TUGAS AKHIR – MK 181810

SISTEM PEMANTAUAN MAHASISWA DALAM KELAS DENGAN *FACE RECOGNITION* BERBASIS *DEEP NEURAL NETWORK*

DEVIN AHMAD FEBRIAN
NRP. 02311540000071

Dosen Pembimbing:
Dr. rer. nat. Ir. Aulia M.T. Nasution, M.Sc
Andi Rahmadiansah, ST. MT.

DEPARTEMEN TEKNIK FISIKA
FAKULTAS TEKNOLOGI INDUSTRI DAN REKAYASA SISTEM
Institut Teknologi Sepuluh Nopember
Surabaya
2020

Halaman ini sengaja dikosongkan

FINAL PROJECT – MK 181810

STUDENT MONITORING SYSTEM IN CLASSROOM BY DEEP NEURAL NETWORK-BASED FACE RECOGNITION

DEVIN AHMAD FEBRIAN
NRP. 02311540000071

Supervisor:
Dr. rer. nat. Ir. Aulia M.T. Nasution, M.Sc
Andi Rahmadiansah, ST. MT.

DEPARTMENT OF ENGINEERING PHYSICS
FACULTY OF INDUSTRIAL TECHNOLOGY AND SYSTEMS ENGINEERING
Institut Teknologi Sepuluh Nopember
Surabaya
2020

Halaman ini sengaja dikosongkan

PERNYATAAN BEBAS PLAGIARISME

Saya yang bertandatangan di bawah ini:

Nama:	:Devin Ahmad Febrian
NRP	:02311540000071
Departemen/Prodi	:Teknik Fisika/ S1 Teknik Fisika
Fakultas	:Fakultas Teknologi Industri dan Rekayasa Sistem
Perguruan Tinggi	:Institut Teknologi Sepuluh Nopember

Dengan ini menyatakan bahwa Tugas Akhir dengan judul **"Sistem Pemantauan Mahasiswa Dalam Kelas dengan *Face Recognition* berbasis *Deep Neural Network*"** adalah benar karya saya sendiri dan bukan plagiat dari karya orang lain. Apabila di kemudian hari terbukti terdapat plagiat pada Tugas Akhir ini, maka saya bersedia menerima sanksi sesuai ketentuan yang berlaku.

Demikian surat pernyataan ini saya buat dengan sebenar-benarnya.

Surabaya, 16 Januari 2020
Yang membuat pernyataan,

Devin Ahmad Febrian
NRP. 02311540000071

Halaman ini sengaja dikosongkan

LEMBAR PENGESAHAN
SISTEM PEMANTAUAN MAHASISWA DALAM KELAS DENGAN *FACE RECOGNITION* BERBASIS *DEEP NEURAL NETWORK*

TUGAS AKHIR
Oleh:

Devin Ahmad Febrian
NRP. 02311540000071

Surabaya, 16 Januari 2020
Menyetuji,
Dosen Pembimbing I

Dr. rer. nat. Ir. Aulia M.T. Nasution, M.Sc
NIP. 19671117 199702 1 001

Menyetuji,
Dosen Pembimbing II

Andi Rahmadiansah, ST. MT.
NIP. 19790517 200312 1 002

Mengetahui,
Kepala Departemen
Teknik Fisika FTIRS-ITS

Dr. Suyanto ST. MT.
NIP. 19711113 199512 1 002

Halaman ini sengaja dikosongkan

LEMBAR PENGESAHAN
SISTEM PEMANTAUAN MAHASISWA DALAM KELAS DENGAN *FACE RECOGNITION* BERBASIS *DEEP NEURAL NETWORK*

TUGAS AKHIR
Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Teknik
pada
Bidang Studi Rekayasa Fotonika
Program Studi S1 Departemen Teknik Fisika
Fakultas Teknologi Industri dan Rekayasa Sistem
Institut Teknologi Sepuluh Nopember

Oleh:
DEVIN AHMAD FEBRIAN
NRP. 02311540000071

Disetujui Oleh Tim Penguji Tugas Akhir:

1. Dr. rer. nat. Ir. Aulia M.T. Nasution, M.Sc. ..... (Pembimbing I)
2. Andi Rahmadiansah, ST. MT. ..... (Pembimbing II)
3. Prof. Dr. Ir. Sekartedjo, M.Sc. ..... (Ketua Penguji)
4. Ir. Wiratno A.Asmoro, M.Sc ..... (Penguji I)
5. Iwan Cony S., ST, M.T ..... (Penguji II)

Surabaya

2020

Halaman ini sengaja dikosongkan

Sistem Pemantauan Mahasiswa Dalam Kelas dengan *Face Recognition* berbasis *Deep Neural Network*

Nama : Devin Ahmad Febrian
NRP :02311540000071
Departemen : Teknik Fisika
Pembimbing : Dr. rer. nat. Ir. Aulia M.T. Nasution, M.Sc
Andi Rahmadiansah, ST. MT.

Abstrak. Pemantauan mahasiswa di dalam kelas dapat dilakukan dengan mengenali mahasiswa menggunakan *Face Recognition Deep Neural Network* Sistem terdiri dari model yang akan memprediksi nama mahasiswa berdasarkan video yang diambil oleh kamera di dalam kelas dan komputer yang akan memproses gambar dan menjalankan prediksi model. Model akan dilatih dengan metode *Faster Region Convolutional Neural Network* dengan arsitektur *ResNet-50*. Model dilatih dan dites dengan dataset primer berupa gambar dari 6 wajah mahasiswa Teknik Fisika (TF) dan dataset sekunder berupa gambar wajah umum dari WIDER FACE. Model akan dilatih dengan WIDER FACE terlebih dahulu agar model bisa mempelajari bentuk wajah secara umum. Lalu model akan dilatih pada dataset mahasiswa TF dengan menggunakan metode *transfer learning.* Digunakan 908 gambar untuk pelatihan dan 375 gambar untuk tes dari dataset mahasiswa Teknik Fisika (TF). Dan digunakan 12.880 gambar untuk pelatihan dan 3.226 gambar untuk tes dari dataset WIDER FACE. Model ini mendapatkan mAP 96.52% dan waktu deteksi 1 sekon.

Kata kunci*: Face Recognition, Region Convolution Neural Network, ResNet-50, WIDER FACE*

Halaman ini sengaja dikosongkan

Student Monitoring System in Classroom By Deep Neural Network-Based Face Recognition

Name	: Devin Ahmad Febrian
NRP	:02311540000071
Departemen	: Teknik Fisika
Supervisor	: Dr. rer. nat. Ir. Aulia M.T. Nasution, M.Sc Andi Rahmadiansah, ST. MT.

Abstract. *A student monitoring system inside classroom could be done by using Deep Neural Network-based Face Recognition method. This system will consist of model that predict students name by video input taken from camera inside classroom, and computer to process the video and run inference for the model. The model is trained with Faster Region Convolutional Neural Network with ResNet-50 architecture. Training will use data from primary and secondary dataset. Primary dataset contain images of faces from six engineering physics department students. Secondary dataset is taken form WIDER FACE dataset containing images of faces in general. First, model is trained on WIDER FACE dataset to learn faces in general term. Then model will be trained on primary dataset with transfer learning method. Primary dataset use 908 images for training and 375 images for testing. Secondary dataset use 12.880 images for training and 3.226 images for testing. This model achive mAP score of 96.52% with 1 second detection time.*

Keywords: Face Recognition, Region Convolution Neural Network, ResNet-50, WIDER FACE

Halaman ini sengaja dikosongkan

KATA PENGANTAR

Puji syukur dipanjatkan pada Tuhan YME karena atas rahmatnya dapat terselesaikan Tugas Akhir dengan judul **"Sistem Pemantauan Mahasiswa Dalam Kelas dengan Face Recognition berbasis Deep Neural Network".**

Pembuatan tugas akhir ini mendapatkan bantuan dari berbagai pihak. Untuk itu diucapkan terimakasih kepada:

1. Orangtua penulis yang setia memberikan dukungan baik secara emosional maupun finansial selama penulis menjalan perkuliahan
2. Dr.rer.nat.Ir. Aulia M. T. Nasution M.Sc dan bapak Andi Rahmadiansah, ST. MT. selaku pembimbing Tugas Akhir penulis yang telah memberikan petunjuk dan arahan selama pengerjaan Tugas Akhir
3. Bapak Agus Muhamad Hatta S.T., M.Si., Ph.D. selaku Kepala Departemen Teknik Fisika ITS periode 2016-2019 dan bapak Dr. Suyanto, S.T., M.T. selaku Kepala Departemen Teknik Fisika ITS periode 2020-2023 atas bimbingan dalam segi akademis dan moril selama masa perkuliahan penulis
4. Prof. Dr. Ir. Sekartedjo, M.Sc. sebagai kepala Laboratorium Rekayasa Fotonika yang senantiasa membimbing penulis dalam masa pembuatan tugas akhir penulis.
5. Bapak Iwan Cony Setiadi yang membantu penulis dalam pengerjaan Tugas Akhir
6. Seluruh keluarga besar Laboratorium Rekayasa Fotonika yang telah memberikan dukungan moral dan menjadi tempat berbagi ilmu selama penulis berkuliah.

Sangat diharapkan kritik dan saran pada laporan tugas akhir ini. diharapkan laporan tugas akhir ini dapat menginspirasi,

membantu dan memberikan wawasan lebih bagi pembaca untuk melakukan penelitian lanjut.

Surabaya, 16 Januari 2020

Penulis

DAFTAR ISI

Halaman ini sengaja dikosongkan

DAFTAR GAMBAR

DAFTAR TABEL

Halaman ini sengaja dikosongkan

BAB I
PENDAHULUAN

1.1. Latar Belakang

Kemajuan teknologi digital diiringi munculnya revolusi industri yang dikenal dengan nama *Industry* 4.0, membuat berbagai aspek kehidupan dan pekerjaan semakin terintegrasi dengan sistem digital. Berbagai macam kegiatan bisa lebih mudah dan efisien dilakukan dengan pengimplementasian sistem digital. Saat komputer diperkenalkan ke dunia pada *Industry* 3.0, teknologi baru ini sangat mempengaruhi cara orang-orang bekerja pada saat itu. Teknologi komputer sangat berpengaruh sehingga sistem dalam bisnis hingga pemerintahan harus diubah demi mengikuti perkembangan teknologi komputer tersebut. Sekarang dengan *Industry 4.0* komputer sudah saling terhubung dan bisa bertukar informasi satu sama lain. Kompleksitas teknologi komputer saat ini sudah sangat tinggi hingga komputer bisa mengambil suatu keputusan tanpa campur tangan manusia. Kombinasi teknologi terkini seperti *Cyber-Physical System, Artificial Intelligence,* dan *Internet of Things* membentuk suatu sistem yang saling melengkapi dan berintegrasi, menyebabkan banyaknya tugas yang sebelumnya membutuhkan manusia bisa dikerjakan oleh komputer.

Salah satu teknologi *Industry 4.0* yaitu *Artificial Intelligence (AI)* bukanlah teknologi yang baru dikembangkan. Riset teknologi AI sudah dilakukan sejak tahun 1959 dalam bentuk komputer yang bisa bermain permainan *checkers* lebih baik dari manusia [1]. Pada jaman sekarang *hardware* dengan komputansi tinggi sudah memiliki harga yang terjangkau sehingga riset dalam bidang AI lebih mudah. Salah satu cara yang digunakan untuk membuat AI disebut dengan *machine learning* [2]. Komputer akan mempelajari data dengan model

statistik dan menggunakan model tersebut untuk mengerjakan berbagai tugas. Beberapa model klasik yang digunakan meliputi *Logistic Regression, Support Vector Machine (SVM), Genetic Algorithm, Decesion Tree,* dan *Artificial Neural Network.* Setiap model memiliki metode dan kelebihan yang berbeda-beda.

Salah satu model yang sedang populer sekarang adalah *Artificial Neural Network (ANN).* Model ini menyerupai bagaimana neuron pada otak manusia bekerja. Model ini bisa menyelesaikan masalah seperti *pattern recognition, prediction, optimization, associative memory,* dan *control* [3]. ANN sangat baik dalam menyelesaikan masalah tersebut, namun saat pertama kali dikembangkan perangkat keras yang menjalankan komputasi ANN pada saat itu tidak memiliki kekuatan komputasi yang cukup. Seiring teknologi perangkat komputasi semakin bagus dengan harga terjangkau, ANN kembali dikembangkan lagi. Salah peneliti yang mempopulerkan model ANN moderen yaitu Yann LeCun lewat papernya [4] pada tahun 1998.

Salah satu perkembangan model ANN akhir-akhir ini adalah *Deep Neural Network*. Model *Deep* ini memiliki jumlah neuron yang lebih banyak dari mode-model sebelumnya, sehingga memungkinkan komputer mengekstrak lebih banyak informasi. Semakin kaya informasi yang diambil, semakin kompleks juga tugas yang bisa dilakukan. Salah satu tugas ini yaitu *Computer Vision(CV)* berupa ekstraksi informasi yang dianggap penting dari suatu gambar. Model *Deep* ini bisa mengekstrak informasi dari pixel pada gambar, mencari pola tertentu, dan menggunakan pola tersebut untuk mengambil keputusan. Implementasi *Computer Vision* meliputi *Image Classification, Object Detection,* dan *Face Recognition.*

Pada penelitian ini akan diterapkan teknologi *Computer Vision* pada lingkungan kampus, terutama di dalam kelas. Akan dibuat sistem yang dapat mengenali wajah mahasiswa di dalam kelas dengan menggunakan *Deep Neural Network* untuk melakukan *Face Recognition*. Sistem akan memantau mahasiswa dan memunculkan nama mahasiswa yang dideteksi. Sistem berupa kamera yang dipasang di kelas untuk menangkap citra wajah dan komputer yang akan memproses citra wajah tersebut. Dengan menggunakan metode *Region Convolutional Neural Network,* akan ditentukan arsitektur yang cocok digunakan untuk sistem ini. Metode lain yang pernah digunakan adalah *Single Shot Detection (SSD).* Penelitian lain yang menggunakan metode SSD untuk mendeteksi wajah yaitu *Accurate Face Detection for High Performance* [38] dan *Single-Shot Scale-Aware Network for Real-Time Face Detection* [39] akan dibandingkan dengan metode yang ditawarkan oleh penelitian penulis. Perbandingan penelitian-penelitian ini dijelaskan lebih lanjut pada Bab 4 dan terlihat pada **Tabel 4**.

1.2. Rumusan Masalah

Rumusan masalah pada pembuatan tugas akhir ini meliputi:

a. Apa metode, arsitektur, dan resolusi video yang akan digunakan untuk sistem pemantau mahasiswa dalam kelas dengan *Face Recognition* berbasis *Deep Neural Network*?
b. Berapa waktu deteksi dan akurasi mAP *(mean average precision)* sistem pemantauan mahasiswa dalam kelas dengan *Face Recognition* berbasis *Deep Neural Network*?
c. Apakah sistem pemantauan mahasiswa dalam kelas dengan *Face Recognition* berbasis *Deep Neural Network* memiliki waktu deteksi yang cukup untuk

bisa dijalankan secara *real-time* dengan menggunakan video dengan *frame rate* 24 fps *(frane per second)*?

1.3. Batasan Penelitian

Adapun batas ruang lingkup dari penelitian ini antara lain:

a. Mahasiswa akan dibatasi yaitu 6 mahasiswa dalam 1 ruang kelas.
b. Mahasiswa yang akan dipantau adalah mahasiswa dari Departemen Teknik Fisika ITS.
c. Ruang kelas dalam keadaan pencahayaan standar.
d. Dilakukan simulasi kelas untuk membuat dataset primer.
e. Kamera yang digunakan adalah kamera belakang dari *handphone Samsung Galaxy S7* dengan tripod.
f. Digunakan dataset sekunder dari *WIDER FACE: A Face Detection Benchmark* yang bisa diakses dari url: http://shuoyang1213.me/WIDERFACE/ [5]
g. Pemilihan arsitektur dibatasi, yaitu: V*GG-16, Resnet50* .
h. Metode *Faster Region Convolution Neural Network (Faster R-CNN)* akan digunakan untuk mendeteksi wajah.
i. Pelatihan dan deteksi dilakukan dengan menggunakan Google Colab, salah satu layanan *cloud* milik Google

1.4. Tujuan

Berdasarkan perumusan masalah tersebut, tujuan dari penelitian ini adalah:

a. Menentukan metode, arsitektur dan resolusi video yang digunakan sistem pemantau mahasiswa dalam kelas dengan *Face Recognition* berbasis *Deep Neural Network*

b. Menentukan waktu deteksi dan akurasi mAP *(mean average precision)* sistem pemantauan mahasiswa dalam kelas dengan *Face Recognition* berbasis *Deep Neural Network*
c. Menentukan apakah sistem pemantauan mahasiswa dalam kelas dengan *Face Recognition* berbasis *Deep Neural Network* memiliki waktu deteksi yang cukup untuk bisa dijalankan secara *real-time* menggunakan video dengan *frame rate* 24 fps.

Halaman ini sengaja dikosongkan

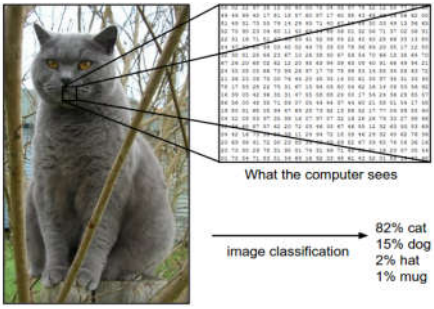
BAB 2
TEORI PENUNJANG

2.1. Image Classification

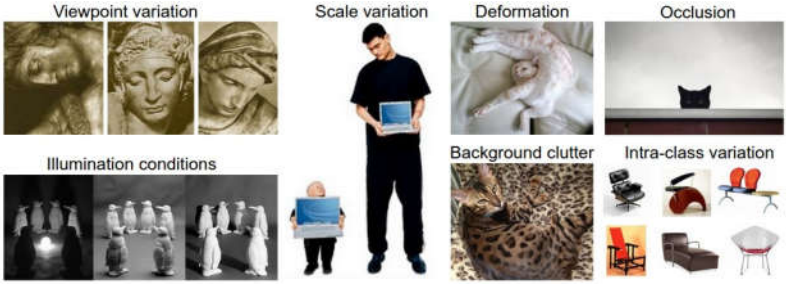
Image classification atau klasifikasi gambar adalah usaha komputer untuk memberikan label pada suatu gambar. Perspektif komputer melihat gambar berbeda dengan manusia. Bagi manusia, gambar memiliki berbagai komponen seperti warna dan pola. Namun untuk komputer, gambar hanyalah kumpulan angka dalam matriks yang besar. Sebuah model klasifikasi gambar harus mampu memberi label sebuah gambar hanya dengan menggunakan angka-angka tersebut.

Sebagai contoh sebuah model klasifikasi gambar diberi satu masukan gambar seperti yang terlihat pada **Gambar 2.1**. dan ditugaskan untuk memprediksi label gambar tersebut dari 4 kelas kemungkinan (kucing, anjing, tikus dan burung). Gambar ini adalah gambar kucing dengan dimensi 248 pixel kali 400 pixel. Gambar ini direpresentasikan dalam bentuk pixel RGB(*red, green, blue*) yang berarti pixel ini merepresentasikan 3 dimensi warna yang berbeda. Maka gambar ini terdiri dari 248 x 400 x 3 = 297,600 pixel. Dimana 1 pixel memilik range antara 0 – 255. Maka tugas model ini adalah merubah 297,600 angka ini menjadi 1 label yang benar, yaitu kucing.

Pembuatan model klasifikasi gambar memiliki banyak tantangan. Beberapa tantangan ini seperti sudut pandang, variasi skala objek, deformasi, dan variasi intra kelas. Seperti contoh yang digambarkan pada **Gambar 2.2.** beberapa gangguan atau distorsi pada gambar atau objek bisa mempersulit pemberian label pada gambar.



Gambar 2. 1. Proses klasifikasi gambar kucing **[6]**



Gambar 2. 2. Tantangan klasifikasi gambar **[6]**

Maka untuk membuat sebuah algoritma untuk mengklasifikasi gambar, kode tidak bisa ditulis secara eksplisit untuk mengenali gambar secara spesifik. Misalkan untuk mengklasifikasi gambar kucing, terlalu banyak variasi gambar yang bisa didefinisikan sebagai kucing. Sehingga untuk membuat model klasifikasi gambar akan digunakan metode yang sama saat kita mengajari anak kecil perbedaan antara kucing dan anjing. Komputer akan diberi sejumlah contoh gambar untuk label tersebut. Jika kita ingin membuat model yang bisa membedakan kucing dan anjing, maka komputer akan mendapatkan input berupa gambar kucing dan anjing. Metode ini disebut *data-driven approach. Dataset* adalah

sekumpulan gambar yang sudah diberi label terlebih dahulu sebelum diberikan ke komputer. *Dataset* tersebut seperti yang terlihat pada **Gambar 2.3.**

Gambar 2. 3. *Dataset* yang akan dipelajari komputer **[6]**

Terdapat 3 tahap dalam membuat model klasifikasi gambar:

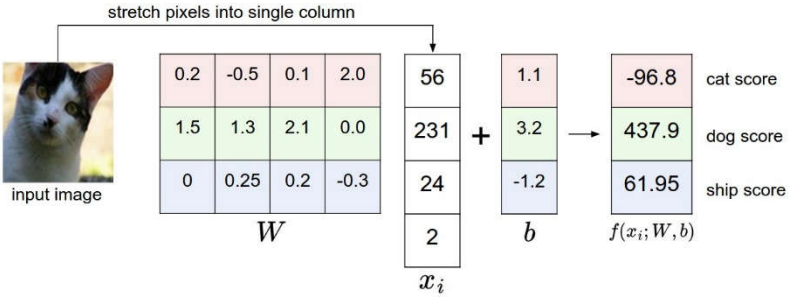
a. *Input*: Berupa *dataset* yang berisi sejumlah N gambar yang masing-masing diberi label dari sejumlah K kelas. Data ini juga disebut *training set.*
b. *Learning:* Pada tahap ini komputer akan mempelajari gambar setiap kelas.
c. *Evaluation:* Terakhir model akan diberi tugas untuk memprediksi label dari gambar yang komputer tersebut belum pernah lihat. Kemudian akan dibandingkan antara label gambar sebenarnya (*ground truth*) dengan label hasil prediksi model.

2.2. Linear Classifier

Terdapat berbagai macam model klasifikasi gambar yang sudah dikembangkan. Salah satunya adalah *Linear Classifier.* Model ini adalah fungsi linear yang memiliki input berupa pixel dari gambar dan output berupa *score* untuk prediksi label gambar tersebut.

$$\boldsymbol{f(x_i, W, b) = Wx_i + b} \quad \textbf{(2.1)}$$

Variabel x_i adalah matriks vector yang didapat dari total pixel pada gambar. Jika gambar memiliki bentuk 2x2x1(*greyscale*), maka matriks memiliki ukuran [4 x 1]. W adalah *weight* berupa matriks dengan ukuran [total pixel x jumlah kelas]. Jika model ingin memprediksi 3 kelas maka matriks *weight* akan berbentuk [4 x 3]. Variabel b adalah bias berupa matriks dengan ukuran [jumlah kelas x 1]. Maka pada contoh ini matriks bias memiliki ukuran [3 x 1]. Gambaran dari contoh fungsi linear ini bisa dilihat seperti pada **Gambar 2.4.**



Gambar 2. 4. Gambaran fungsi klasifikasi linear **[7]**

Score merepresentasikan prediksi model. *Score* yang tinggi menunjukan kecenderungan model untuk memberikan label gambar pada kelas tersebut. Pada contoh ini, model lebih cenderung memberi label anjing kepada gambar. *Score* kucing

sendiri memiliki *score* negatif yang menunjukan kelas kucing sangat kecil kemungkinannya untuk menjadi label gambar. Klasifikasi diatas sangatlah buruk karena prediksi model tidak mengikuti label gambar sebenarnya atau *ground truth.*

Untuk mengatasi masalah ini, harus ada variabel yang diubah untuk mengubah hasil *score* agar prediksi menjadi akurat. Variabel x diambil dari pixel gambar itu sendiri, maka variabel ini tidak bisa diubah. Variabel atau parameter yang bisa diubah untuk mempengaruhi *score* adalah W dan b. Maka masalah klasifikasi linear ini bisa dianggap sebagai masalah optimasi. Dimana kita menginkan *score* sebesar mungkin dengan mengatur nilai parameter W dan b.

Namun dalam praktik nilai bias tidak dimasukan dalam fungsi lagi. Nilai bias terkadang digunakan untuk mengkompensasi *dataset* yang tidak seimbang, dimana jumlah gambar antar kelas tidak seimbang. Bias juga bisa digabungkan ke variabel W, maka bentuk sederhana dari fungsi klasifikasi linear adalah sebagai berikut:

$$\boldsymbol{f(x_i, W) = Wx_i} \qquad \textbf{(2.2)}$$

2.3. Loss Function

Untuk mengoptimalkan hasil prediksi model, kita perlu mengukur seberapa buruk model dalam memprediksi suatu gambar. *Loss function* adalah fungsi yang akan menunjukan seberapa buruk model kita. Semakin tinggi nilai *loss* semakin buruk prediksi model kita.

Ada beberapa macam fungsi *loss* yang bisa digunakan. Salah satunya adalah *multiclas support vector machine (SVM) loss*. Fungsi ini bekerja dengan mengambil selisih *score*

tertinggi dengan *score* dari kelas sebenarnya. Hasil selisih kemudian dijumlahkan dengan nilai konstan Δ. Hasilnya merupakan nilai loss, kecuali jika hasilnya negatif maka loss dianggap nol. Dalam bentuk rumus SVM *loss* adalah seperti tertulis pada rumus **(2.3)**

$$\boldsymbol{L_i = \sum_{j \neq y_i} max\,(0, s_j - s_i + \Delta)} \qquad \textbf{(2.3)}$$

Variabel $\mathbf{s_j}$ adalah *score* dari masing-masing kelas yang diprediksi. Pada contoh di **Gambar 2.4.** karena kita memprediksi 3 kelas maka nilai s_j adalah -96.8, 437.9, dan 61.95. Variabel s_i adalah *score* dari label *ground truth.* Karena kita memprediksi gambar kucing *score* ini diambil dari label kucing yaitu -96.8. Variabel Δ bisa ditentukan oleh pengguna bergantung pada seberapa besar selisih yang diinginkan untuk mendapatkan *loss.*

L_i adalah *loss* untuk satu gambar. Untuk menghitung *loss* keseluruhan *dataset*, *loss* akan ditotal dan dibagi jumlah gambar N dalam *dataset,* seperti yang terlihat pada rumus **(2.4)**

$$\boldsymbol{L = \frac{1}{N} \sum L_i} \qquad \textbf{(2.4)}$$

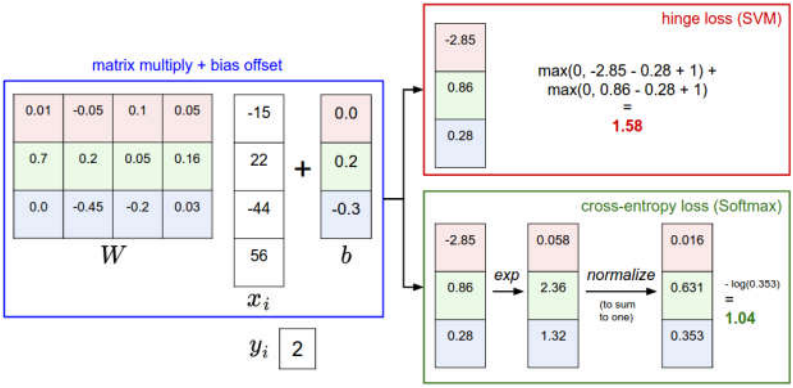
Dengan *loss* ini, model akan mendorong W agar memiliki nilai negatif. Sehingga menghasilkan *loss* nol dan prediksi yang akurat.

Selain SVM *loss, Softmax classifier* adalah fungsi *loss* lain yang lebih sering dijumpai pada model klasifikasi gambar. Ekspresi matematika fungsi *loss* tertulis pada rumus **(2.5)**

$$\boldsymbol{Li = -log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)} \qquad \textbf{(2.5)}$$

Fungsi *loss* ini menghitung probabilitas exponen *score* kelas *ground truth* s_{yi} dengan jumlah exponen *score* semua kelas s_j. Hasil probabilitas ini kemudian dikalikan dengan negatif log untuk menghasilkan *loss* untuk gambar tersebut.

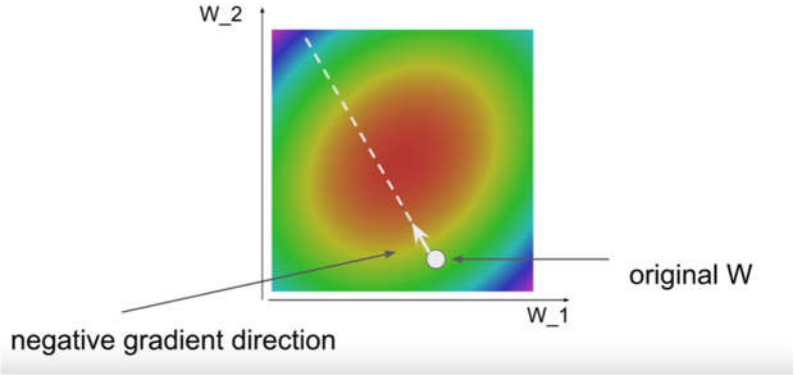
Pada SVM *loss,* hasil *loss* adalah diantara 0 sampai tak hingga. Sedangkan pada *softmax classifier* hasil *loss* berada diantara 0 hingga 1 karena dalam bentuk probabilitas. Pada **Gambar 2.5.** terlihat perbandingan dari kedua fungsi *loss:*



Gambar 2. 5. Perbandingan *SVM* dengan *Softmax* **[7]**

2.4. Optimization: Gradient Descent

Setelah didapatkan funsi *loss* kita bisa menentukan secara kuantitatif seberapa buruk performa model kita. Untuk memperbaiki performa, parameter W harus dioptimisasi untuk mendapatkan *loss* terendah. Fungsi *loss* bisa divisualisasikan dalam 2 dimensi seperti **Gambar 2.6.**



Gambar 2. 6. Visualisasi persebaran *loss* dalam 2 dimensi **[8]**

Sumbu x dan y diisi dengan variasi beban W1 dan W2. Warna merah melambangkan *loss* terendah. Semakin keluar dari pusat, *loss* akan semakin naik. *Gradient descent* adalah jenis optimasi yang menghitung gradient pada suatu titik beban W. Tujuan dari optimasi ini adalah untuk turun ke loss terendah dengan mengambil langkah kecil setiap kali gradient dihitung. Gradient sendiri menghitung kemiringan permukaan *loss.* Maka langkah titik W akan diambil mengarah ke kemiringan paling curam untuk mendapatkan *loss* terendah.

Gradient sendiri hanyalah turunan dari setiap dimensi input . Ekspresi gradien dari satu dimensi tertulis pada rumus **(2.6)**

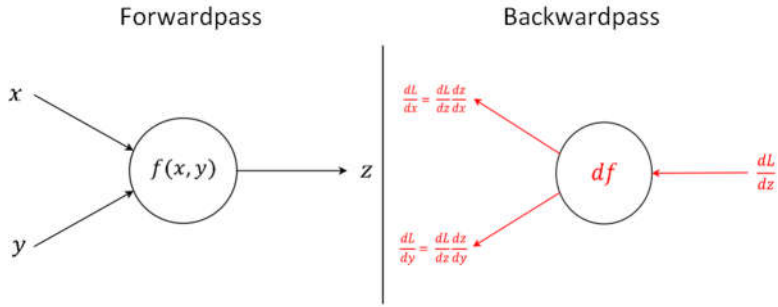
$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \qquad \textbf{(2.6)}$$

Jika input bersifat multi-dimensi maka gradien nya adalah turunan parsial dari setiap dimensi. Maka ekspresi untuk memperbarui beban W menuju loss terendah tertulis pada rumus **(2.7)**.

$$W_t = W_{t-1} - \alpha \nabla_W \quad (2.7)$$

$\mathbf{W_t}$ adalah beban yang telah diperbarui. $\mathbf{W_{t-1}}$ adalah beban sebelum optimasi. $\boldsymbol{\alpha}$ adalah yang disebut *learning rate,* salah satu *hyperparameter* penting dalam *deep neural network.* Parameter ini menentukan seberapa besar "langkah" yang diambil W menuju *loss* terendah. $\boldsymbol{\nabla_W}$ adalah arah yang harus diambil W hasil optimasi *gradient descent*. $\boldsymbol{\nabla_W}$ memiliki nilai negatif karena untuk mencapai *loss* terendah, gradient harus mengarah ke arah yang paling curam kebawah, maka digunakan negatif.

Metode menghitung gradient dalam *deep neural network* disebut juga sebagai *backpropagation* atau *backpass.* Metode ini menggunakan *chain rule* untuk menghitung gradient.



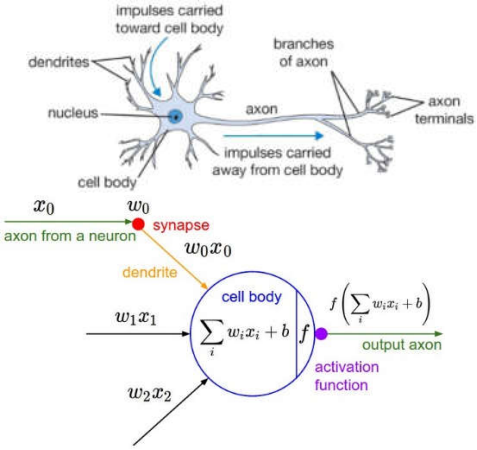
Gambar 2. 7. *Forwardpass* dan *backwardpass* **[9]**

Penerapan *forwardpass* dan *backpass* terlihat seperti pada **Gambar 2.7.** Tahap pertama dalam *deep neural network* adalah *forwardpass* dimana *score* dihitung berdasarkan input dan *loss* didapatkan dengan membandingkan *score* dengan *ground truth*. Tahap kedua yaitu *backpass* bertujuan untuk mencari gradient *loss* di akhir *neural network* terhadap variabel input di *neuron* sebelumnya. Variabel ini termasuk input x,

beban W, dan parameter lain yang digunakan. Tahap ini bekerja mundur dari output menuju input.

2.5. Neural Network

Neural network pada dasarnya adalah kumpulan *linear classifier* yang ditumpuk menjadi satu. Setiap fungsi klasifikasi linear berperan sebagai noda dalam jaringan neuron yang besar, layaknya jaringan saraf dalam otak manusia. Pada **Gambar 2.8.** terlihat perbandingan antara neuron pada otak dan saraf buatan.

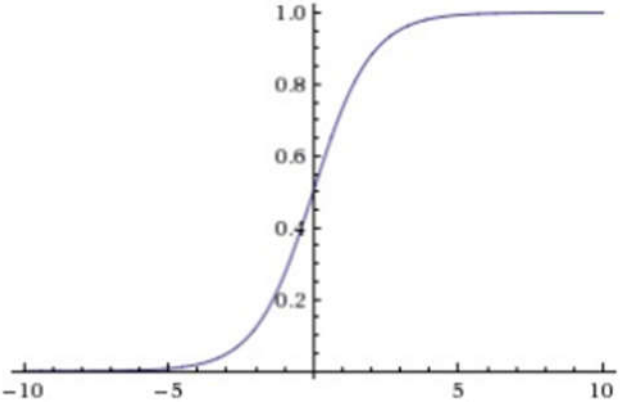


Gambar 2. 8. Perbandingan antara neuron pada otak dengan jaringan saraf buatan **[10]**

Dendrit berperan menerima sinyal yang dikirim dari neuron sebelumnya. Badan sel memproses sinyal tersebut dan jika neuron menjadi aktif, sinyal akan diteruskannya ke axon untuk dikirim ke neuron berikutnya.

Pada *neural network* fungsi klasifikasi linear menerima hasil proses fungsi linear sebelumnya seperti halnya badan sel menerima sel dari neuron sebelumnya. Yang menentukan apakah noda tersebut aktif atau tidak adalah *activation function* atau fungsi aktifasi. Fungsi ini berperan sebagai filter sinyal yang diterima noda. Ada beberapa jenis fungsi aktivasi yang sudah dikembangkan. Masing-masing memiliki kelebihan dan kekurangan:

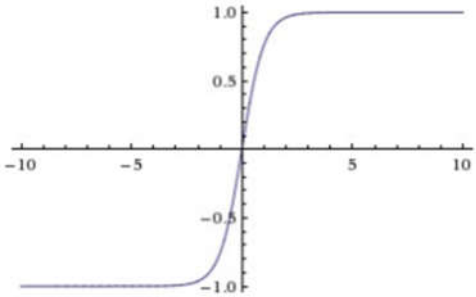
a. Sigmoid



Gambar 2. 9. Grafik fungsi sigmoid **[10]**

Fungsi sigmoid merubah sinyal masuk menjadi nilai di antara 0 hingga 1. Fungsi ini akan mensaturasi sinyal jika nilai input terlalu besar atau terlalu negatif. Fungsi ini sudah tidak digunakan lagi karena cenderung 'mematikan' noda. Jika nilai tersaturasi, gradien antar noda akan sangat kecil sehingga menghentikan proses belajar model. Bentuk grafik fungsi terlihat pada **Gambar 2.9.**

b. Tanh



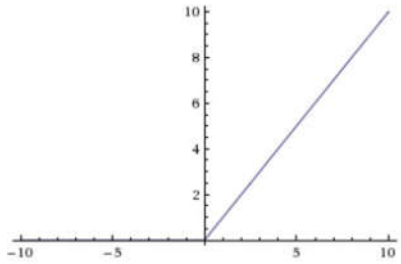
Gambar 2. 10. Grafik fungsi tanh **[10]**

Fungsi tanh atau hiperbolik tangen merubah sinyal masuk menjadi nilai di antara 1 dan -1. Keluaran dari fungsi ini bersifat *zero centered.* Data dengan sifat ini memiliki performa lebih baik dalam *neural network,* makan tanh lebih sering digunakan dari pada sigmoid. Namun masalah saturasi belum bisa diatasi. Bentuk grafik fungsi terlihat pada **Gambar 2.10.**

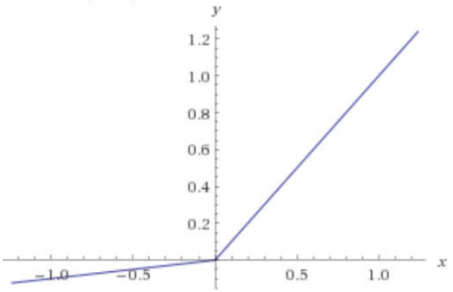
c. ReLU

Fungsi *ReLu* atau *rectified linear unit* membatasi input pada domain positif. Jika sinyal masuk memiliki nilai negatif, noda tidak akan aktif. Sinyal akan diteruskan secara linear jika bernilai positif. Fungsi ini sangat populer karena mempercepat *convergence* 6x lipat dibandingkan tanh [11]. Fungsi ini juga memiliki beban komputasi lebih rendah dibandingkan fungsi sebelumnya karena tidak memiliki operasi eksponen. Namun masalah saturasi masih ada pada bagian negatif fungsi. Bentuk grafik fungsi *ReLU* terlihat pada **Gambar 2.11.** Variasi lain *ReLU* yaitu *Leaky ReLU* memperbaiki masalah ini

dengan memberi sedikit toleransi terhadap input negatif. Bentuk grafik fungsi *Leaky ReLU* terlihat pada **Gambar 2.12.**



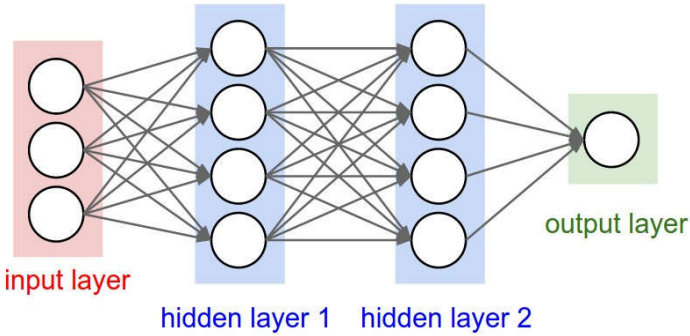
Gambar 2. 11. Grafik fungsi ReLU **[10]**



Gambar 2. 12. Grafik fungsi *Leaky ReLU* **[10]**

Susunan terkecil dari *neural network* adalah noda atau neuron. Noda ini kemudian dikumpulkan dalam grup bernama *layer* berdasarkan fungsinya. *Layer* ini akan kemudian disambungkan ke *layer* lain, membuat jaringan yang besar. Noda dalam *layer* tidak boleh terkoneksi kembali kedirinya

sendiri sehingga menyebabkan *loop*. Hal ini dikarenakan *neural network* membutuhkan fase linear yang dimulai dari input ke output yang disebut *forwardpass* dan sebaliknya yang disebut *backpass/backpropagation.*



Gambar 2. 13. *Neural network* dengan *2 fully-connected hidden layer* **[10]**

Salah satu tipe *layer* yang umum adalah *fully-connected layer.* Noda pada *layer* ini terkoneksi ke semua noda di *layer* setelahnya. Noda dalam layer juga tidak boleh terkoneksi ke noda di *layer* yang sama.

Variasi ukuran, jumlah, dan fungsi noda disebut dengan arsitektur. Semakin besar dan kompleks arsitektur, semakin banyak data yang bisa diolah. Namun hal ini tidak menjamin performa *neural netowork* untuk memprediksi menjadi lebih baik.

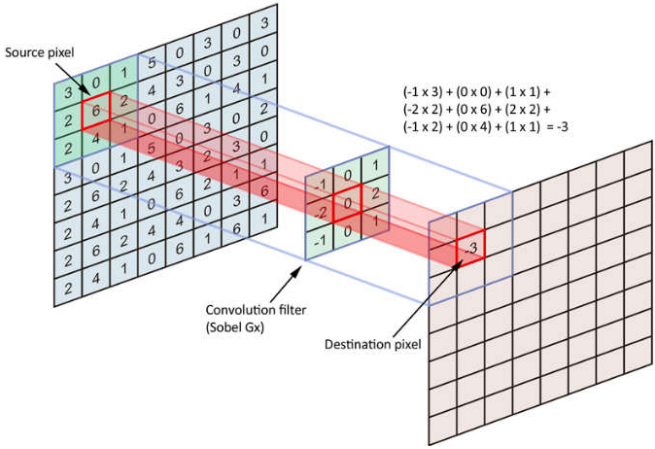
2.6. Convolutional Neural Network

Convolutional neural network atau bisa juga disebut *ConvNet* atau singkatnya CNN adalah jenis lain dari *neural network.* CNN adalah salah satu *neural network* yang digunakan untuk memproses gambar untuk membuat prediksi. Arsitektur CNN terdiri dari 3 jenis layer utama: *convolution layer, pooling layer,* dan *fully-connected layer.* Bentuk umum arsitektur CNN adalah [INPUT-CONV-RELU-POOL-FC]:

a. INPUT: Masukan memiliki bentuk HxWxD. H adalah *height* atau tinggi gambar W adalah *width* atau lebar gambar. Dan D adalah *depth* atau kedalaman dimensi gambar. RGB memiliki kedalaman 3 dan *greyscale* memiliki kedalam 1. Maka sebuah gambar berwarna RGB dengan tinggi dan lebar 32 pixel memiliki bentuk input 32x32x3=3072.
b. CONV: *Layer* ini menerapkan proses konvolusi terhadap gambar input. Konvolusi adalah proses penerapan filter terhadap gambar input. Filter ini dapat belajar dengan konsep yang sama dengan beban W di subbab sebelumnya. Filter ini memiliki dimensi spasial (tinggi dan lebar) yang lebih kecil dari gambar input, namun memiliki dimensi kedalaman *(depth/channel)* yang sama.
c. RELU: *Layer* ini akan berperan sebagai fungsi aktivasi
d. POOL: *Layer* ini berperan sebagai *dimensional reduction.* Dimensi spasial akan diperkecil sebelum dilanjutkan ke *layer* berikutnya.
e. FC: *Fully-connected layer* berperan memberi skor akhir hasil prediksi. Biasanya menggunakan *softmax classifier.*

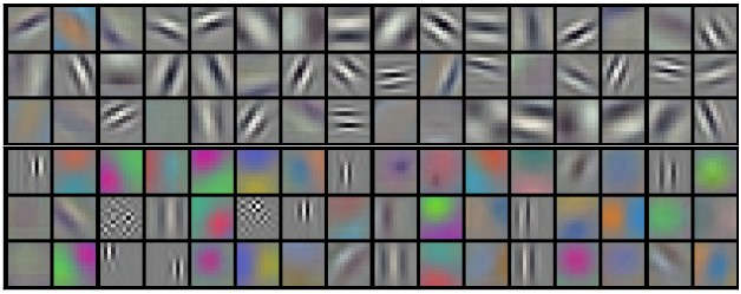
2.6.1. Convolution Layer

Layer ini adalah tulang belakang dari CNN karena lewat *layer* ini CNN mempelajari karakteristik dan pola suatu gambar. Cara kerja *layer* ini adalah dengan menerapkan proses konvolusi gambar dengan filter.



Gambar 2. 14. Proses konvolusi pada gambar **[12]**

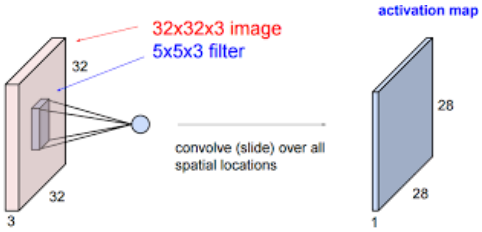
Seperti contoh pada **Gambar 2.14** akan dilakukan perkalian penjumlahan pixel pada gambar dengan filter konvolusi. Hasilnya akan membentuk gambar baru yang disebut dengan *activation map.* Filter ini digunakan untuk menemukan pola tertentu pada gambar seperti tepi atau perbedaan warna. Hasil konvolusi terlihat pada **Gambar 2.15.** Nilai pada filter bersifat dinamis dan akan terus diperbarui layaknya beban W lewat optimasi *gradient descent.* Jumlah filter yang dikonvolusikan ke gambar menunjukan variasi pola yang akan dipelajari oleh CNN.



Gambar 2. 15. Hasil konvolusi 96 filter berukuran [11x11x3] dari Krizhevky et al. **[11]** masing-masing filter mencari pola yang berbeda

Gambar yang melalui proses konvolusi akan mengalami *dimensional reduction* atau reduksi dimensi. Hal ini terjadi karena sifat alami konvolusi. Reduksi dimensi ini bisa diatur dengan 3 *hyperparameter* ini:

a. *Depth* atau dimensi kedalaman output diatur oleh banyaknya filter yang diterapkan. Jika input adalah gambar original dengan kedalaman 3 (RGB) dan dikonvolusikan dengan 96 filter maka hasil outputnya memiliki dimensi kedalaman 96.
b. *Stride* adalah seberapa besar filter bergeser untuk menerapkan konvolusi. Filter dengan *stride* 1 akan bergeser 1 pixel setiap konvolusi. Dengan *stride* 2, filter agar bergeser 2 pixel setiap konvolusi. Semakin besar *stride*, semakin besar reduksi dimensi spasial (tinggi dan lebar) yang dialami gambar.
c. *Zero Padding* adalah penambalan pixel yang hilang dari proses konvolusi. Pixel dengan nilai 0 akan dimasukan untuk menjaga agar dimensi spasial tidak berubah.

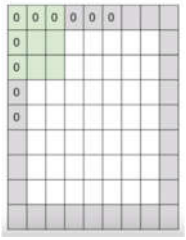


Gambar 2. 16. Proses konvolusi **[12]**

Contoh pada **Gambar 2.16** memiliki input gambar dengan dimensi 32x32x3 yang dikonvolusi dengan filter 5x5x3 dan *stride* 1. Hasil dari konvolusi adalah *activation map* dengan dimensi yang menyusut yaitu 28x28x1. Penyusutan dimensi *depth* disebabkan hanya digunakan 1 filter untuk konvolusi. Penyusutan dimensi spasial bisa diekspresikan dalam persamaan berikut:

$$\boldsymbol{Output\ size} = \frac{\boldsymbol{N-F}}{\boldsymbol{s}} + \mathbf{1} \qquad \textbf{(2.8)}$$

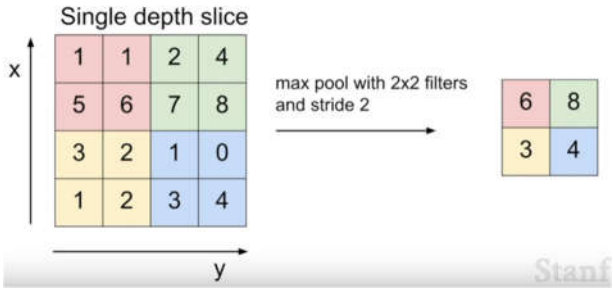
N adalah dimensi spasial gambar sebelum dikonvolusi. F adalah dimensi spasial filter. s adalah besar *stride.* Penyusutan dimensi spasial bisa diatasi dengan penerapan *zero padding* seperti pada **Gambar 2.17.** dengan memasukan pixel 0 di sisi terluar gambar.



Gambar 2. 17. *Zero padding* **[12]**

2.6.2. Pooling Layer

Layer ini berperan khusus untuk mengurangi dimensi spasial gambar. Semakin dalam *deep neural network,* dimensi gambar akan diperkecil untuk mengurangi parameter yang harus dihitung dan meringankan komputasi. *Max pooling* adalah salah satu tipe *pooling* yang umum. *Max pooling* bekerja mirip dengan konvolusi, namun nilai pixel tidak dikali dengan filter. Nilai pixel dalam area spasial filter tertinggi akan diambil dan ditempatkan di output. Contoh *max pooling* dengan ukuran 2x2 stride 2 terlihat pada **Gambar 2.18**.

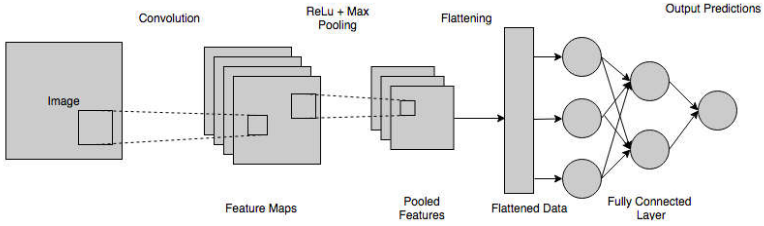


Gambar 2. 18. *Max pooling* **[12]**

2.6.3. Fully-connected Layer

Pada *layer* ini semua noda terkoneksi dengan noda di *layer* sebelumnya dan setelahnya. Pada *layer* sebelumnya, input memiliki dimensi dengan bentuk HxWxD(*height, width, depth*). Pada layer ini, input akan diratkan atau *flattened* menjadi satu kolom vektor yang panjang. Jika input memiliki dimensi 32x32x3, input akan diratakan menjadi 3072x1. 2 jenis *layer* sebelumnya bertugas untuk mengekstrak fitur atau *feature extraction* dari gambar. Fitur inilah yang mencirikan arti dari gambar input. Pada *layer* ini, fitur tersebut dialokasikan ke kelas yang ingin kita prediksi. Noda akhir pada

layer ini menentukan banyaknya kelas yang ingin kita prediksi. Rangkuman dari 3 jenis layer ini terlihat seperti pada **Gambar 2.19**.



Gambar 2. 19. Contoh *convolutional neural network* **[12]**

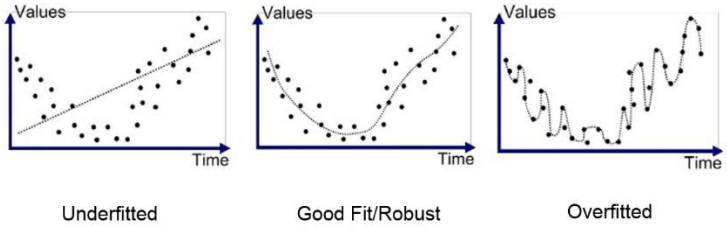
2.7. Training Neural Network

Dalam penggunaan *deep neural network,* sebuah model akan memiliki 2 fase. *Training* dan *inference. Training* adalah fase dimana model mempelajari suatu dataset. Proses pelatihan ini akan dilakukan dengan cara yang disebut *forwardpass* dan *backpass. Forwardpass* memasukan data kedalam *input layer* dan diteruskan hingga sampai *output layer* dimana prediksi akan dilakukan. *Backpass* akan bekerja untuk memperbarui beban W yang dimulai dari *output layer* hingga *input layer.* Satu tahap bolak-balik ini disebut dengan *epoch.* Setiap *epoch* model akan berusaha menurunkan *loss* dengan mengubah nilai beban W lewat optimasi. Model akan terus mengulangi fase ini hingga model mencapai *loss* yang diinginkan.

Pada fase *inference,* parameter beban W pada sudah diperbarui untuk mendapatkan prediksi seakurat mungkin. Untuk membuat prediksi, cukup lakukan *forwardpass* seperti pada fase *training.* Hasil akhir dari *forwardpass* adalah prediksi model terhadap input yang dimasukan.

Untuk melakukan *training* suatu model, banyak hal yang diperlukan untuk memastikan model dapat memprediksi input secara akurat. Seberapa bagus model dalam memprediksi data yang model tersebut belum pernah temui disebut dengan generalisasi [13]. Kebalikan dari generalisasi adalah *overfitting,* dimana model terlalu detil mempelajari dataset yang diberikan sehingga tidak bisa memprediksi data yang model belum pernah liat. Contoh berbagai performa model digambarkan pada **Gambar 2.20.** Pada contoh ini model diharapkan bisa memprediksi jalur data berdasarkan data yang diberikan. *Underfit* berarti model memiliki akurasi rendah dalam memprediksi data. *Overfit* berarti data terlalu "akurat" dalam memprediksi jalur dan hanya mengikuti data yang telah diberikan secara kaku. Berbagai macam performa model terlihat pada **Gambar 2.20.**

Underfit terjadi karena model tidak mendapat cukup data untuk dipelajari atau proses *training* terhambat sehingga model tidak bisa belajar dari dataset lagi. *Underfit* bisa diatasi dengan menambah jumlah data atau memilih jenis optimasi yang lebih baik.



Gambar 2. 20. Berbagai macam performa model **[14]**

Overfit terjadi saat model terlalu kompleks dalam mempelajari data yang diberikan. Untuk menurunkan kompleksitas model, bisa digunakan metode yang disebut dengan *regulatization.* Metode ini akan dijelaskan lebih lanjut pada subbab yang bersangkutan.

Dengan mempertimbakan berbagai macam performa yang bisa didapatkan model, berikut ini adalah metode dan cara yang penting dan perlu diperhatikan dalam fase *training* model. Fase ini akan dibagi 3: *pre-training, while training,* dan *post-training:*

a. *Pre-training: Activation function, data preprocessing, optimization, regularization*
b. *While training: babysitting the learning process, hyperparameter optimization*
c. *Post-training: loss curve*

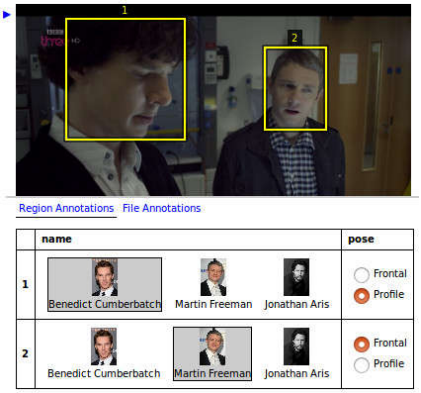
2.7.1. Activation Function

Fungsi aktivasi seperti yang sudah dijelaskan di subbab 2.5. adalah filter yang merubah sinyal dari noda sebelumnya menjadi nilai baru. Pemilihan fungsi aktivasi akan mempengaruhi dinamika data dalam model. Umumnya fungsi aktivasi ReLU sudah cukup untuk mendapat performa model yang baik. Pilihan fungsi aktivasi variasi dari ReLU adalah *Leaky ReLu* dan *Exponential Linear Unit* (ELU).

2.7.2. Data Preprocessing

Metode ini adalah pemrosesan data yang akan digunakan sebelum dimasukan ke dalam model. Proses akan berbeda bergantung pada jenis data yang digunakan. Untuk data berupa gambar, data akan melalui proses anotasi, perubahan resolusi dan *zero-centering.*

Anotasi adalah tahap pemberian label pada gambar. Pemberian label pada gambar ditujukan untuk menandakan kelas apa gambar tersebut termasuk. Pemberian label bisa dilakukan dengan cara pemberian nama *file* gambar seperti 'kucing_1.jpg' atau dengan menggunakan *software tool* tertentu seperti *VIA annotation tool* seperti pada **Gambar 2.21.**



Gambar 2. 21. Anotasi dengan *VIA annotation tool* **[15]**

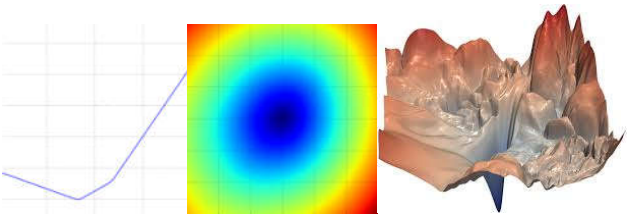
Perubahan resolusi bertujuan untuk menseragamkan resolusi seluruh gambar dalam dataset. Model pada umumnya hanya menerima resolusi tertentu seperti 224x224x3 (VGGNet)

2.7.3. Optimization

Selain *gradient descent,* terdapat banyak jenis metode optimasi lain yang sudah dikembangkan. Metode optimasi ini berupa modifikasi dari metode *gradient descent* yang ditujukan untuk membenahi beberapa kelemahan pada metode tersebut.

Kelemahan *gradient descent* bisa digambarkan lewat *loss landscape. Loss landscape* [8] adalah gambaran persebaran parameter beban W beserta *loss* yang dihasilkan

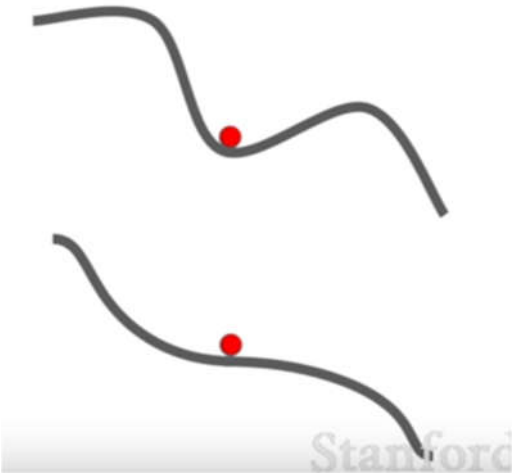
parameter tersebut. Tujuan dari optimasi adalah memandu koordinat beban W yang diletakan secara acak pada *loss lanscape* untuk mencapai titik terendah atau *global minima* dari *loss landscape.* Pada **Gambar 2.22.** terlihat berbagai penggambaran *loss lanscape* dalam 1,2, dan 3 dimensi. Warna biru menunjukan *loss* yang rendah dan merah menunjukan *loss* yang tinggi. Perlu diingat dalam praktik, *loss landscape* akan sulit untuk digambarkan karena sifat *neural network* yang memiliki parameter beban W multidimensi. Dimensi *loss landscape* proporsional dengan jumlah parameter yang dilatih pada model *neural network.*



Gambar 2. 22. Penggambaran *loss landscape* dalam berbagai dimensi. Kiri: satu dimensi, Tengah: dua dimensi. Kanan: tiga dimensi **[8]**

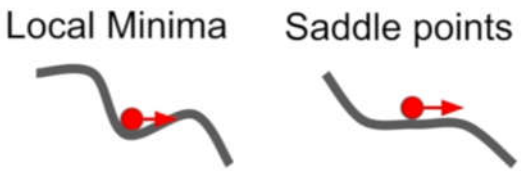
Pada *loss landscape,* terdapat beberapa titik yang membuat pembelajaran model terhambat. Titik ini disebut *local minima* dan *saddle points.* Pada *local minima,* model tidak bisa belajar lagi karena secara lokal sudah tidak ada penurunan gradien. Sehingga model "tersangkut" pada *local minima* sedangkan ada kemungkinan masih ada titik yang memiliki *loss* lebih rendah. Pada *saddle points,* model akan sulit belajar karena gradien yang sangat kecil. Permukaan *loss landscape* yang landai menyebabkan kemiringan yang sangat kecil sehingga pembelajaran model membutuhkan waktu yang

lebih lama. Visualisasi *local minima* dan *saddle point* terlihat pada **Gambar 2.23.**



Gambar 2. 23. Atas: *local minima.* Bawah: *saddle points* dalam *loss lanscape* satu dimensi. **[16]**

Untuk mengatasi masalah ini, digunakan metode *momentum* untuk membantu model keluar dari *local minima* dan *saddle points* [17]. *Momentum* bekerja seperti pada hukum fisika, dimana kecepatan akan terakumulasi seiring model turun mencari *loss* yang lebih rendah. Akumulasi kecepatan ini akan digunakan untuk membantu model mengatasi *local minima* dan *saddle points.* Visualisasi penyelesaian masalah ini terlihat pada **Gambar 2.24.**



Gambar 2. 24. *Momentum* mengatasi model terjebak di *local minima* dan *saddle points* **[16]**

Perbandingan ekspresi matematika antara *gradient descent* dengan *momentum* bisa dilihat seperti dibawah:

Gradient descent:

$$\boldsymbol{W_t = W_{t-1} - \alpha \nabla_W} \quad \textbf{(2.7)}$$

Momentum:

$$\boldsymbol{v_t = \rho v_{t-1} + \nabla_W} \quad \textbf{(2.9)}$$

$$\boldsymbol{W_t = W_{t-1} - \alpha v_t} \quad \textbf{(2.10)}$$

Pembaruan beban W pada metode *momentum* dibagi 2 tahap. Pertama $\boldsymbol{v_t}$ kecepatan akan dihitung berdasarkan kecepatan sebelumnya $\boldsymbol{v_{t-1}}$ ditambah dengan $\boldsymbol{\nabla_W}$ (arah yang harus diambil W hasil optimasi *gradient descent).* $\boldsymbol{\rho}$ adalah friksi yang digunakan untuk mengendalikan akumulasi kecepatan seiring beban W diperbarui.

Kemudian tahap kedua adalah pembaruan beban W. Pembaruan mengambil beban W lama dan menjumlahkannya dengan kecepatan $\boldsymbol{v_t}$ yang sudah terakumulasi dari kecepatan sebelumnya. $\boldsymbol{v_t}$ memiliki arah turun ke *loss* terendah, maka variabel ini memiliki notasi negatif. *Learning rate* $\boldsymbol{\alpha}$ mengatur seberapa besar "langkah" yang diambil oleh model.

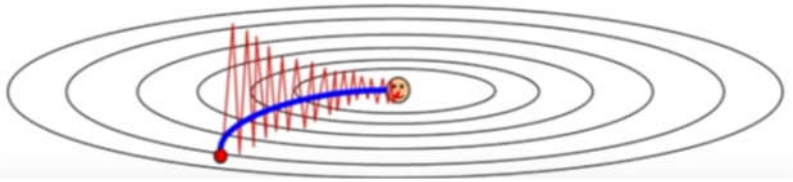
AdaGrad adalah metode lain yang mengakumulasi gradient kuadrat selama pelatihan [17]. Kemudian akmulasi gradient tersebut akan digunakan untuk membantu pembaruan beban W.

$$\boldsymbol{G_t = G_{t-1} + \nabla_w^2} \qquad \textbf{(2.11)}$$

$$\boldsymbol{W_t = W_{t-1} - \frac{\alpha \nabla_W}{\sqrt{G}+10^{-7}}} \qquad \textbf{(2.12)}$$

Gradient sendiri adalah vektor dengan dimensi sejumlah parameter beban W. Dengan membagi akumulasi gradient kuadrat, *AdaGrad* membantu model untuk mempercepat ke arah dimensi yang memiliki gradient kecil dan memperlambat ke arah dimensi yang memiliki gradient besar. $\mathbf{G_t}$ adalah akumulasi dari gradient kuadrat dari pembaruan beban W sebelumnya. 10^{-7} digunakan untuk memastikan persamaan tidak membagi dengan 0. AdaGrad mencegah model untuk "*overshoot"* saat memperbarui beban W. Dengan ini, waktu pelatihan model akan lebih cepat. Seperti yang terlihat pada **Gambar 2.25.** model memperbarui beban ke arah tengah dimana *loss* terendah. *AdaGrad* (biru) menempuh jarak terdekat dan tercepat ke tengah dibandingkan tanpa *AdaGrad* (merah) yang menghabiskan waktu dengan menempuh jalur yang lebih jauh.

Namun terdapat masalah pada *AdaGrad.* Akumulasi gradient kuadrat dapat menyebabkan pembaruan beban W diluar kendali, menyebabkan pelatihan model yang sangat lama hingga tidak tercapainya *loss* yang rendah. Salah satu metode yang membenahi masalah ini adalah *RMSProp* [17].



Gambar 2. 25. Pembaruan beban W sebelum (merah) dan sesudah (biru) AdaGrad. **[16]**

RMSProp menambahkan parameter *decay* pada akumulasi gradient kuadrat untuk mengontrol akumulasi gradient yang terjadi. Berikut ekspresi matematika dari metode tersebut:

$$\boldsymbol{G_t = \beta G_{t-1} + (1-\beta)\nabla_w^2} \qquad \textbf{(2.13)}$$

$$\boldsymbol{W_t = W_{t-1} - \frac{\alpha \nabla_W}{\sqrt{G_t + 10^{-7}}}} \qquad \textbf{(2.12)}$$

Tahap metode *RMSProp* mirip dengan *AdaGrad*. Pada tahap akumulasi, gradient kuadrat akan dikontrol dengan parameter *decay* $\boldsymbol{\beta}$. Parameter ini akan membatasi akumulasi gradient yang terjadi. Tahap pembaruan beban W sama dengan metode *AdaGrad.*

Metode berikutnya adalah salah satu metode optimasi *neural network* yang populer saat ini, yaitu *Adam* atau *adaptive momentum* [17]. Metode ini menggambungkan *momentum* dan *RMSProp. Momentum* mempercepat pelatihan dengan akumulasi kecepatan dan *RMSProp* mengontrol jalur pelatihan model untuk menghindari *overshoot*. Perbandingan *Adam* dengan metode optimasi lain terlihat pada **Gambar 2.26.** Berikut ekspresi dari metode ini:

$$\boldsymbol{v_t = \beta_1 v_{t-1} + (1-\beta_1)\nabla_w} \qquad \textbf{(2.14)}$$

$$G_t = \beta_2 G_{t-1} + (1 - \beta_2)\nabla_w^2 \quad \textbf{(2.15)}$$

$$W_t = W_{t-1} - \frac{\alpha v_t}{\sqrt{G_t} + 10^{-7}} \quad \textbf{(2.16)}$$

Ekspresi (2.14) adalah bagian *momentum* dan ekspresi (2.15) adalah bagian *RMSProp.* Parameter *decay* β_1 dan β_2 berguna untuk mengontrol akumulasi v_t dan G_t.

2.7.4. Regularization

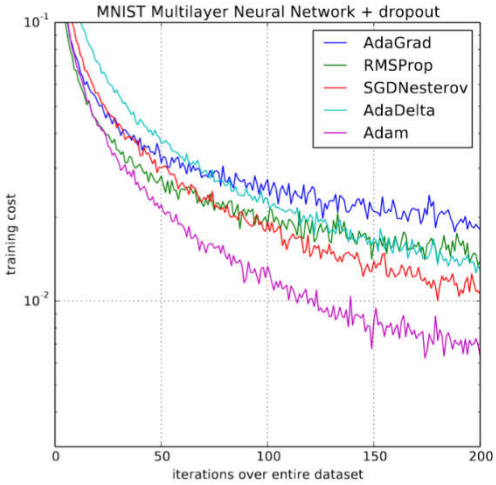
Regularization adalah metode untuk mengurangi *overfitting.* Metode ini mengurangi ketergantungan model terhadap data yang digunakan untuk latihan dan meningkatkan generalisasi model terhadap data yang model belum pernah liat.

Dropout adalah salah satu metode *regularization. Dropout* bekerja dengan memutus beberapa koneksi antara noda secara acak pada *neural network.* Pemutusan koneksi noda ini bertujuan untuk mengurangi ketergantungan model terhadap beberapa noda. *Dropout* hanya digunakan pada tahap *training,* tidak pada tahap *test* dan *inference.* Model yang menggunakan *dropout* terlihat pada **Gambar 2.27.**

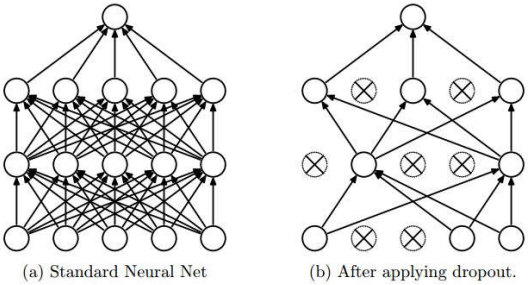
Metode *regularization* lain adalah *data augmentation.* Metode ini memodifikasi dataset input yang akan digunakan untuk latihan. Modifikasi ini dilakukan untuk membuat variasi pada kelas dan memperbanyak dataset yang bisa dipelajari oleh model. Banyak modifikasi yang bisa dilakukan untuk dataset gambar, berikut beberapa contohnya:

a. *Translate*
b. *Rotate*
c. *Shear*
d. *Collor jitter*
e. *Random crop and scales*

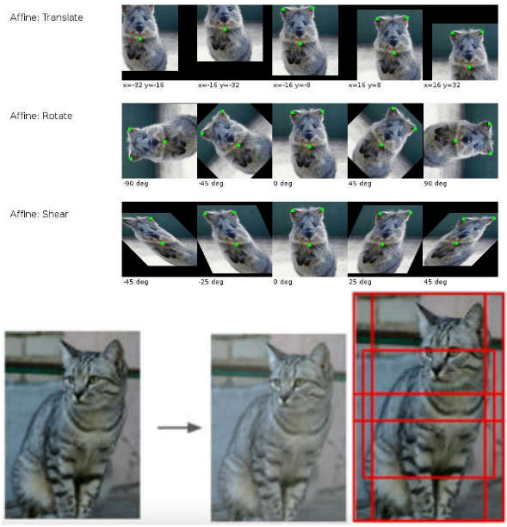
Contoh gambar yang melalui modifikasi terlihat pada **Gambar 2.28.**



Gambar 2. 26. Perbandingan performa optimasi Adam dengan metode optimasi lain **[18]**.



Gambar 2. 27. Model dengan penerapan *dropout* **[19]**



Gambar 2. 28. *Data augmentation* pada data gambar. Atas: *translate, rotate, shear.* Kiri: *collor jitter.* Kanan: *random crop* **[16]**

2.7.5. Babysitting The Learning Process

Babysitting adalah metode pengawasan model saat *training.* Pengawasan ini bertujuan agar model bisa mempelajari dataset secara maksimal dan menghasilkan *score* prediksi yang memuaskan. Berikut tahap-tahap dalam *babysitting:*

a. Pilih Arsitektur: Terdapat banyak arsitektur yang sudah dikembangkan seperti AlexNet, VGGNet, dan ResNet. Model bisa dibangun menggunakan aristektur tersebut atau dengan membuat arsitektur baru.
b. Preprocessing data: Data akan diproses agar ideal untuk pelatihan model

c. Lakukan *forwardpass*: Tahap ini dilakukan untuk mendapatkan *original loss,* yaitu *loss* yang didapatkan sebelum penerapan *regularization.*
d. Tambah *regularization*: Tahap ini dilakukan untuk melihat efek *regularization* terhadap *loss.* Jika *loss* bertambah, maka *regularization* berjalan dengan baik.
e. Mulai *training* dengan dataset kecil: *Training* dengan dataset yang kecil akan memastikan *overfit* terjadi pada model. Ini menandakan model dapat mempelajari data dengan baik.
f. Mulai *training* dengan dataset utuh: Mulai dengan nilai *regularization* yang kecil. Pada tahap ini *hyperparameter learning rate* akan diatur sedemikian rupa agar model mendapatkan *loss* terendah. Jika *loss* turun terlalu lambat dari *original loss,* maka *learning rate* terlalu kecil. Jika *loss* turun terlalu cepat atau kode mendapat error NaN yang berarti *loss explode* maka *learning rate* terlalu besar.

2.7.6. Hyperparameter Optimization

Hyperparameter adalah variabel yang mengatur bagaimana suatu model *neural network* dilatih. *Hyperparameter* ditetapkan sebelum pelatihan model dimulai. Terdapat banyak macam *hyperparameter* yang digunakan yang bergantung pada jenis arsitektur dan metode optimasi yang dipilih. *Learning rate* adalah salah satu *hyperparameter* yang selalu digunakan di berbagai macam arsitektur dan metode optimasi.

Ada beberapa macam metode untuk melakukan *hyperparameter optimization,* yaitu:

a. Manual: Pemilihan nilai *hyperparameter* berdasarkan intuisi dan pengalaman. Model dilatih

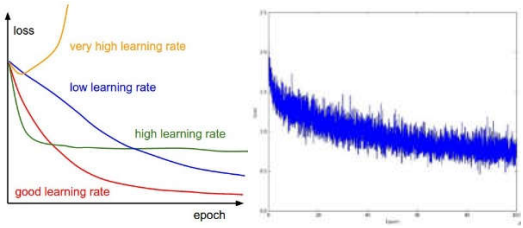
dengan berbagai nilai *hyperparameter* dan skor hasil tes akan dibandingkan. Tahap ini akan diulang hingga mendapatkan hasil yang memuaskan.

b. Grid search: Kombinasi nilai *hyperparameter* akan disiapkan dalam bentuk *grid.* Kemudian model dilatih dengan menggunakan semua kombinasi nilai *hyperparameter* tersebut. Metode ini membutuhkan waktu yang lama karena semua kombinasi nilai akan dicoba untuk menentukan skor tertinggi.
c. Random search: Kombinasi nilai *hyperparamer* akan dipilih secara acak dan akan digunakan untuk melatih model.

2.7.7. Lost Curve

Setiap akhir *training,* terdapat beberapa variabel yang bisa digunakan sebagai panduan yang memperlihatkan performa model selama *training.* Salah satu variabel tersebut adalah *loss curve.*

Loss curve menandakan *loss* yang didapatkan model setiap *epoch* atau iterasi *training.* Seperti yang terlihat pada **Gambar 2.29.** *loss curve* bisa memperlihatkan kondisi *learning rate* yang di tetapkan untuk pelatihan model. Sehingga *loss curve* bisa digunakan untuk mengoptimasi nilai *learning rate.*

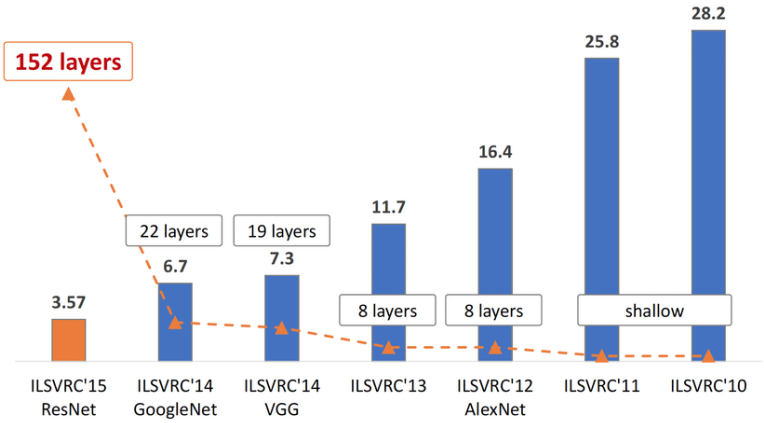


Gambar 2. 29. *Loss curve* **[17]**

2.8. Neural Network Architecture

Arsitektur menentukan susunan noda, jumlah noda, jumlah *layer,* dan konfigurasi sambungan antar *layer* dalam suatu *neural network.* Arsitektur dari model *neural network* menentukan bagaimana sebuah data diproses dan digunakan untuk melatih suatu model. Terdapat berbagai macam jenis arsitektur dengan berbagai bentuk dan kegunaan. Berikut beberapa macam arsitektur yang sudah dikembangkan untuk memproses gambar menggunakan *convolutional neural network:*

a. AlexNet
b. VGG
c. GoogleNet
d. ResNet



Gambar 2. 30. Arsitektur pemenang dalam *ImageNet Large Scale Visual Recognition Challange* **[20]**

Arsitektur tersebut adalah pemenang dari *ImageNet Large Scale Visual Recognition Challange.* ILSVRC adalah

tolak ukur dalam klasifikasi dan deteksi objek pada gambar yang telah digelar sejak 2010. Perbandingan antar arsitektur dalam kompetisi *ImageNet* terlihat pada **Gambar 2.30.** Kompetisi ini menantang peserta untuk memprediksi ratusan kategori kelas dari jutaan gambar yang dikumpulkan dalam dataset *ImageNet.* Kompetisi ini menghasilkan berbagai inovasi baru dalam dunia klasifikasi objek yang sangat penting untuk pengembangan berikutnya. *Deep neural network* pertama kali digunakan pada 2012 oleh arsitektur AlexNet. Kemudian pada 2014 lewat arsitektur VGG konsep CNN diperdalam dengan menambahkan jumlah *layer* yang digunakan. Ditahun yang sama arsitektur GoogLeNet mengenalkan konsep *inception module* yang berbeda dengan metode CNN pada umumnya. Dan akhirnya pada 2015 ResNet berhasil memecahkan rekor dengan mengalahkan performa manusia dalam mengklasifikasi objek pada gambar ImageNet. Dimana ResNet mendapat *error rate* 3.57%, lebih rendah dari *error rate* manusia yaitu 5.1% [21].

2.9. Transfer Learning

Dalam praktek, model CNN tidak pernah dilatih dari awal karena keterbatasan dataset yang mencukupi. Pada umumnya digunakan metode *transfer learning* yaitu menggunakan model yang sudah dilatih di dataset lain sebagai titik awal model dimana beban W sudah terlatih untuk memprediksi dataset sebelumnya. Terdapat 3 skenario penggunaan metode *transfer learning* [22]*:*

a. CNN sebagai *feature extractor:* Sebagai contoh akan digunakan model yang dilatih pada dataset ImageNet. Dataset ini berisi 1.2 juta gambar dengan 1000 kategori. Layer output yang bertugas untuk mengkategorikan 1000 kelas dari dataset ImageNet akan dihilangkan dan

diganti dengan layer output dengan jumlah sama dengan kelas dari dataset baru seperti yang terlihat pada **Gambar 2.31.** Pelatihan akan dilakukan hanya pada *layer* terakhir**.** Kemudian klasifikasi linear akan dijalankan untuk mendapatkan prediksi dari dataset baru, pada skenario ini *layer* sebelum *layer output* berpean sebagai *feature extractor* seperti yang terlihat di **Gambar 2.32** bagian kiri.

b. Fine-tuning the CNN: Pada metode ini *backpropagation* akan dilakukan untuk memperbarui beban W pada *layer* lain sebelum *layer* output. Semua *layer* bisa dilatih ulang atau hanya beberapa *layer* teratas saja yang dilatih ulang seperti pada **Gambar 2.32** bagian kanan. Pertimbangan mana saja *layer* yang akan dilatih ulang harus mempertimbangkan bahwa *layer* yang lebih rendah bisa saja digunakan untuk mempelajari fitur-fitur generik seperti *edge detection* dan *blob detector.*

c. Pretrained model: Karena pelatihan model dengan dataset besar seperti ImageNet membutuhkan waktu yang cukup lama (2-3 minggu dengan GPU), maka orang tidak melatih modelnya sendiri. Sudah ada model yang terlatih tersedia untuk digunakan orang lain yang menggunakan metode *transfer learning.*

Perlu beberapa pertimbangan dalam menggunakan metode ini. Jumlah dataset baru dan kemiripan dataset baru dengan dataset yang model gunakan sebelumnya perlu menjadi pertimbangan. Sebagai panduan, ada 4 skenario pertimbangan dalam menggunakan metode ini:

a. Dataset baru berukuran kecil dan mirip dengan dataset lama: Data tidak akan cukup untuk melakukan *fine-tuning.* Karena dataset sebelumnya mirip, maka akan diasumsikan bahwa *layer* memiliki fitur level tinggi

yang relevan dengan dataset baru. Maka akan digunakan klasifikasi linear pada model ini.

b. Dataset baru berukuran besar dan mirip dengan dataset lama: Karen dataset cukup banyak, maka bisa digunakan *fine-tuning* untuk model ini.
c. Dataset baru berukuran kecil dan berbeda dengan dataset lama: Karena data tidak mencukupi, klasifikasi linear akan digunakan. Namun karena tidak ada kemiripan antar dataset, klasifikasi linear akan diletakan di *layer* yang lebih rendah dikarenakan *layer* yang lebih tinggi memiliki fitur level tinggi yang dilatih secara spesifik untuk dataset lama.
d. Dataset baru berukuran besar dan berbeda dengan dataset lama: Dengan dataset yang cukup besar, model bisa dilatih dari nol. Namun dalam praktik, menerapkan inisialisasi beban W dari model yang sudah terlatih masih berguna dan bisa dilakukan.

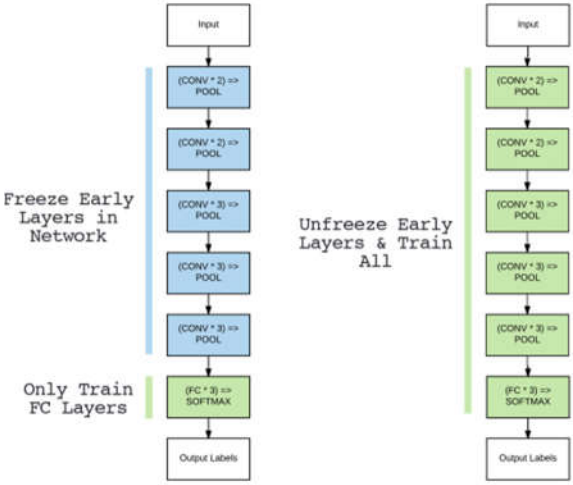
2.10. Object Detection

Tujuan dari klasifikasi gambar adalah memprediksi label untuk objek pada suatu gambar. Namun komputer tidak mengetahui bagian mana dari gambar yang merepresentasikan objek yang diprediksi. Tugas untuk mendeteksi posisi objek pada gambar disebut dengan *localization.* Tugas ini tidak hanya memiliki output berupa kelas, namun juga koordinat dari objek yang dideteksi. Koordinat ini memiliki bentuk x,y,h,w. Variabel x & y adalah koordinat posisi objek dan h & w adalah tinggi dan lebar objek. Maka untuk membedakan 2 jenis objek, model *classification* + *localization* memiliki output minimal 6 noda. 2 untuk kelas yang diprediksi dan 4 untuk koordinatnya.

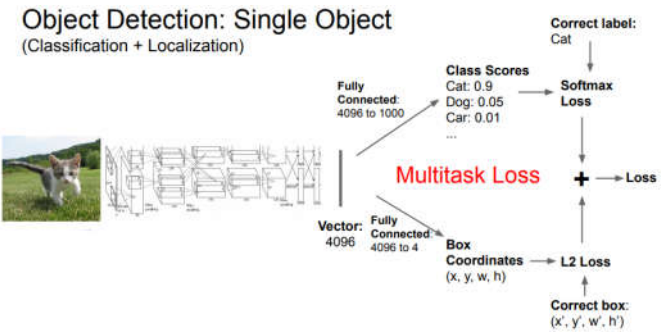
Gambar 2. 31. Metode *transfer learning* **[23]**. Kiri: model original. Tengah: Model setelah *layer* klasifikasi dihilangkan. Kanan: model diberi *layer* klasifikasi baru sesuai dataset baru

Jenis *loss* juga berbeda, untuk prediksi koordinat menggunakan *L2 loss.* Kombinasi ini menyebabkan *multitask loss* dimana model harus mengoptimasi parameter berdasarkan gabungan dari kedua *loss.* Tahap *classification + localization* terlihat seperti pada **Gambar 2.33.**

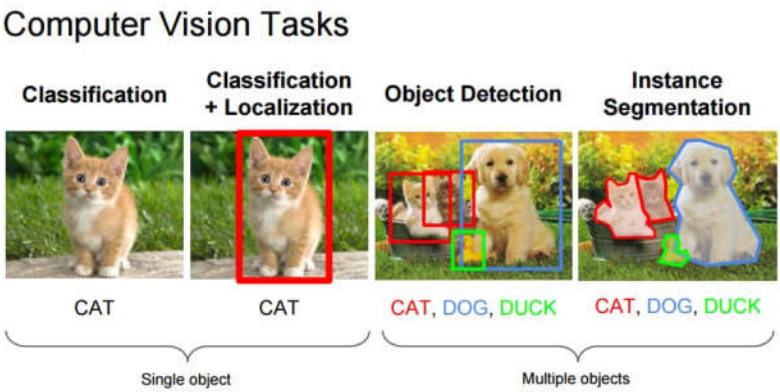
Localization tidak mampu mendeteksi objek lebih dari satu pada gambar yang sama. Untuk mendeteksi multi-objek, digunakan metode *object detection. Object detection* adalah salah satu tugas pada bidang *computer vision,* seperti terlihat pada **Gambar 2.34.** Metode ini tidak bisa mengikuti alur metode *localization* dimana output koordinat sudah ditentukan sejak awal. Pada metode *localization* sudah diketahui lebih dulu bahwa hanya ada 1 objek dalam gambar. Permasalahan muncul saat menentukan jumlah noda output seperti yang digambarkan pada **Gambar 2.35.**



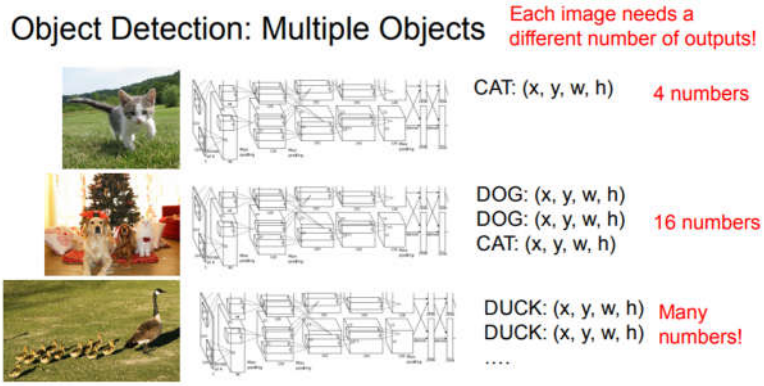
Gambar 2. 32. Kiri: *Layer* sebelumnya digunakan sebagai *feature extractor* dan tidak dilatih. Kanan: *fine-tuning* dilakukan pada seluruh *layer* **[23]**



Gambar 2. 33. Metode *classification+localization* **[24]**



Gambar 2. 34. Berbagai macam tugas penglihatan komputer **[24]**



Gambar 2. 35. Permasalahan *localization* untuk gambar multi objek **[24]**

Maka metode *object detection* memiliki pendekatan berbeda dengan *localization.* Metode ini memiliki pendekatan *region proposal* dimana sebelum deteksi objek dimulai,

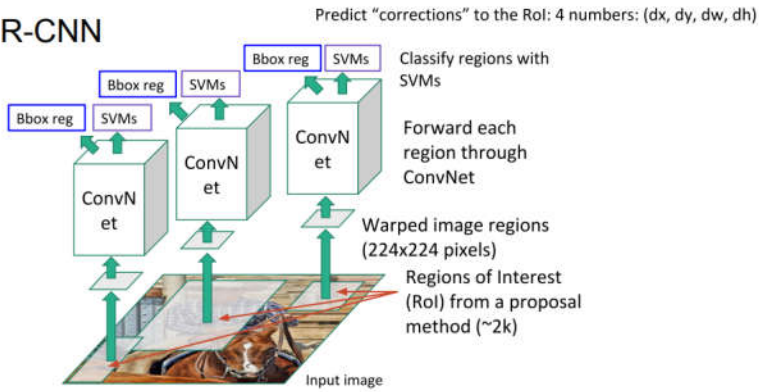
proposal daerah yang memungkinan keberadaan objek akan diberikan. Pendekatan proposal daerah ini juga disebut dengan *Region Convolutional Neural Network* atau R-CNN.

2.11. R-CNN

Metode R-CNN [25] dibagi menjadi beberapa tahap untuk mendeteksi gambar dengan multi-objek:

a. Kurang lebih 2000 proposal daerah atau *Region of Interest (ROI)* akan dipilih dari gambar input dengan menggunakan algoritma *Selective Search.*
b. ROI tersebut akan diubah dimensinya sesuai arsitektur yang digunakan.
c. ROI akan diteruskan ke CNN dengan arsitektur yang sudah ditentukan.
d. CNN akan mengklasifikasi ROI yang diberikan dan memberi *bounding box* sebagai koreksi lokasi objek yang lebih akurat.

Tahap ini divisualisasikan pada **Gambar 2.36.**



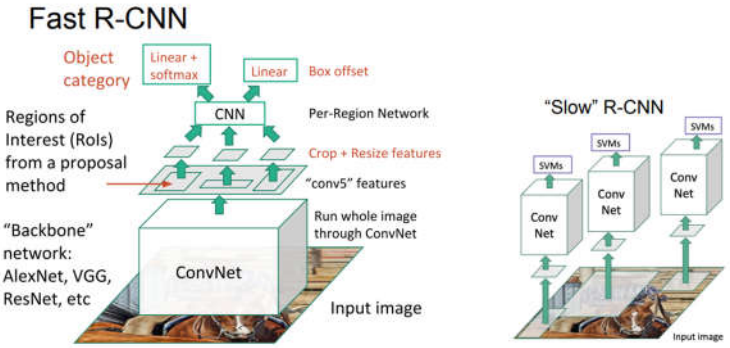
Gambar 2. 36. Tahap-tahap R-CNN **[24]**

Namun metode ini membutuhkan waktu yang lama. Masing-masing ROI yang berjumlah 2000 harus melewati model CNN yang berbeda sehingga memerlukan daya komputasi yang sangat tinggi. Hal ini menyebabkan waktu pelatihan yang lama yaitu untuk *training* 84 jam dan waktu *inference* 47 detik. Maka R-CNN dikembangkan menjadi *Fast R-CNN.*

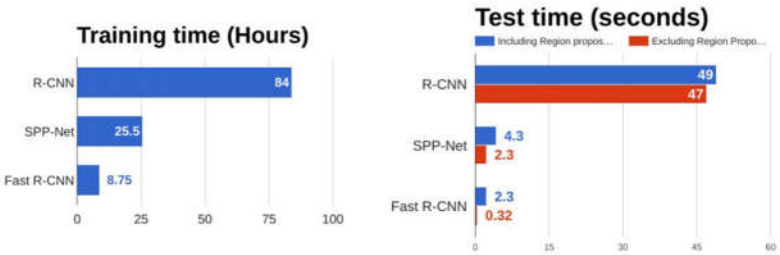
Fast R-CNN [26] melakukan deteksi objek dengan menggunakan tahap berikut:

a. Gambar input diproses oleh CNN dengan arsitektur pilihan.
b. Hasil CNN yang berbentuk *feature map* akan dijalankan dengan algoritma *selective search* untuk mendapatkan proposal *ROI.*
c. ROI akan dipotong dan diubah dimensinya sesuai input arsitektur selanjutnya.
d. ROI akan dimasukan ke CNN yang bertugas untuk mengklasifikasi objek dan memberi *bounding box* kepada objek.

Perbedaan R-CNN dan *fast* R-CNN bisa dilihat pada **Gambar 2.37.** Karena metode ini hanya menggunakan dua model CNN, maka metode ini jauh lebih cepat dari metode sebelumnya. ROI tidak diterapkan langsung kepada input gambar, melainkan pada *feature map* hasil model CNN. Metode ini terbukti lebih cepat pada fase *training* dan *inference (test)* dibandingkan metode sebelumnya seperti yang terlihat pada **Gambar 2.38.**



Gambar 2. 37. Perbandingan *Fast R-CNN* dengan *'slow' R-CNN* **[24]**



Gambar 2. 38. Waktu *training* dan *test* berbagai metode *region proposal* **[25] [26]**

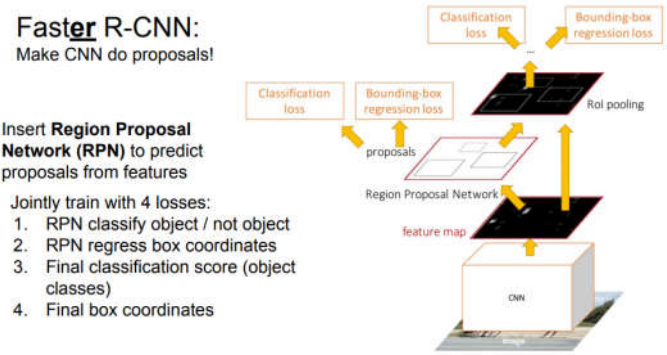
Terlihat pada G**ambar 2.38.** sebelah kanan, bar biru menunjukan waktu termasuk *region proposal* dan bar merah menunjukan waktu tanpa *region proposal.* Terlihat pata metode *fast R-CNN* banyak waktu terbuang pada tahap *region proposal.* Masalah ini akan diatasi dengan metode berikutnya yaitu *faster R-CNN.*

Pada metode *faster R-CNN* [27], tahap *region proposal* yang ditujukan untuk mencari ROI akan digantikan dengan

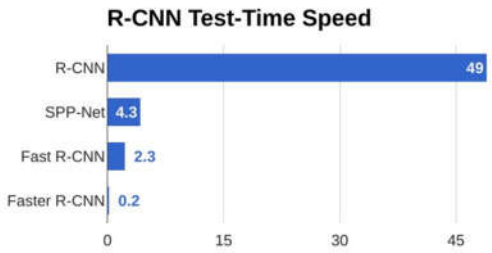
model *neural network.* Perubahan ini akan mempercepat proses deteksi objek dan menambahkan parameter yang bisa dipelajari oleh model secara keseluruhan. Metode ini memiliki tahap-tahap berikut:

a. Gambar input akan diproses CNN untuk mendapatkan *feature map*
b. ROI akan didapatkan dari *feature map* menggunakan *Region Proposal Network (RPN).* RPN adalah model *neural network* yang memprediksi kemungkinan keberadaan objek pada daerah tertentu. Model ini memiliki *loss* tersendiri yaitu *loss* klasifikasi keberadaan objek dan *loss* koordinat ROI.
c. ROI yang didapatkan dari RPN akan digabungkan bersama dengan *feature map* untuk dilakukan klasifikasi dan prediksi objek dalam ROI. Di model terakhir, terdapat 2 loss yang didapatkan. *Loss* klasifikasi objek dan *loss* koordinat objek.

Tahap *faster R-CNN* terlihat pada **Gambar 2.39.** Pada akhirnya metode ini adalah metode tercepat diantara metode sebelumnya, seperti yang terlihat pada **Gambar 2.40.**



Gambar 2. 39. Tahap-tahap *faster R-CNN* **[24]**

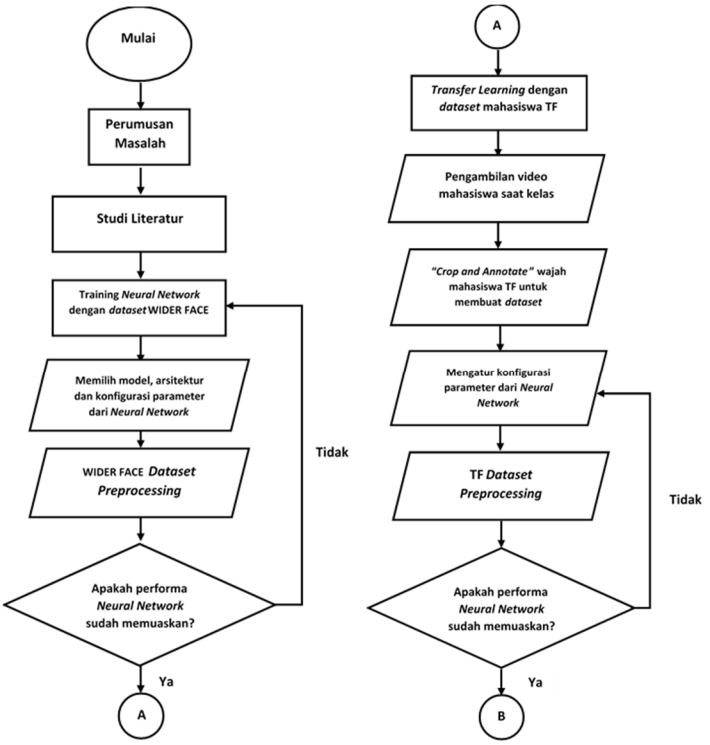


Gambar 2. 40, Perbandingan waktu *test* berbagai macam metode *region propsal* **[24]**

Halaman ini sengaja dikosongkan

BAB 3
METODOLOGI PENELITIAN

Flowchart metodologi penelitian pada tugas akhir ini bisa dilihat pada **Gambar 3.1.** dan **Gambar 3.2.**



Gambar 3. 1. *Flowchart* Metodologi Penelitian

3.1. Perumusan Masalah

Menyusun masalah dari penelitian ini yang kemudian akan dicari solusinya lewat studi literatur. Perumusan masalah membantu mengidentifikasi tujuan dari penelitian ini.

Gambar 3. 2. Lanjutan flowchart metodologi penelitian

3.2. Studi Literatur

Mencari penelitian sebelumnya serta paper dan jurnal yang mendukung penelitian ini. Penelitian yang dipelajari meliputi penelitian mengenai face recognition menggunakan deep neural network secara umum dan secara khusus yang digunakan untuk memantau mahasiswa dalam kelas. Studi mengenai teknologi face recognition yang sudah dikembangkan perusahaan IT juga dipelajari.

3.3. *Training* dengan dataset WIDER FACE

Training adalah proses pelatihan suatu *neural network* untuk mempelajari sebuah dataset. Untuk *face recognition,* dataset yang akan dipelajari adalah gambar wajah. WIDER FACE adalah dataset *benchmark* untuk *face detection.* Dataset ini berisi 32.203 gambar dengan 393.703 wajah yang telah diberi label. Contoh gambar dalam dataset ini terlihat pada **Gambar 3.3.** Dataset ini juga digunakan pada paper *Automatic Attendance Face Recognition for Real Classroom Environments* [28]

Sebelum *training* dimulai, perlu ditentukan model, arsitektur, dan parameter dari *neural network* yang akan digunakan:

a. **Model** adalah algoritma yang digunakan untuk melakukan *face recognition.* Setiap model memiliki metode yang berbeda dalam mengekstrak informasi dari pixel dalam gambar untuk mengenali objek di dalamnya. Pemilihan model sangatlah penting karena model mempengaruhi akurasi dan beban komputasi dari *neural network* yang akan dibuat. Setiap model memiliki kelemahan dan kelebihan, bergantung pada kompleksitas model tersebut. Beberapa model yang ada seperti: *Convolutional Neural Network(CNN), Recursive Convolutional Neural Network(RCNN), You Only Look Once (YOLO), DeepFace,* dan *FaceNet.*
b. **Arsitektur** adalah fondasi dari *neural network.* Arsitektur menentukan susunan neuron pada layer, pola koneksi antar layer, fungsi aktivasi dan metode pembelajaran [29]. Arsitektur dari *neural network* menentukan seberapa banyak informasi yang bisa di input dan menentukan seberapa banyak hal yang bisa dipelajari. Misalkan kita ingin mengolah gambar dengan resolusi 32x32 pixel untuk membedakan 2 wajah orang yang berbeda. Maka jumlah noda dalam layer input adalah 1024 sesuai total pixel. Dan jumlah noda dalam layer output adalah 2 sesuai jumlah wajah yang dipelajari.
c. **Hyperparameter** adalah variabel dalam model suatu *neural network* yang mempengaruhi performa. Parameter ini harus di *tuning* sedimikian rupa agar mendapatkan hasil terbaik. Setiap model memiliki parameter tersendiri, namun ada parameter yang digunakan di semua model seperti *learning rate, regularization, decay, momentum* dan optimisasi. Pemilihan parameter sangatlah penting untuk memastikan *neural network* bisa mempelajari dataset secara optimal. Beberapa contoh dari parameter

adalah *learning rate* yang menentukan seberapa cepat *neural network* belajar, optimisasi yang menentukan metode pembelajaran suatu *neural network,* dan *regularization* yang membatasi informasi yang dipelajari oleh *neural network.*

Dataset juga harus melewati tahap *preprocessing* dahulu sebelum masuk tahap *training.* Dataset perlu perlakuan khusus seperti perubahan dimensi gambar sebelum siap digunakan untuk *training neural network.*

Gambar 3. 3. Kumpulan gambar dalam dataset WIDER FACE **[5]**

3.4. *Transfer learning* dengan dataset Mahasiswa TF

Karena *neural network* dilatih dengan dataset WIDER FACE maka *neural network* sudah bisa mengenali wajah-wajah orang dalam dataset tersebut. Namun penelitian ini ditujukan untuk mengenali wajah mahasiswa TF. Maka kemudian *neural network* harus dilatih dengan menggunakan dataset berisi gambar wajah mahasiswa TF.

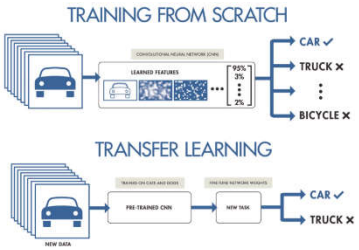
Kenapa *neural network* tidak dilatih langsung dengan dataset mahasiswa TF? Karena untuk melatih *neural network* dari nol dibutuhkan ribuan gambar sebagai dataset. Sangat tidak praktis untuk mengumpulkan ribuan gambar wajah

mahasiswa TF dalam jangka waktu penelitan ini. Maka digunakanlah dataset WIDER FACE yang sudah siap dipakai untuk *training* dengan dataset yang besar sejumlah 32.203 gambar. Tujuan dari *training* dengan dataset WIDER FACE adalah untuk melatih *neural network* dalam mengenali fitur wajah secara umum. Dataset WIDER FACE juga digunakan sebagai *benchmark* dari akurasi prediksi *neural network* tersebut. Dataset WIDER FACE sudah sering digunakan sebagai *benchmark* untuk melihat akurasi dari model *face recognition* yang sedang dikembangkan.

Transfer learning adalah proses pelatihan suatu *neural network* dengan menggunakan *neural network* lain yang sudah dilatih. *Neural network* yang telah dilatih dengan dataset WIDER FACE akan digunakan untuk melatih *neural network* dengan dataset mahasiswa TF. Dengan *transfer learning,* dataset wajah mahasiswa TF tidak perlu sebanyak dataset WIDER FACE untuk mencapai akurasi tinggi, karena *neural network* sudah dilatih untuk mengenali bentuk wajah secara umum. Hal ini memperingan proses pengumpulan wajah mahasiswa TF. Tentunya dataset mahasiswa TF harus dibuat terlebih dahulu, tidak seperti dataset WIDER FACE yang sudah siap digunakan dan bisa diunduh. Perbandingan *transfer learning* dengan *training* dari awal terlihat pada **Gambar 3.4.**

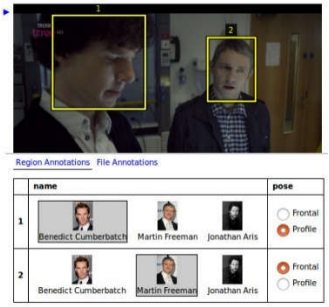
3.5. Membuat dataset mahasiswa TF

Untuk membuat dataset, pertama akan dilakukan simulasi pembelajaran dalam kelas. Video akan diambil dari depan kelas yang memperlihatkan sejumlah mahasiswa (6 orang) dalam kelas. Video akan diambil dengan kamera belakang *Samsung Galaxy S7* yang dipasang dengan tripod. Video akan diambil dengan variasi resolusi 720p dan 1080p dengan durasi 20 detik setiap pengambilan dan dengan kecepatan ambil video 24 fps.



Gambar 3. 4. Perbedaan *training* dari 'nol' dan *transfer learning* **[30]**

Setelah video diambil, wajah mahasiswa dalam video akan dipotong *(crop)* dan ditandai (*annotate*). Proses ini dilakukan di setiap *frame* dari video. Maka video dengan durasi 20 detik dan 24 fps, dihasilkan sebanyak 480 frame. Dengan wajah mahasiswa yang sudah dipotong dan ditandai, dataset mahasiswa TF siap digunakan untuk *training.* Contoh prosess *annotate* terlihat pada **Gambar 3.5.**



Gambar 3. 5. Contoh proses *annotate* untuk membuat dataset wajah **[15]**

3.6. Sistem pemantauan mahasiswa dalam kelas

Setelah *neural network* melalui proses *transfer learning, neural network* ini mampu mengenali wajah mahasiswa dari

dataset mahasiswa TF. Berikutnya akan dibuat sistem untuk memantau mahasiswa dalam kelas. Fungsi pemantauan ini meliputi mengetahui nama mahasiswa dalam kelas dan menghitung jumlah mahasiswa dalam kelas. Sistem ini mengatur kebutuhan pendukung agar model bisa diimplementasikan dalam kelas. Hal ini meliputi pemilihan jenis kamera, GPU, dan penempatan posisi kamera.

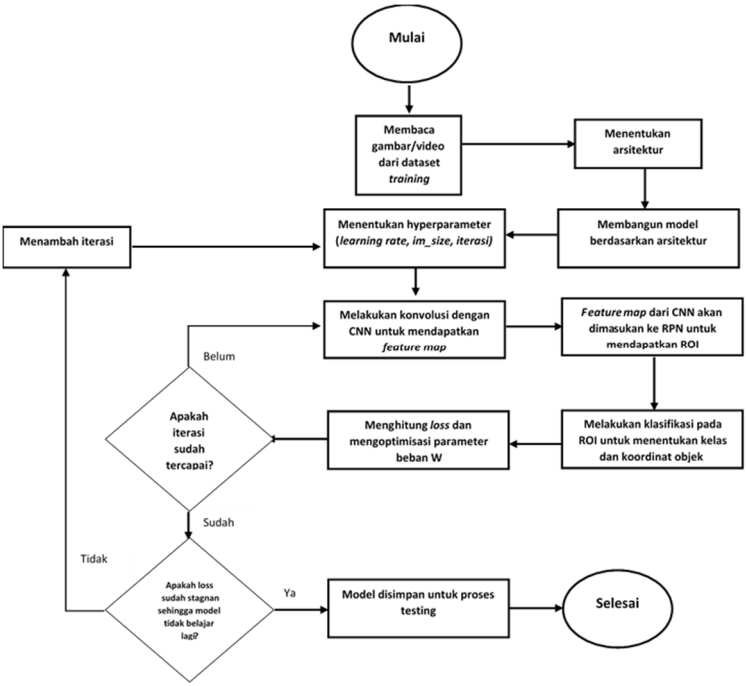
Halaman ini sengaja dikosongkan

BAB 4
HASIL DAN PEMBAHASAN

Untuk membuat program yang bisa mendeteksi wajah diperlukan beberapa variabel yang harus ditentukan sebelum memulai pelatihan. Model deteksi objek yang akan digunakan adalah *Region Convolution Neural Network* atau R-CNN. Kemudian dibutuhkan dataset yang berkaitan dengan objek yang akan dideteksi. Akan digunakan dua jenis dataset: dataset sekunder yaitu WIDER FACE, dan dataset primer yaitu dataset berisi 6 wajah mahasiswa teknik fisika ITS. Setelah dataset, arsitektur dari model *deep neural network* harus ditentukan. Akan diuji 2 jenis arsitektur yaitu VGG-16 dan Resnet-50. Pelatihan model akan dibagi 2 tahap. Tahap pertama model yang bisa mendeteksi wajah secara umum akan dilatih dengan menggunakan dataset WIDER FACE. Dan pada tahap kedua akan dilakukan *transfer learning* dengan menggunakan dataset mahasiswa TF. Model ini akan dilatih untuk bisa mendeteksi wajah 6 mahasiswa teknik fisika beserta nama masing-masing.

4.1. Diagram Alir Program

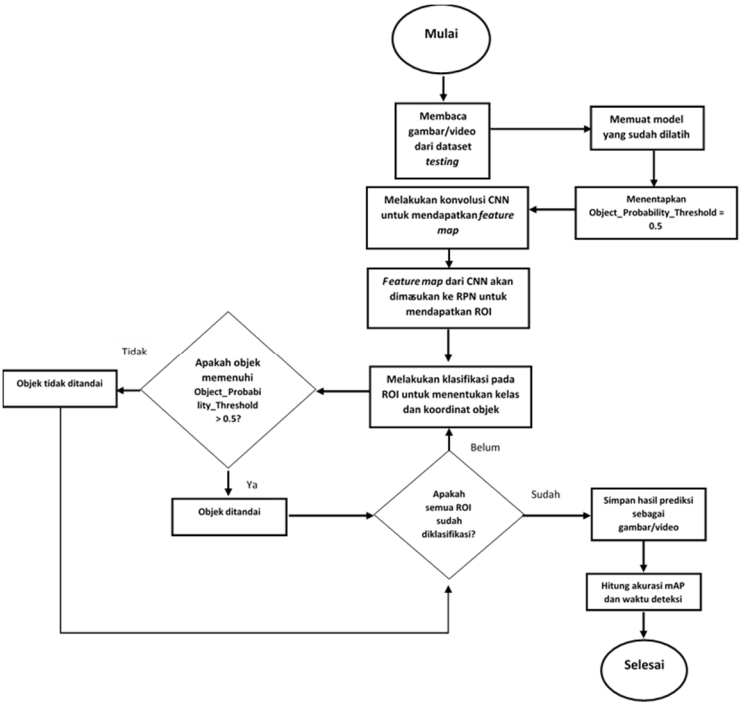
Akan dibuat 2 program untuk sistem pemantauan mahasiswa. Program pertama digunakan untuk melatih model dengan input dataset *training.* Program kedua digunakan untuk menguji model yang sudah dilatih di program sebelumnya dengan menggunakan dataset *test*. Hasil tes berupa gambar dengan prediksi wajah serta metrik pengukuran akurasi mAP *(mean average precision)* dan waktu deteksi. Diagram alir kedua program terlihat pada **Gambar 4. 1** dan **Gambar 4. 2**.



Gambar 4. 1. Diagram alir program *training*

Pada program *training* model akan dilatih menggunakan dataset *training.* Arsitektur yang akan digunakan model akan ditentukan antara arsitektur VGG-16 dan ResNet-50. Kemudian *hyperparameter* berupa *learning rate, im_size,* dan iterasi akan ditentukan sebelum *training* dimulai. *Training* dimulai dengan konvolusi yang akan dilakukan dengan CNN *(convolutional neural network)* untuk mendapatkan *feature map* yang berisi informasi dan ciri-ciri objek pada gambar. Kemudian *feature map* akan dimasukan ke RPN *(region proposal network)* untuk mendapatkan ROI *(region of interest)*, yaitu daerah yang memiliki potensi objek. ROI

tersebut akan diklasifikasi untuk mendapatkan kelas objek dan koordinatnya. Setelah klasifikasi, *loss* akan dihitung berdasarkan prediksi kelas dan koordinat objek. Dan pada akhirnya parameter beban W akan dipotimisasi untuk menurunkan *loss* di iterasi berikutnya. Proses *training* ini akan diulang sesuai banyak iterasi yang sudah ditentukan. Jika *loss* sudah stagnan dan model tidak bisa belajar lagi dari dataset, model akan disimpan pada iterasi dengan *loss* terendah.



Gambar 4. 2. Diagram alir program *testing*

Pada program *testing* model yang sudah dilatih di program sebelumnya akan dimuat dan digunakan untuk menguji dataset *testing.* Sebelum dilakukan *testing,* variabel *Object_Probability_Threshold* akan ditentukan sebesar 0.5. Proses *testing* akan dilakukan menyerupai proses *training.* Dataset akan dikonvolusi dengan CNN untuk mendapatkan *feature map,* RPN akan menghasilkan ROI, dan ROI akan diklasifikasi untuk mendapatkan kelas dan koordinat objek. Tahap *testing* tidak memiliki iterasi seperti tahap *training*. Jika objek memiliki *Object_probability* dibawah 0.5, maka objek tidak akan ditandai. Jika probabilitas objek diatas *threshold* 0.5*,* maka objek akan ditandai dengan gambar kotak pada objek beserta teks dengan nama kelas objek tersebut. Hasil prediksi kemudian akan disimpan dalam bentuk gambar atau video. Dan akhirnya akurasi mAP dan waktu deteksi akan dihitung berdasarkan prediksi objek yang dilakukan oleh model.

4.2. Model R-CNN

R-CNN adalah salah satu jenis model deteksi objek yang tidak melihat gambar secara utuh. Proses deteksi dilakukan pada beberapa daerah yang memiliki potensi tinggi mengandung objek. Selain R-CNN terdapat model deteksi objek lain yaitu *Single Shot Detection* atau SSD. Tidak seperti R-CNN yang melakukan beberapa kali deteksi pada daerah tertentu, SSD hanya melakukan satu kali deteksi dengan melihat gambar secara utuh.

Untuk program deteksi wajah ini, R-CNN adalah model yang akan digunakan. Dengan mempertimbangkan mAP dan waktu deteksi, model R-CNN lebih baik dari model SSD dalam mendeteksi objek yang kecil [31]. Dan dalam program ini, wajah yang dideteksi berukuran kecil relatif terhadap ukuran gambar secara keseluruhan.

4.3. Dataset WIDER FACE

Dataset ini menjadi basis pembelajaran program dalam mengenali wajah secara umum. Dataset ini terbagi menjadi 2 yaitu dataset untuk *training* dan dataset untuk *test.* Dataset training adalah data yang akan dipelajari langsung oleh program. Sedangkan dataset *test* akan digunakan untuk menilai performa model.

Dataset *training* terdiri dari 12.880 gambar yang memiliki 79.111 wajah. Sedangkan dataset *test* terdiri dari 3.226 gambar yang memiliki 19.762 wajah. Data ini memiliki berbagai macam variasi wajah seperti ekspresi, pencahayaan, dan lain lain untuk menambah kekayaan data yang dipelajari model. Variasi data ini terlihat pada **Gambar 4. 3**.

Gambar 4. 3. Kumpulan gambar dalam dataset WIDER FACE **[5]**

4.4. Dataset Mahasiswa TF

Dataset ini menjadi basis pembelajaran program dalam mengenali 6 wajah mahasiswa teknik fisika beserta nama masing-masing mahasiswa. Dataset ini berisi gambar yang diambil dari video dalam kelas kampus dengan durasi 20 detik dalam 24 fps. Video diambil dengan menggunakan kamera belakang telefon pintar *Samsung Galaxy S7* dari posisi di depan

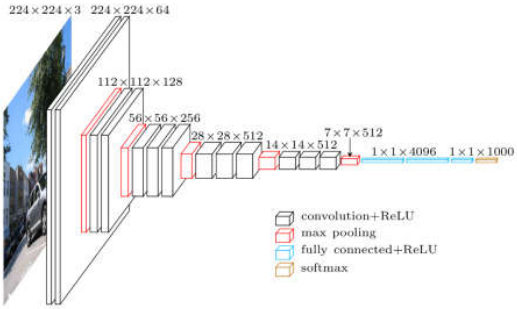
papan tulis menghadap ke arah mahasiswa. Contoh gambar dalam dataset dapat dilihat pada **Gambar 4. 4.**

Gambar 4. 4. Salah satu gambar dalam dataset mahasiswa TF

Seperti dataset sebelumnya, dataset ini dibagi menjadi dataset *training* dan dataset *test.* Namun dataset *test* terbagi menjadi 2 berdasarkan format resolusi saat pengambilan video yaitu 720p dan 1080p. Dataset *training* terdiri dari 908 gambar yang memiliki 5.148 wajah. Sedangkan dataset *test* memiliki 375 gambar yang memiliki 2.250 gambar. Pada saat pengambilan video mahasiswa melakukan beberapa pose seperti menoleh ke samping, ke atas dan ke bawah. Hal ini dilakukan untuk menambah variasi data. Ekspresi wajah tidak diambil dari wajah mahasiswa ini karena dari dataset sekunder WIDER FACE sudah terdapat variasi ekspresi yang lebih kaya.

4.5. Arsitektur VGG-16

Arsitektur VGG-16 adalah arsitektur *deep neural network* yang dibuat oleh Simonyan dan Zisserman dalam paper mereka dengan judul *Very Deep Convolutional Networks for Large Scale Image Recognition* [32] yang dipublikasikan pada tahun 2014. Susunan arsitektur ini bisa dilihat pada **Gambar 4. 5.**



Gambar 4. 5. Arsitektur VGG-16 **[33]**

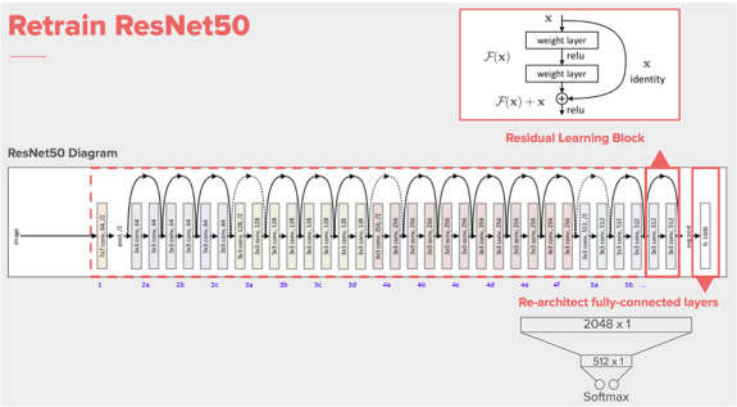
Ciri khas VGG-16 adalah simplisitas susunan *layer* dalam arsitektur ini. VGG terdiri dari 3 layer utama, *convolution layer, pooling layer,* dan *fully-connected layer.* Sesuai namanya, VGG-16 memiliki total 16 layer. VGG adalah salah satu pemenang dalam kompetisi *ImageNet Large Scale Visual Recognition Challange* pada tahun 2016 dengan *error rate* sebesar 7.3%. Namun VGG memiliki kelemahan yaitu membutuhkan waktu *training* yang lama dan model yang dihasilkan memiliki ukuran relatif besar yaitu 512 MB.

4.6. Arsitektur ResNet-50

ResNet adalah salah satu arsitektur yang berhasil mengalahkan performa manusia dalam mengklasifikasikan gambar pada kompetisi *ImageNet Large Scale Visual Recognition Challange.* Dimana ResNet mendapatkan *error rate* sebesar 3.57%, lebih rendah dibandingkan *error rate* manusia yaitu 5.1%. ResNet juga memiliki layer yang jauh lebih dalam dibandingkan dengan arsitektur pemenang kompetisi *ImageNet* pada tahun sebelumnya yaitu 152 layer dibandingkan dengan GoogleNet, pemenang tahun sebelumnya

dengan 22 layer. Namun pada penelitian ini akan digunakan variasinya yaitu ResNet-50 dengan total 50 layer.

Resnet dibuat oleh Kaiming He, Xiangyu Zhang, Shaoqing Ren dan Jian Sun dalam paper mereka dengan judul *Deep Residual Learning for Image Recognition* [34]. Dijelaskan dalam paper ini bahwa masalah terbesar dari arsitektur dengan *layer* yang sangat dalam adalah fenomena *vanishing gradient.* Dimana seiring gradient mengalami *backprogation,* ukuran gradient menjadi semakin mengecil karena proses perkalian berkali-kali. Sampai pada titik gradient sangat kecil sehingga performa arsitekur menjadi stagnan. Solusi *vanishing gradient* pada paper disini adalah dengan menggunakan *residual block* seperti yang terlihat pada **Gambar 4. 6.** *Residual block* berfungsi sebagai jalan pintas gradient saat *backpropagation* sehingga efek *vanishing gradient* bisa dikurangi.



Gambar 4. 6. Arsitektur ResNet50 beserta diagram *residual block* **[35]**

4.7. Metrik Pengukuran yang Digunakan

Dalam penelitian ini akan digunakan beberapa macam metrik pengukuran untuk mengukur performa model dalam mendeteksi wajah. Metrik ini adalah mAP(*mean average precision*), *Loss,* dan waktu deteksi.

4.7.1. mAP

mAP atau *mean average precision* adalah bentuk pengukuran akurasi yang populer digunakan dalam bidang *object detection.* mAP dihitung dari rata-rata *average precision* dari seluruh kelas. Untuk menghitung *average precision,* dibutuhkan pemahaman mengenai *confusion matrix*, *precision*, *recall* dan IoU.

Confusion matrix menilai performa model dengan membaginya menjadi 4 kategori:

a. *True Negative* adalah saat model memprediksi negatif dan data sesungguhnya adalah negatif
b. *True Positive* adalah saat model memprediksi positif dan data sesungguhnya adalah positif
c. *False Negative* adalah saat model memprediksi negatif dan data sesungguhnya adalah positif.
d. *False Postive* adalah saat model memprediksi positif dan data sesungguhnya adalah negatif.

Bagan *confusion matrix* terlihat pada **Gambar 4. 7.** *Precision* menghitung seberapa akurat prediksi model. Jika prediksi model salah mengkategorikan kelas, maka nilai *precision* akan berkurang. *Recall* menghitung seberapa baik model dalam mendeteksi objek positif, maka nilai *recall* akan berkurang jika model gagal mendeteksi objek positif. Rumus *precision* dan *recall* terlihat pada **Gambar 4. 8.**

		Predicted	
		Negative	**Positive**
Actual	**Negative**	True Negative	False Positive
	Positive	False Negative	True Positive

Gambar 4. 7. *Confusion Matrix*

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

TP = True positive

TN = True negative

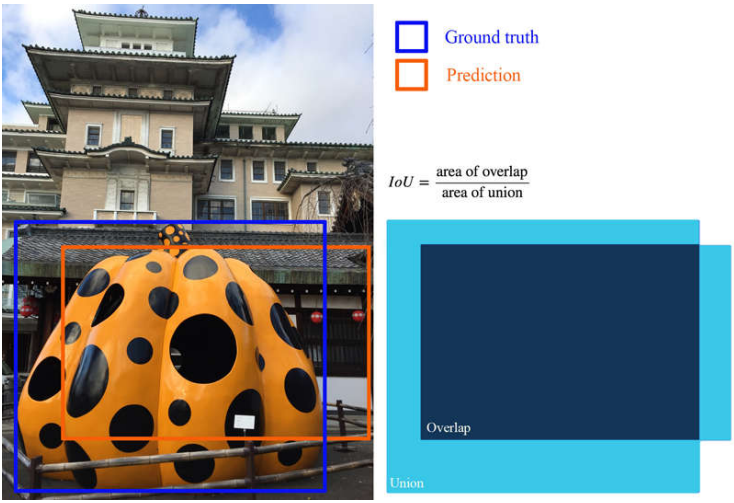
FP = False positive

FN = False negative

Gambar 4. 8. Rumus *precision* dan *recall* **[36]**

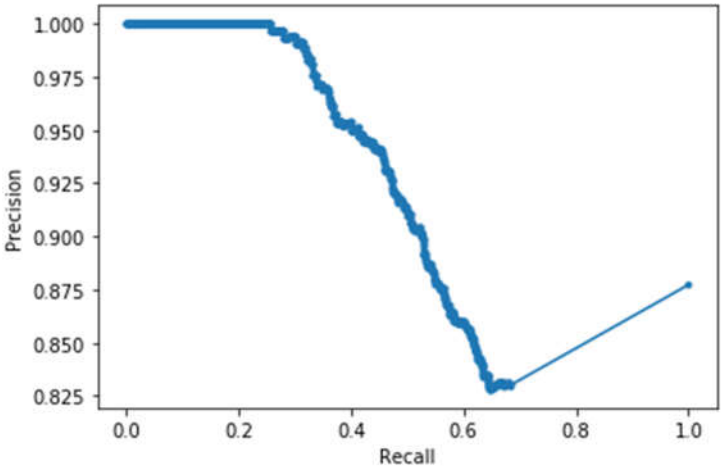
Pada penelitian ini *true positive* adalah ketika model mendeteksi wajah dan memprediksi nama mahasiswa dengan benar. *False positive* adalah ketika model salah memprediksi nama mahasiswa. Dan *false negative* adalah ketika model gagal mendeteksi wajah. Prediksi dinyatakan *true positive* jika prediksi wajah berhasil mengklasifikasikan wajah dengan nama mahasiswa yang benar dan prediksi memiliki IoU diatas 0,5.

IoU atau *intersection over union* adalah luas area peropotongan kotak *ground truth* dengan kotak *prediction* dibandingkan dengan total luas kedua kotak jika digabungkan. Penggambaran IoU terlihat pada **Gambar 4. 9**.



Gambar 4. 9. *Intersection over Union* **[36]**

Pada *object detection,* positif (objek) akan selalu lebih sedikit daripada negatif (selain objek/latar belakang). Terdapat ketidakseimbangan kelas saat model mencoba untuk mendeteksi objek. Akan terjadi bias terhadap negatif dan akurasi akan berubah drastis jika model gagal mendeteksi objek. Maka untuk *object detection,* metrik pengukuran yang digunakan adalah *average precision* atau AP. AP diambil dari *precision recall curve.* Kurva ini menunjukan keseimbangan antara sensitivitas deteksi (*recall*) dan akurasi model dalam memprediksi objek dengan kelas yang benar (*precision*). Contoh kurva ini terlihat pada **Gambar 4. 10**. AP adalah luas dibawah kurva tersebut. Rumus AP terlihat pada **Gambar 4. 11** , dimana p adalah *precision* dan r adalah *recall*.



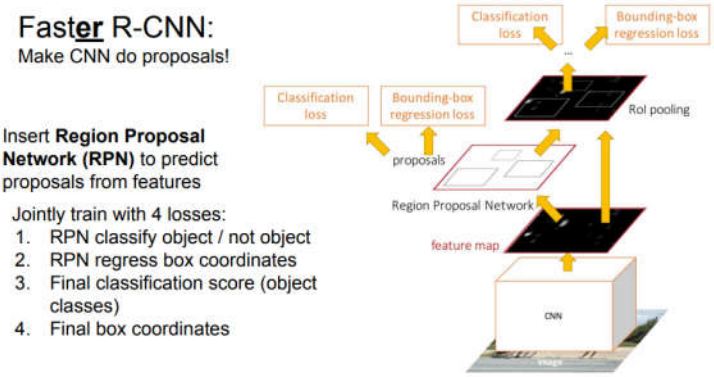
Gambar 4. 10. Contoh *Precision-Recall Curve* pada test ResNet-50 1080p.

$$\mathrm{AP} = \int_0^1 p(r)dr$$

Gambar 4. 11. Rumus *average precision*

4.7.2. Loss

Loss dihitung dengan menjumlahkan 4 jenis *loss.*pada metode *faster R-CNN.* Berdasarkan *layer, loss* dibagi menjadi 2 pada *region proposal network layer* dan *final classification layer.* Setiap *layer* tersebut memiliki *classifier loss* dan *bounding box loss. Classifier loss* dihitung dengan menggunakan *softmax classifier* dan *bounding box loss* dihitung dengan menggunakan L1 *loss.* Pembagian 4 jenis loss pada metode *Faster RCNN* terlihat pada **Gambar 4. 12**. Rumus L1 *loss* terlihat pada **Gambar 4. 13**, dimana y_i adalah *ground truth* dan $f(x_i)$ adalah skor prediksi.



Gambar 4. 12. Jenis *loss* pada *faster R-CNN* **[24]**

$$S = \sum_{i=1}^{n} |y_i - f(x_i)|.$$

Gambar 4. 13. Rumus L1 *loss*

4.7.3. Detection Time

Detection time adalah waktu yang dibutuhkan model untuk memproses gambar, melakukan prediksi, hingga menampilkan hasilnya kembali dalam bentuk gambar. Waktu deteksi dihitung dalam sekon, yaitu waktu yang dibutuhkan program untuk memproses satu gambar. Banyak faktor yang mempengaruhi waktu deteksi, seperti:

a. Kecepatan *storage* membaca dan menulis data
b. Pilihan GPU dalam proses *inference* dan *rendering* gambar
c. *Latency* atau delay kecepatan internet (jika deteksi dilakukan lewat internet)

Dengan berbagai faktor tersebut, perhitungan waktu deteksi akan menggunakan asumsi dan kondisi berikut:

a. Waktu menulis dan membaca sesuai performa server milik Google Colab
b. GPU yang digunakan adalah Tesla K80 12GB DDR5
c. Tidak ada *latency* atau delay kecepatan internet

4.7.4. Error rate

Error rate mengukur seberapa banyak prediksi kelas yang salah dibandingkan dengan kelas sebenarnya. *Error rate* digunakan untuk memilih kedua arsitektur yang digunakan pada penelitiah ini (VGG-16, dan ResNet-50). *Error rate* kedua arsitektur ini diambil saat meprediksi dataset ImageNet pada kompetisi *ImageNet Large Scale Visual Recognition Challange* [20]. *Error rate* dihitung menggunakan rumus pada **Gambar 4. 14**

$$\mathrm{ERR} = \frac{\mathrm{FP} + \mathrm{FN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FN} + \mathrm{FP}}$$

Gambar 4. 14. Rumus perhitungan *error rate*

4.8. Optimisasi *Hyperparameter*

Sebelum pelatihan dimulai, nilai berbagai *hyperparameter* akan ditentukan. Proses optimisasi nilai *hyperparameter* akan dilakukan dengan metode manual, yaitu dengan memilih nilai *hyperparameter* berdasarkan intuisi, pengalaman, dan refrensi. Nilai *hyperparameter* yang akan dioptimisasi adalah *learning rate, im_size,* dan *loss.*

4.8.1. *Learning Rate*

Learning rate adalah *hyperparameter* yang menentukan seberapa cepat *optimizer* mengoptimisasi parameter beban W. Pada penelitian ini akan digunakan *Adam* sebagai metode optimisasi. Pemilihan *learning rate* sangatlah penting karena menentukan seberapa cepat model belajar dari dataset. Jika *learning rate* terlalu rendah, model akan kesulitan mencapai titik *loss* terendah. Jika *learning rate* terlalu tinggi, nilai *loss* model akan naik dan turun terlalu cepat sehingga mengakibatkan proses training *crash.* Untuk mengoptimasi *learning rate,* akan diambil nilai berdasarkan refrensi dari *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks* [27] dan *Faster R-CNN (object detection) implemented by Keras for custom data from Google's Open Images Dataset V4* [37].

Nilai *learning rate* ini akan dites dengan melatih model dengan menggunakan arsitektur VGG-16 dan dataset WIDER FACE selama 22 epoch. Hasil test terlihat pada **Tabel 1.** yang membandingkan setiap nilai *learning rate* dengan *loss* masing-masing nilai.

Data hasil tes pada **Tabel 1.** menunjukan *learning rate* terbaik dalam segi *loss* adalah 1.00E-05. Maka *learning rate* 1.00E-05 akan digunakan untuk melatih model. *Loss* selama 22 epoch dengan 3 variasi *learning rate* terlihat pada **Gambar 4. 15.**

Gambar 4. 15. *Loss* selama 22 epoch dengan 3 variasi *learning rate.* Semakin rendah semakin baik.

Tabel 1. Hasil tes berbagai nilai *learning rate* terhadap *loss* di epoch ke-22

Sumber	Learning Rate	Loss
[27]	1.00E-03	5.04
[27]	1.00E-04	5.392
[37]	1.00E-05	1.868

4.8.2. *Im_size*

Sebelum gambar diproses oleh model pada tahap pelatihan, resolusi gambar akan diubah agar ukuran semua gambar pada dataset training memiliki resolusi yang sama tanpa merubah aspect rasio. Ukuran gambar mempengaruhi banyaknya pixel yang diproses oleh model dan lamanya proses pelatihan. Semakin besar resolusi dan banyak pixel, lebih banyak informasi yang bisa dipelajari oleh model. Namun waktu pelatihan akan bertambah. Ukuran gambar setelah diubah diatur oleh *hyperparameter im_size*. Jika *im_size=300,*

maka sisi terpendek gambar akan menjadi 300 pixel dan sisi lain akan mengikuti sesuai agar tidak mengubah *aspect rasio* gambar.

Untuk mengoptimalkan *im_size* akan dites 2 variasi nilai yang diambil dari refrensi dari sumber yang sama dengan optimasi *learning rate.* Nilai *im_size* ini akan dites dengan melatih model dengan menggunakan arsitektur VGG-16 dan dataset WIDER FACE selama 40 epoch. Hasil test terlihat pada **Tabel 2.** yang membandingkan setiap nilai *im_size* dengan *loss,* dan waktu pelatihan masing-masing nilai. Grafik pada **Gambar 4. 16.** *loss* selama 40 epoch.

Gambar 4. 16. *Loss* selama 40 epoch dengan 2 variasi im_size

Tabel 2. Hasil tes berbagai nilai *im_size* terhadap *loss,* dan waktu pelatihan di epoch ke-40

Sumber	im_size	Loss	Waktu pelatihan per epoch (menit)
[37]	300	1.711	14
[27]	600	1.455	26

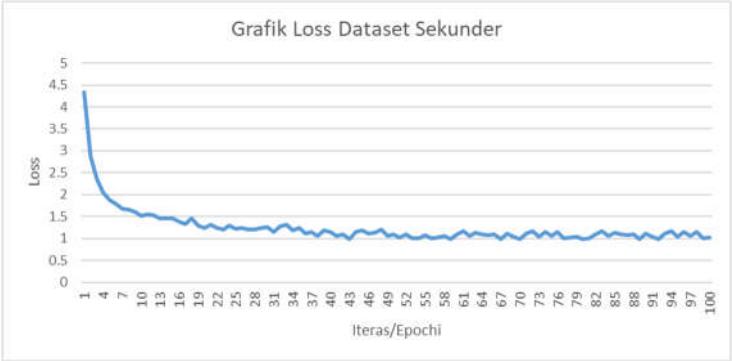
Data hasil tes pada **Tabel 2.** menunjukan *im_size* 600 memiliki *loss* terbaik. Namun waktu pelatihan *im_size* 600 lebih tinggi karena pixel yang harus diproses lebih banyak. Karena *hyperparameter im_size* hanya berlaku pada proses pelatihan, penambahan waktu tidak akan mempengaruhi waktu deteksi wajah oleh model. Maka *im_size* 600 akan digunakan dalam proses pelatihan model.

4.8.3. Iterasi

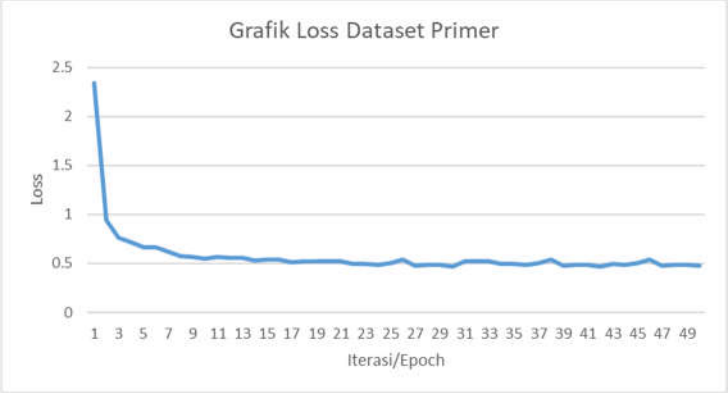
Iterasi atau *epoch* adalah banyaknya pengulangan pelatihan model. Setiap iterasi, model akan menghitung *loss* dan melakukan optimisasi beban W untuk menurunkan *loss* pada iterasi selanjutnya. Pemilihan banyak iterasi dilakukan bersamaan dengan *training* model dengan memperhatikan grafik *loss.* Berikut grafik *loss* dari *training* dataset sekunder dan dataset primer terlihat pada **Gambar 4. 17** dan **Gambar 4. 18**.

Pada grafik *loss* tersebut akan dilatih model dengan iterasi kelipatan 50. Setiap akhir iterasi kelipatan 50, akan dicek apakah terjadi stagnansi pada grafik *loss.* Jika terjadi stagnansi, berarti model tidak bisa belajar lagi dan iterasi tidak akan dilanjutkan. Akan dipilih jumlah iterasi dimana model mendapatkan *loss* terendah. Iterasi yang akan digunakan adalah

pada 80 *epoch* dengan *loss* 0.989 untuk dataset sekunder dan 30 *epoch* dengan *loss* 0.464 untuk dataset primer.



Gambar 4. 17. Grafik Loss Dataset Sekunder dengan iterasi 100



Gambar 4. 18. Grafik Loss Dataset Primer dengan iterasi 50

4.9. *Training* dengan dataset WIDER FACE

Semua pelatihan model akan dilakukan dengan menggunakan metode *faster R-CNN*. Pelatihan juga dilakukan di *cloud* dengan menggunakan salah satu layanan *cloud* dari

Google yaitu Google Colab. Pelatihan model akan dimulai dengan menggunakan dataset WIDER FACE. Model akan mempelajari 2 kelas yaitu wajah dan latar belakang. Dengan melatih dengan dataset ini, model akan mempelajari bentuk wajah dari berbagai sudut pandang. Akan digunakan aristektur VGG-16 dan ResNet50 untuk membandingkan arsitektur mana yang memiliki performa terbaik. Perbandingan antar aristektur terlihat pada **Tabel 3.**

Untuk membandingkan kedua aristektur, model dilatih dengan iterasi 80 epoch. Kemudian model dites dengan data validasi dari dataset WIDER FACE. mAP dan *detection time* adalah faktor yang akan menentukan performa model.

Tabel 3. Hasil tes dengan menggunakan data validasi WIDER FACE

Architecture	Dataset	Epoch	mAP (%)	Detection Time(sekon)
VGG-16	WIDER FACE	80	54	0.5
ResNet50	WIDER FACE	80	97	1

Data hasil tes pada **Tabel 3.** menunjukan arsitektur ResNet-50 unggul jauh dalam nilai mAP. Namun VGG-16 memiliki kenggulan sedikit lebih cepat dalam waktu deteksi. Dengan pertimbangan bahwa nilai mAP jauh lebih tinggi dan perbedaan waktu deteksi yang tidak terlalu besar, arsitektur ResNet-50 adalah arsitektur yang akan digunakan oleh model.

Hasil tes bisa dibandingkan dengan penelitian serupa mengenai *face recognition* menggunakan dataset yang sama. Dataset validasi WIDERFACE membagi 3 katagori tes berdasarkan kesulitan deteksi yaitu *easy, medium* dan *hard*. Semakin tinggi kesulitannya, semakin rendah nilai *recall/detection rate.* Rata-rata *recall* pada dataset ini adalah 92%, 76%, 34% untuk kesulitan *easy, medium* dan *hard*

menurut paper WIDER FACE [5]. Pembagian ini juga didasari ukuran wajah pada gambar. Data *easy* memiliki ukuran wajah diatas 300 pixel, *medium* diantara 50-300 pixel, dan *hard* diantara 10-50 pixel. Perbandingan metode yang ditawarkan dengan paper lain terlihat pada **Tabel 4.** Dengan melihat perbandingan tersebut, metode yang ditawarkan penelitian ini unggul pada dataset kesulitan *hard,* namun performa pada kesulitan *easy,* dan *medium* lebih buruk dibandingkan paper lain. Hal ini menunjukan metode pada penelitian ini lebih baik dalam mendeteksi wajah yang berukuran kecil.

4.10. *Transfer learning* dengan dataset mahasiswa TF

Setelah model dilatih dengan dataset WIDER FACE, model akan dilatih menggunakan dataset mahasiswa TF. Tahap ini disebut dengan *transfer learning* karena parameter beban W yang sudah dilatih pada dataset sebelumnya akan ditransfer ke pelatihan model berikutnya. Sehingga model sudah memiliki pengetahuan dasar untuk mendeteksi wajah. Dataset mahasiswa TF yang lebih kecil tidak menjadi masalah karena dataset sekunder WIDER FACE sudah memiliki berbagai macam variasi wajah seperti ekspresi yang sudah dilatih ke model. Pengetahuan mengenai berbagai macam variasi wajah akan dipindahkan ke model ini dan akan diterapkan pada deteksi wajah mahasiswa TF. Pada *transfer learning,* tidak semua *layer* akan dilatih. Karena dataset mahasiswa TF jauh lebih sedikit dari WIDER FACE, hanya layer *classifier* atau *layer* terakhir yang akan dilatih.

Pada pelatihan ini model akan mempelajari 6 kelas berupa 6 mahasiswa yang ada di dataset mahasiswa TF. Model akan mempelajari wajah masing-masing mahasiswa dan akan mencoba memprediksi nama mahasiswa dengan menganalisa

wajah pada gambar. Model akan dilatih menggunakan arsitektur ResNet-50 dengan banyak iterasi sebesar 24 epoch.

Tabel 4. Perbandingan mAP hasil tes menggunakan dataset validasi WIDER FACE dibandingkan dengan metode lain

Paper	mAP(%)			Metode/Arsitektur
	Easy	Medium	Hard	
Metode yang ditawarkan	92.03	92.21	94.16	R-CNN/ ResNet-50
Accurate Face Detection for High Performance [38]	97	96	91.8	AInnoFace/RetinaNet One-Stage CNN
Single-Shot Scale-Aware Network for Real-Time Face Detection [39]	95.3	94.2	88.3	Single-Shot/ResNet-18

Tes akan dilakukan dengan menggunakan video yang diambil dengan 2 jenis resolusi, 720p dan 1080p. Tujuan tes dengan 2 resolusi berbeda adalah untuk menentukan jenis kamera yang akan dipasang saat sistem akan diimplementasikan langsung di kelas. Hasil tes terlihat pada **Tabel 5.**

Tabel 5. Hasil tes pada dataset mahasiswa TF dengan variasi resolusi video 720 dan 1080

Architecture	Dataset	Epoch	mAP (%)	Detection Time(sekon)
ResNet50	TF 720	30	95.02	0.9
ResNet50	TF 1080	30	96.52	1

Dari data hasil tes pada **Tabel 5.** terlihat bahwa variasi resolusi tidak terlalu berpengaruh besar dalam waktu deteksi model. Namun nilai mAP pada resolusi 1080 sedikit lebih tinggi dari pada resolusi 720. Maka disarankan untuk menggunakan kamera dengan resolusi video 1080 untuk mendapatkan performa tertinggi yaitu dengan mAP=96.52 dan waktu deteksi 1 sekon.

4.11. Menyusun sistem pemantauan mahasiswa

Model yang bisa mendeteksi wajah mahasiswa akan digunakan oleh sistem untuk memantau siswa dalam kelas. Fungsi pemanatuan ini meliputi:

a. Mengetahui nama mahasiswa dalam kelas berdasarkan deteksi wajah.
b. Menghitung jumlah mahasiswa dalam kelas.

Fungsi pemantauan lain bisa diterapkan jika mengintegrasikan sistem lain dan pengembangan lebih lanjut, seperti:

a. Membuat catatan waktu yang berkaitan dengan mahasiswa yang dipantau. Seperti waktu mahasiswa masuk atau keluar kelas.
b. Memantau posisi mahasiswa dalam lingkungan kampus jika sistem diintegrasikan ke beberapa kelas atau ruangan lain.
c. Mengetahui pergerakan mahasiswa dalam kelas.

d. Mengetahui posisi duduk mahasiswa dalam kelas.

Sistem ini memiliki faktor-faktor penting agar model bisa mendeteksi wajah dengan baik. Faktor ini adalah performa deteksi *real-time,* GPU yang digunakan untuk deteksi, dan posisi penempatan kamera yang akan memantau mahasiswa.

Dilihat dari waktu deteksi akhir dengan menggunakan arsitektur ResNet-50 yaitu 1 sekon per gambar, maka sistem pemantauan mahasiswa tidak bisa dijalankan secara *real-time.* Agar sistem bisa dijalankan secara *real-time,* waktu deteksi harus sama atau lebih baik dengan *frame rate* kamera yang digunakan. Dalam penelitian ini, kamera yang digunakan memiliki *frame rate* 24 fps. Maka model minimal harus memiliki waktu deteksi 1/24 sekon agar deteksi wajah bisa dilakukan secara *real-time.* Maka proses deteksi harus dilakukan setelah video dalam kelas diambil. Dimana proses deteksi wajah dan pengambilan video dilakukan di waktu yang terpisah.

GPU yang digunakan untuk deteksi adalah Nvidia Tesla K80 dengan RAM 12GB. GPU ini merupakan investasi yang cukup besar dengan harga pasaran hingga Rp.50.000.000,00 per GPU. Alternatif lain adalah dengan menyewa *virtual machine* pada *cloud* dengan GPU yang sama. Pada Google Cloud Platform, harga penyewaan GPU Tesla K80 12 GB adalah $0.135 per jam atau setara dengan Rp.1.887,00 per jam pada saat penulisan penelitian ini. Bila menggunakan *cloud,* maka komputer yang melakukan deteksi harus memiliki koneksi internet.

Analisa video deteksi menunjukan model mengalami kesulitan dalam mendeteksi mahasiswa pada baris belakang. Contoh gambar dari video deteksi terlihat pada **Gambar 4. 19**.

Kemudian akan dilakukan test untuk menentukan posisi kamera. 2 variasi posisi ini adalah sejajar dengan papan tulis *(bottom)* dan posisi kamera diatas papan tulis *(top)*. Perbandingan akurasi mAP dari kedua posisi ini terlihat pada **Tabel 6.**

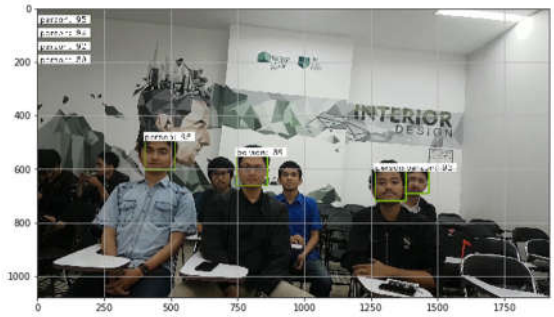
Tabel 6. Perbandingan akurasi mAP antara dua posisi *(Bottom vs Top)*

Posisi kamera	mAP (%)
Bottom	93.11
Top	94.13

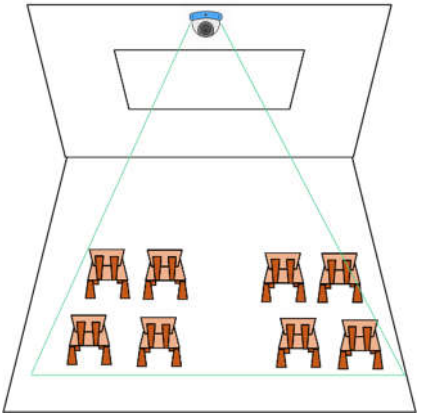
Maka disarankan penempatan kamera dengan elevasi tinggi diatas papan tulis, dimana semua wajah mahasiswa dapat terlihat dengan jelas dan memiliki akurasi lebih tinggi dari posisi sejajar dengan papan tulis, seperti yang terlihat pada **Gambar 4. 20.**

Dengan mempertimbangkan semua faktor tersebut, sistem pemantauan mahasiswa akan memiliki diagram seperti terlihat pada **Gambar 4. 21.**

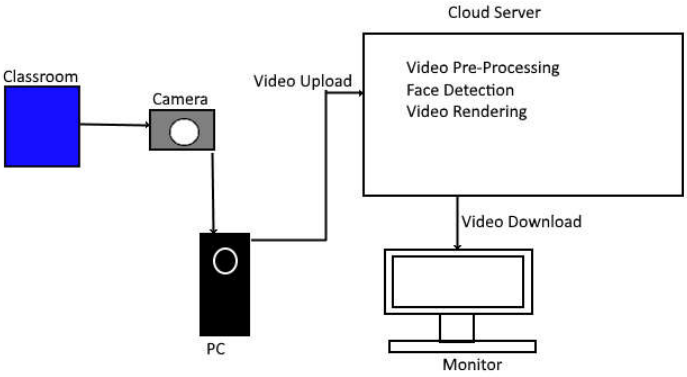
Informasi deteksi mahasiswa dalam kelas bisa ditampilkan dalam text dan di print pada kertas. Informasi ini meliputi nama dan nrp mahasiswa yang terdeteksi di dalam kelas tertentu. Contoh *print out* dalam bentuk text terlihat pada **Gambar 4. 22**. Informasi ini bisa membantu tenaga didik kampus untuk keperluan absensi dan mengecek kehadiran mahasiswa di kelas.



Gambar 4. 19. Contoh hasil deteksi dimana 2 mahasiswa pada baris belakang tidak terdeteksi.



Gambar 4. 20. Penempatan kamera di tempat tinggi agar wajah mahasiswa terlihat jelas.



Gambar 4. 21. Diagram sistem pemantauan mahasiswa dalam kelas

```
Sistem deteksi mahasiswa

Kelas: C-125

Mahasiswa dalam kelas:
Agung - 0231164xxxxxxx
Budi - 0231164xxxxxxx
Chandra - 0231164xxxxxxx
Kelvin - 0231154xxxxxxx
Sapto - 0231154xxxxxxx
Rifki - 0231154xxxxxxx
```

Gambar 4. 22. Contoh *print out* hasil deteksi mahasiswa dalam kelas

Halaman ini sengaja dikosongkan

BAB 5
KESIMPULAN DAN SARAN

5.1. Kesimpulan

Berdasarkan pengolahan data dan pelatihan model yang sudah dilakukan,didapatkan kesimpulan pada tugas akhir sebagai berikut:

a. Pemantauan mahasiswa dalam kelas dengan menggunakan *Face Recognition* berbasis *Deep Neural Network* dilakukan dengan metode *faster* R-CNN. Arsitektur yang digunakan adalah ResNet-50. Dan resolusi kamera yang digunakan adalah 1080p. Metode R-CNN unggul dalam mendeteksi ukuran wajah yang kecil, cocok untuk kondisi dalam kelas. Arsitektur ResNet-50 memiliki akurasi mAP yang lebih tinggi dari VGG-16. Dan resolusi 1080p mendapatkan akurasi lebih tinggi dari 720p karena memuat lebih banyak pixel untuk dideteksi.
b. Waktu deteksi dan akurasi mAP yang didapatkan berdasarkan data tes adalah 1 sekon per gambar dan 96.52% secara berturut-turut.
c. Proses pemantauan mahasiswa dalam kelas tidak bisa dilakukan secara *real-time* karena waktu deteksi model lebih rendah dari *frame rate* kamera yang digunakan. Waktu deteksi harus mencapai 1/24 sekon agar sistem bisa berjalan secara *real-time*. Dengan asumsi kamera mengambil video dengan *frame rate* 24 fps.

5.2. Saran

Adapun saran untuk penelitian lebih lanjut pada tugas akhir ini adalah sebagai berikut:

a. Untuk meningkatkan waktu deteksi agar mencapai performa *real-time,* disarankan menggunakan metode

yang lebih ringan dari R-CNN seperti YOLO atau MobileNet.

b. Menetapkan minimum banyaknya wajah mahasiswa yang dibutuhkan agar model bisa mendeteksi wajah mahasiswa tersebut.
c. *Fine tuning* dengan mengoptimisasi *hyperparameter* menggunakan metode *grid search* atau *random search* untuk meningkatkan performa model.

DAFTAR PUSTAKA

[1] A. L. Samuel, "Some Studies in Machine Learning Using the Game of Checkers," *IBM Journal of Research and Development ,* vol. 3, no. 3, pp. 210 - 229, 1959.

[2] C. M. Bishop, Pattern Recognition and Machine Learning, India: Springer Private Limited, 2013.

[3] K. Mohiuddin, J. Mao and A. K. Jain, Artificial Neural Networks: A Tutorial, Los Alamitos, CA, USA: IEEE Computer Society, 1996.

[4] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE,* vol. 86, no. 11, pp. 2278 - 2324, 1998.

[5] S. Yang, P. Luo, C. C. Loy and X. Tang, "WIDER FACE: A Face Detection Benchmark," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, Nevada, USA, 2016.

[6] F.-F. Li, J. Johnson and S. Yeung, "Image Classification: Data-driven Approach, k-Nearest Neighbor, train/val/test splits," Standford University, 28 February 2017. [Online]. Available: https://intip.in/RRLJ.

[7] F.-F. Li, J. Johnson and S. Yeung, "Linear classification: Support Vector Machine, Softmax," Standford University, 28 February 2017. [Online]. Available: https://intip.in/cCU3.

[8] F.-F. Li, J. Johnson and S. Yeung, "Optimization: Stochastic Gradient Descent," Standford University, 28 February 2017. [Online]. Available: https://intip.in/tGrM.

[9] L. A. d. Santos, "Backpropagation Artificial Inteligence," Gitbooks, [Online]. Available: https://intip.in/Xdhf.

[10] F.-F. Li, J. Johnson and S. Yeung, "Neural Networks Part 1: Setting up the Architecture," Standford University, 28 February 2017. [Online]. Available: https://intip.in/AJdK.

[11] A. Krizhevsky, I. Sutskever and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Network," *COMMUNICATIONS OF THE ACM,* vol. 60, no. 6, pp. 84-90, 2017.

[12] A. Dertat, "Applied Deep Learning - Part 4: Convolutional Neural Networks," Towards Data Science, 8 November 2017. [Online]. Available: https://intip.in/UZGZ.

[13] D. Jakubovitz, R. Giryes and M. R. D. Rodrigues, "Generalization Error in Deep Learning," in *Compressed Sensing and Its Applications*, Cham, Birkhäuser, 2018, pp. pp 153-193.

[14] A. Bhande, "What is underfitting and overfitting in machine learning and how to deal with it.," greyatom, 12 March 2018. [Online]. Available: https://intip.in/L9tz.

[15] A. Dutta, "VGG Image Annotator (VIA)," Visual Geometry Group, 24 October 2019. [Online]. Available: https://intip.in/yhUY.

[16] F.-F. Li, J. Johnson and S. Yeung, "CS231n Online Lecture 08," Standford University, 25 April 2019. [Online]. Available: https://intip.in/1Yic.

[17] F.-F. Li, J. Johnson and S. Yeung, "Neural Networks Part 3: Learning and Evaluation," Stanford University, 28 February 2017. [Online]. Available: https://intip.in/pHwR.

[18] J. L. B. Diederik P. Kingma, "Adam: A Method for Stochastic Optimization," in *3rd International Conference for Learning Representations*, San Diego, 2015.

[19] N. Srivastava, A. Krizhevsky, G. Hinton and I. Sutskever, "Dropout: A Simple Way to Prevent Neural Networks from," *Journal of Machine Learning Research,* vol. 15, no. 1, pp. 1929-1958, 2014.

[20] S. Das, "CNN Architectures: LeNet, AlexNet, VGG, GoogLeNet, ResNet and more…," Medium, 16 November 2017. [Online]. Available: https://intip.in/RbaY.

[21] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of*

Computer Vision (IJCV), vol. 115, no. 3, pp. 211-252, 2015.

[22] F.-F. Li, J. Johnson and S. Yeung, "Transfer Learning and Fine-tuning Convolutional Neural Networks," Stanford University, February 2017. [Online]. Available: https://intip.in/AQ0r.

[23] A. Rosebrock, Deep Learning for Computer Vision with Phyton, Practicioner Bundle, New York: pyimagesearch, 2017.

[24] F.-F. Li, J. Johnson and S. Yeung, "CS231n Online Lecture12," 14 May 2019. [Online]. Available: https://intip.in/RRLJ.

[25] R. Girshick, J. Donahue, T. Darrell and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, Columbus, OH, USA, 2014.

[26] R. Girshick, "Fast R-CNN," in *2015 IEEE International Conference on Computer Vision (ICCV)*, Santiago, Chile, 2015.

[27] S. Ren, K. He, R. Girshick and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Regional Proposal Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 39, no. 6, pp. 1137-1149, 2016.

[28] C. Ping, H. F. Da-Peng and L. Zu-Ying, "Automatic Attendence Face Recognition for Real Classroom

Environments," in *Proceedings of the 2018 2nd International Conference on Big Data and Internet of Things*, Beijing, China, 2018.

[29] S. A. Kalogirou, Solar Energy Engineering, Amsterdam: Elsevier Inc., 2009.

[30] R. Piere, "Deep Blue Sea: Using Deep Learning to Detect Hundreds of Different Plankton Species," Towards Data Science, 12 April 2019. [Online]. Available: https://intip.in/X1LT.

[31] J. Huang, "Speed/accuracy trade-offs for modern convolutional object detectors," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, USA, 2017.

[32] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," in *International Conference on Learning Representations*, San Diego, USA, 2015.

[33] R. Thakur, "Step by step VGG16 implementation in Keras for beginners," Towards Data Science, 6 August 2019. [Online]. Available: https://intip.in/EOqW.

[34] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, 2016.

[35] K. He, "deep-residual-networks," GitHub, 29 July 2016. [Online]. Available: https://intip.in/oKjF.

[36] J. Hui, "mAP (mean Average Precision) for Object Detection," Medium, 7 March 2018. [Online]. Available: https://intip.in/0Rkh.

[37] Y. Xu, "Faster R-CNN (object detection) implemented by Keras for custom data from Google's Open Images Dataset V4," Towards Data Science, 20 November 2018. [Online]. Available: https://intip.in/T0Bj.

[38] F. Zhang, X. Fan, G. Ai, J. Song, Y. Qin and J. Wu, "Accurate Face Detection for High Performance," eprint arXiv:1905.01585, 2019.

[39] S. Zhang, L. Wen, H. Shi, Z. Lei, S. Lyu and S. Li, "Single-Shot Scale-Aware Network for Real-Time Face Detection," *arXiv,* vol. abs/1905.01585, 2019.

LAMPIRAN A

Kode program yang digunakan pada tugas akhir ini menggunakan bahasa pemograman python 3 dengan modul Tensorflow, Keras, Numpy, Scikit-image, OpenCV dan Pandas. Untuk mendapatkan kode secara penuh mohon menghubungi penulis pada email f.devin.ahmad@gmail.com.

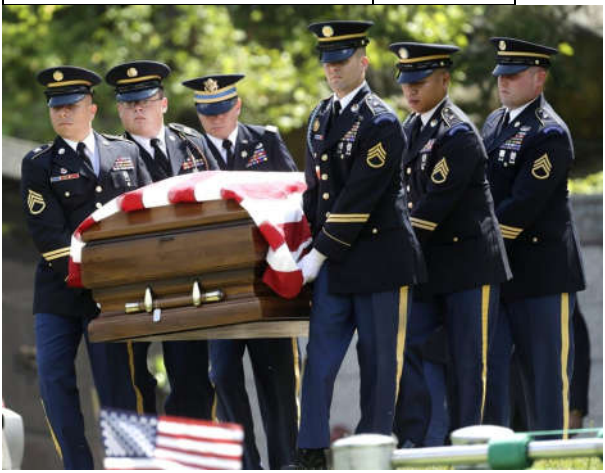
LAMPIRAN B

Dataset sekunder bisa didapat dari website WIDER FACE dengan link: http://shuoyang1213.me/WIDERFACE/. Penelitian ini menggunakan dataset WIDER Face Training Images untuk melatih model dan WIDER Face Validation Images untuk mengetes model. Berikut kategori dataset yang digunakan untuk melatih dan mengetes model beserta tingkat kesulitannya.

Kategori gambar	Kesulitan
0--Parade	Hard
1--Handshaking	Hard
2--Demonstration	Hard
3--Riot	Hard
4--Dancing	Hard
5--Car_Accident	Hard
6--Funeral	Hard
7--Cheering	Hard
8--Election_Campain	Hard
9--Press_Conference	Hard
10--People_Marching	Hard
11--Meeting	Hard
12--Group	Hard
13--Interview	Hard
14--Traffic	Hard
15--Stock_Market	Hard
16--Award_Ceremony	Hard
17--Ceremony	Hard
18--Concerts	Hard

19--Couple	Hard
20--Family_Group	Hard
21--Festival	Medium
22--Picnic	Medium
23--Shoppers	Medium
24--Soldier_Firing	Medium
25--Soldier_Patrol	Medium
26--Soldier_Drilling	Medium
27--Spa	Medium
28--Sports_Fan	Medium
29--Students_Schoolkids	Medium
30--Surgeons	Medium
31--Waiter_Waitress	Medium
32--Worker_Laborer	Medium
33--Running	Medium
34--Baseball	Medium
35--Basketball	Medium
36--Football	Medium
37--Soccer	Medium
38--Tennis	Medium
39--Ice_Skating	Medium
40--Gymnastics	Medium
41--Swimming	Easy
42--Car_Racing	Easy
43--Row_Boat	Easy
44--Aerobics	Easy
45--Balloonist	Easy
46--Jockey	Easy

47--Matador_Bullfighter	Easy
48--Parachutist_Paratrooper	Easy
49--Greeting	Easy
50--Celebration_Or_Party	Easy
51--Dresses	Easy
52--Photographers	Easy
53--Raid	Easy
54--Rescue	Easy
55--Sports_Coach_Trainer	Easy
56--Voter	Easy
57--Angler	Easy
58--Hockey	Easy
59--people--driving--car	Easy
61--Street_Battle	Easy

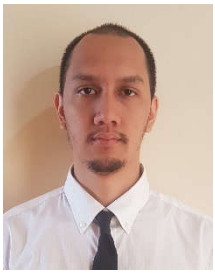


Contoh gambar dari kategori 6--Funeral

Contoh gambar dari kategori 29--Students_Schoolkids

Contoh gambar dari kategori 61--Street_Battle

Biodata Penulis



Devin Ahmad Febrian lahir pada 28 Februari 1997 di Bogor. Penulis menempuh pendidikan formal pada SMA Marsudirini Parung kemudian melanjutkan pendidikan pada program studi sarjana pada Departemen Teknik Fisika ITS.

Pada bulan Januari 2020 penulis menyelesaikan Tugas Akhir dengan judul **Sistem Pemantauan Mahasiswa Dalam Kelas dengan *Face Recognition* berbasis *Deep Neural Network.*** Bagi pembaca yang memiliki kritik, saran maupun ingin berdiskusi lebih lanjut mengenai penelitian tugas akhir ini dapat menghubungi penulis pada email f.devin.ahmad@gmail.com.