.610 | 0.403 | 2.128 | 2.617 |
| 20 | 0.643 | 0.416 | 2.073 | 2.577 |
| 21 | 0.659 | 0.419 | 2.020 | 2.537 |
| 22 | 0.694 | 0.444 | 1.973 | 2.500 |
| 23 | 0.701 | 0.456 | 1.918 | 2.459 |
| 24 | 0.721 | 0.447 | 1.860 | 2.436 |
| 25 | 0.744 | 0.466 | 1.806 | 2.385 |
| 26 | 0.751 | 0.497 | 1.785 | 2.361 |
| 27 | 0.763 | 0.497 | 1.734 | 2.333 |
| 28 | 0.770 | 0.522 | 1.695 | 2.298 |
| 29 | 0.786 | 0.516 | 1.644 | 2.270 |
| 30 | 0.801 | 0.534 | 1.593 | 2.249 |
| 31 | 0.808 | 0.541 | 1.575 | 2.217 |
| 32 | 0.817 | 0.559 | 1.548 | 2.165 |
| 33 | 0.829 | 0.556 | 1.483 | 2.165 |
| 34 | 0.833 | 0.522 | 1.448 | 2.170 |
| 35 | 0.833 | 0.572 | 1.431 | 2.105 |
| 36 | 0.849 | 0.578 | 1.377 | 2.065 |
| 37 | 0.858 | 0.569 | 1.358 | 2.048 |
| 38 | 0.861 | 0.597 | 1.327 | 2.036 |
| 39 | 0.869 | 0.581 | 1.299 | 2.003 |
| 40 | 0.874 | 0.597 | 1.258 | 2.002 |
| 41 | 0.884 | 0.588 | 1.228 | 1.978 |
| 42 | 0.885 | 0.594 | 1.222 | 1.930 |
| 43 | 0.895 | 0.600 | 1.191 | 1.917 |
| 44 | 0.908 | 0.600 | 1.152 | 1.904 |
| 45 | 0.896 | 0.609 | 1.130 | 1.886 |
| 46 | 0.904 | 0.616 | 1.103 | 1.865 |
| 47 | 0.910 | 0.622 | 1.077 | 1.840 |
| 48 | 0.918 | 0.622 | 1.053 | 1.820 |
| 49 | 0.911 | 0.631 | 1.042 | 1.802 |
| 50 | 0.927 | 0.622 | 1.011 | 1.795 |

**Tabel learning rate 0.001**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Akurasi** | **Validasi Akurasi** | **Kesalahan** | **Validasi Kesalahan** |
| 1 | 0.074 | 0.072 | 3.683 | 3.576 |
| 2 | 0.281 | 0.234 | 2.887 | 2.864 |
| 3 | 0.477 | 0.416 | 2.333 | 2.463 |
| 4 | 0.633 | 0.456 | 1.893 | 2.242 |
| 5 | 0.738 | 0.578 | 1.535 | 1.948 |
| 6 | 0.814 | 0.653 | 1.265 | 1.736 |
| 7 | 0.865 | 0.628 | 1.055 | 1.659 |
| 8 | 0.898 | 0.684 | 0.879 | 1.565 |
| 9 | 0.926 | 0.741 | 0.713 | 1.313 |
| 10 | 0.947 | 0.759 | 0.604 | 1.258 |
| 11 | 0.959 | 0.756 | 0.511 | 1.263 |
| 12 | 0.967 | 0.766 | 0.446 | 1.129 |
| 13 | 0.970 | 0.784 | 0.397 | 1.038 |
| 14 | 0.977 | 0.816 | 0.331 | 0.998 |
| 15 | 0.981 | 0.816 | 0.281 | 0.957 |
| 16 | 0.981 | 0.803 | 0.252 | 0.914 |
| 17 | 0.984 | 0.834 | 0.225 | 0.883 |
| 18 | 0.984 | 0.822 | 0.208 | 0.875 |
| 19 | 0.986 | 0.816 | 0.186 | 0.866 |
| 20 | 0.987 | 0.812 | 0.172 | 0.858 |
| 21 | 0.986 | 0.828 | 0.162 | 0.810 |
| 22 | 0.989 | 0.828 | 0.148 | 0.800 |
| 23 | 0.988 | 0.828 | 0.136 | 0.784 |
| 24 | 0.988 | 0.819 | 0.122 | 0.774 |
| 25 | 0.989 | 0.822 | 0.119 | 0.771 |
| 26 | 0.991 | 0.822 | 0.106 | 0.754 |
| 27 | 0.988 | 0.825 | 0.102 | 0.754 |
| 28 | 0.988 | 0.831 | 0.098 | 0.727 |
| 29 | 0.991 | 0.828 | 0.089 | 0.738 |
| 30 | 0.986 | 0.828 | 0.093 | 0.719 |
| 31 | 0.993 | 0.831 | 0.080 | 0.736 |
| 32 | 0.991 | 0.828 | 0.081 | 0.711 |
| 33 | 0.990 | 0.825 | 0.077 | 0.712 |
| 34 | 0.991 | 0.831 | 0.077 | 0.726 |
| 35 | 0.991 | 0.837 | 0.071 | 0.688 |
| 36 | 0.993 | 0.837 | 0.069 | 0.683 |
| 37 | 0.992 | 0.837 | 0.066 | 0.687 |
| 38 | 0.991 | 0.828 | 0.063 | 0.703 |
| 39 | 0.991 | 0.825 | 0.065 | 0.694 |
| 40 | 0.991 | 0.825 | 0.060 | 0.702 |
| 41 | 0.992 | 0.828 | 0.058 | 0.713 |
| 42 | 0.991 | 0.831 | 0.059 | 0.665 |
| 43 | 0.992 | 0.831 | 0.053 | 0.690 |
| 44 | 0.991 | 0.834 | 0.054 | 0.659 |
| 45 | 0.994 | 0.828 | 0.051 | 0.665 |
| 46 | 0.993 | 0.841 | 0.051 | 0.655 |
| 47 | 0.993 | 0.837 | 0.050 | 0.651 |
| 48 | 0.992 | 0.834 | 0.052 | 0.687 |
| 49 | 0.992 | 0.834 | 0.051 | 0.655 |
| 50 | 0.993 | 0.834 | 0.048 | 0.671 |

**Tabel learning rate 0.01**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No** | **Akurasi** | **Validasi Akurasi** | **Kesalahan** | **Validasi Kesalahan** |
| 1 | 0.110 | 0.200 | 3.801 | 2.979 |
| 2 | 0.546 | 0.534 | 1.676 | 1.613 |
| 3 | 0.815 | 0.766 | 0.777 | 0.934 |
| 4 | 0.896 | 0.762 | 0.448 | 0.886 |
| 5 | 0.949 | 0.856 | 0.252 | 0.589 |
| 6 | 0.973 | 0.900 | 0.150 | 0.436 |
| 7 | 0.980 | 0.872 | 0.126 | 0.514 |
| 8 | 0.987 | 0.903 | 0.084 | 0.392 |
| 9 | 0.986 | 0.909 | 0.070 | 0.365 |
| 10 | 0.987 | 0.909 | 0.061 | 0.377 |
| 11 | 0.989 | 0.919 | 0.051 | 0.322 |
| 12 | 0.991 | 0.906 | 0.045 | 0.366 |
| 13 | 0.988 | 0.887 | 0.048 | 0.409 |
| 14 | 0.992 | 0.891 | 0.044 | 0.500 |
| 15 | 0.986 | 0.909 | 0.061 | 0.377 |
| 16 | 0.994 | 0.909 | 0.032 | 0.310 |
| 17 | 0.990 | 0.922 | 0.033 | 0.321 |
| 18 | 0.990 | 0.919 | 0.031 | 0.306 |
| 19 | 0.992 | 0.916 | 0.028 | 0.290 |
| 20 | 0.993 | 0.919 | 0.028 | 0.297 |
| 21 | 0.992 | 0.925 | 0.033 | 0.315 |
| 22 | 0.992 | 0.906 | 0.035 | 0.310 |
| 23 | 0.995 | 0.916 | 0.021 | 0.314 |
| 24 | 0.991 | 0.922 | 0.027 | 0.297 |
| 25 | 0.995 | 0.906 | 0.023 | 0.333 |
| 26 | 0.992 | 0.909 | 0.023 | 0.321 |
| 27 | 0.994 | 0.909 | 0.024 | 0.303 |
| 28 | 0.993 | 0.916 | 0.025 | 0.318 |
| 29 | 0.995 | 0.919 | 0.018 | 0.302 |
| 30 | 0.994 | 0.922 | 0.023 | 0.299 |
| 31 | 0.994 | 0.916 | 0.021 | 0.277 |
| 32 | 0.995 | 0.925 | 0.018 | 0.287 |
| 33 | 0.995 | 0.922 | 0.015 | 0.266 |
| 34 | 0.995 | 0.922 | 0.013 | 0.280 |
| 35 | 0.995 | 0.925 | 0.018 | 0.264 |
| 36 | 0.997 | 0.925 | 0.013 | 0.287 |
| 37 | 0.996 | 0.922 | 0.015 | 0.255 |
| 38 | 0.997 | 0.925 | 0.014 | 0.273 |
| 39 | 0.993 | 0.928 | 0.018 | 0.247 |
| 40 | 0.995 | 0.934 | 0.016 | 0.242 |
| 41 | 0.996 | 0.922 | 0.014 | 0.257 |
| 42 | 0.993 | 0.919 | 0.019 | 0.287 |
| 43 | 0.994 | 0.925 | 0.017 | 0.281 |
| 44 | 0.995 | 0.922 | 0.015 | 0.259 |
| 45 | 0.995 | 0.925 | 0.012 | 0.282 |
| 46 | 0.995 | 0.922 | 0.014 | 0.274 |
| 47 | 0.995 | 0.931 | 0.013 | 0.260 |
| 48 | 0.995 | 0.925 | 0.012 | 0.265 |
| 49 | 0.995 | 0.928 | 0.011 | 0.250 |
| 50 | 0.995 | 0.934 | 0.013 | 0.260 |

**Lampiran 2. *Source Code***

|  |
| --- |
| # import library *keras*  import *keras*  # import library open computer vision  import cv2  # import library komputasi numerik  import numpy as np  # import ilbrary pendukung open cv untuk keperluan manajemen data  from imutils import paths  from *keras*.layers import Activation, Flatten, Conv2D, MaxPooling2D, Dense  from *keras*.layers import AveragePooling2D, Input, Dropout  from *keras*.models import Sequential, Model  from *keras* import utils  from *keras*.models import load\_model  from sklearn.datasets import load\_files  from *keras*.utils import np\_utils  from glob import glob  from *keras* import applications  from *keras*.preprocessing.image import ImageDataGenerator  from *keras* import optimizers  from *keras*.models import Sequential,Model,load\_model  from *keras*.layers import Dense, Dropout, Flatten, Conv2D, MaxPool2D,GlobalAveragePooling2D  from *keras*.callbacks import TensorBoard,ReduceLROnPlateau,ModelCheckpoint  import os  # siapkan variabel kosong untuk menampung gambarnya  data = []  # siapkan variabel kosong untuk menampung label yang akan diprediksi  labels = []  # load dataset gambarnya dari path 'dataset/animals/'  dataset\_path = 'dataset/train/yale/'  # dataset\_path = 'dataset/train/'  imagePaths = sorted(list(paths.list\_images(dataset\_path)))  # masukan data gambar hasil load tadi satu persatu ke dalam variabel data yang kosong tadi  for imagePath in imagePaths:    #ubah ke grayscale  # baca gambarnya  img = cv2.imread(imagePath)  # img2 = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)      # ubah ukuran / dimensi gambarnya menjadi 32x32 px, serta convert ke dalam format vektor  img\_flat = cv2.resize(img, (32,32))    # simpan / tumpuk gambar yang sudah dibaca tadi  data.append(img\_flat)    # baca labelnya dengan melihat path foldernya  label = imagePath.split(os.path.sep)[-2]    # simpan / tumpuk label yang sudah dibaca tadi  labels.append(label)  # ubah data label tadi menjadi format array agar lebih memudahkan dalam komputasi  lbl = np.array(labels)  # ubah var data menjadi array numpy agar lebih memudahkan dalam komputasi  dt = np.array(data, dtype='float32')  # import library matplotlib untuk melihat gambarnya  import matplotlib.pyplot as plt  # # coba kita lihat sebuah gambar pada indeks ke-2 pada iamgePaths  gambar = imagePaths[0]  # baca gambarnya  gbr = plt.imread(gambar)  # tambahkan judul yang dilengkapi dengan nama file gambarnya + dimensi asli gambarnya  plt.title('gambar = '+str(gambar.split('/')[3])+' ; ukuran = '+str(gbr.shape))  # munculkan  plt.imshow(gbr)  # import library untuk memecah datasetnya menjadi bagian train dan test  from sklearn.model\_selection import train\_test\_split  # pecah datasetnya dengan jumlah data testing sebanyak 25% dari keseluruhan total data  x\_train, x\_test, y\_train, y\_test = train\_test\_split(dt, lbl,  test\_size=0.20,  random\_state=42)  # cek apakah sudah benar atau belum pemecaannya  print('x\_train : ',x\_train.shape)  print('x\_test : ', x\_test.shape)  print('y\_train : ', y\_train.shape)  print('y\_test : ', y\_test.shape)  # ubah tipe data gambarnya ke float  x\_train = x\_train.astype('float')  x\_test = x\_test.astype('float')  x\_train = x\_train.astype('float32')  x\_test = x\_test.astype('float32')  # normalisasi data gambarnya agar setiap nilai pixel memiliki rentang dari 0 sampai 1  x\_train = x\_train / 255  x\_test = x\_test / 255  # import library untuk melakukan one-hot-encoding dengan memanfaatkan method LabelBinarizer  from sklearn.preprocessing import LabelBinarizer  # membuat object construktur  lb = LabelBinarizer()  # memulai melakukan one-hot-encoding untuk y\_train dan y\_test  y\_train\_encoded = lb.fit\_transform(y\_train)  y\_test\_encoded = lb.transform(y\_test)  # cek nilainya  y\_test\_encoded  from *keras*.layers import BatchNormalization  import *TensorFlow* as tf  num\_class = 39  # inisialisasi input  masukan = Input(shape=(x\_train.shape[1:]))  # full pre-activation block  def pre\_activation\_block(inputan, filterin):  bn\_1 = BatchNormalization()(inputan)  relu\_1 = Activation('relu')(bn\_1)  weight\_1 = Conv2D(filterin, (3,3), padding = 'same')(relu\_1)  bn\_2 = BatchNormalization()(weight\_1)  relu\_2 = Activation('relu')(bn\_2)  weight\_2 = Conv2D(filterin, (3,3), padding = 'same')(relu\_2)  keluaran = weight\_2  addition = *keras*.layers.add([inputan, keluaran]) # shape[3] (inputan harus sama dengan keluaran)  return addition  # residual network  # kelomok 64  x = Conv2D(64, (7,7), input\_shape = x\_train.shape[1:], strides = (2,2), padding = 'same')(masukan)  pool = MaxPooling2D(pool\_size = (2,2))(x)  blok\_1\_1 = pre\_activation\_block(inputan = pool,filterin = pool.shape.as\_list()[3])  blok\_1\_2 = pre\_activation\_block(inputan = blok\_1\_1, filterin = blok\_1\_1.shape.as\_list()[3])  blok\_1\_3 = pre\_activation\_block(inputan = blok\_1\_2, filterin = blok\_1\_2.shape.as\_list()[3])  # kelomok 128  x1 = Conv2D(128, (3,3), strides = (2,2))(blok\_1\_3)  blok\_2\_1 = pre\_activation\_block(inputan = x1, filterin = x1.shape.as\_list()[3])  blok\_2\_2 = pre\_activation\_block(inputan = blok\_2\_1, filterin = blok\_2\_1.shape.as\_list()[3])  blok\_2\_3 = pre\_activation\_block(inputan = blok\_2\_2, filterin = blok\_2\_2.shape.as\_list()[3])  blok\_2\_4 = pre\_activation\_block(inputan = blok\_2\_3, filterin = blok\_2\_3.shape.as\_list()[3])  y = AveragePooling2D(pool\_size = (2,2))(blok\_2\_4)  luruskan\_niat = Flatten()(y)  output\_dense = Dense(num\_class)(luruskan\_niat)  aktivasi = Activation('softmax')(output\_dense)  keluaran = aktivasi  model\_*resnet* = Model(masukan, keluaran)  import time  class TimeHistory(*keras*.callbacks.Callback):  def on\_train\_begin(self, logs={}):  self.times = []  def on\_epoch\_begin(self, epoch, logs={}):  self.epoch\_time\_start = time.time()  def on\_epoch\_end(self, epoch, logs={}):  self.times.append(time.time() - self.epoch\_time\_start)  time\_callback = TimeHistory()  # model.fit(..., callbacks=[..., time\_callback],...)  # import library optimizer, sebagai contoh SGD optimizer  from *keras*.optimizers import SGD  # tentukan nilai learning-rate untuk optimizer yang digunakan  opt = SGD(lr=0.001)  # sgd = optimizers.SGD(lr=0.00, decay=1e-6, momentum=0.9, nesterov=True)  # compile model dan tentukan fungsi loss, optimizer dan metriks pengujian model yang akan dilatih  model\_*resnet*.compile(loss='categorical\_crossentropy', optimizer=opt, metrics=['accuracy'])  # memulai pelatihan  history = model\_*resnet*.fit(x\_train, y\_train\_encoded,  validation\_data=(x\_test, y\_test\_encoded),  callbacks=[time\_callback],  batch\_size=32,  epochs=50)  #KITA LIAT WAKTU YANG TEREKAM  times = time\_callback.times  print(times)  #Melakukan Prediksi  from sklearn.metrics import classification\_report  akurasi = model\_*resnet*.evaluate(x\_test, y\_test\_encoded)[1]  loss = model\_*resnet*.evaluate(x\_test, y\_test\_encoded)[0]  print()  print('akurasi : %.2f%%' % (akurasi))  print('loss : %.2f%%' % loss, '\n')  prediksi = model\_*resnet*.predict(x\_test)  print(classification\_report(y\_test\_encoded.argmax(axis = 1), prediksi.argmax(axis = 1), target\_names=lb.classes\_))  #menyimpan data ke csv  import csv  for i in loss:  a = list()  a.append(i)  with open("loss.csv", "a") as out\_file:  writer = csv.writer(out\_file)  writer.writerow(a)  out\_file.close() |