

**ANALISIS OPTIMIZER PADA ARSITEKTUR NEURAL
NETWORK UNTUK MENINGKATKAN AKURASI
PENGENALAN WAJAH**

TUGAS AKHIR

Hanjaya Suryalim

1118011



**PROGRAM STUDI INFORMATIKA
INSTITUT TEKNOLOGI HARAPAN BANGSA
BANDUNG
2022**

**ANALISIS OPTIMIZER PADA ARSITEKTUR NEURAL
NETWORK UNTUK MENINGKATKAN AKURASI
PENGENALAN WAJAH**

TUGAS AKHIR

**Diajukan sebagai salah satu syarat untuk memperoleh
gelar sarjana dalam bidang Informatika**

**Hanjaya Suryalim
1118011**



**PROGRAM STUDI INFORMATIKA
INSTITUT TEKNOLOGI HARAPAN BANGSA
BANDUNG
2022**

HALAMAN PERNYATAAN ORISINALITAS

**Saya menyatakan bahwa Tugas Akhir yang saya susun ini
adalah hasil karya saya sendiri.**

**Semua sumber yang dikutip maupun dirujuk
telah saya nyatakan dengan benar.**

**Saya bersedia menerima sanksi pencabutan gelar akademik
apabila di kemudian hari Tugas Akhir ini terbukti plagiat.**

Bandung, 2 Juni 2021



**Hanjaya Suryalim
1118011**

HALAMAN PENGESAHAN TUGAS AKHIR

Tugas Akhir dengan judul:

**ANALISIS OPTIMIZER PADA ARSITEKTUR NEURAL NETWORK UNTUK
MENINGKATKAN AKURASI PENGENALAN WAJAH**

yang disusun oleh:

Hanjaya Suryalim

1118011

telah berhasil dipertahankan di hadapan Dewan Penguji Sidang Tugas Akhir yang dilaksanakan pada:

Hari / tanggal : Rabu, 2 Juni 2021

Waktu : Jam 07.00 WIB

Menyetujui,

Pembimbing Utama:

Pembimbing Pendamping:

Ken Ratri Retno Wardani, S.Kom,M.T
105033

Dr. Hery Heryanto, M.Kom.
116007

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI

TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademik Institut Teknologi Harapan Bangsa, saya yang bertanda tangan di bawah ini:

Nama : Hanjaya Suryalim

NIM : 1118011

Program Studi : Informatika

demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Institut Teknologi Harapan Bangsa **Hak Bebas Royalti Noneksklusif (Non-exclusive Royalty Free Rights)** atas karya ilmiah saya yang berjudul:

ANALISIS OPTIMIZER PADA ARSITEKTUR NEURAL NETWORK UNTUK MENINGKATKAN AKURASI PENGENALAN WAJAH

beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini Institut Teknologi Harapan Bangsa berhak menyimpan, mengalihmediakan, mengelola dalam pangkalan data, dan memublikasikan karya ilmiah saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Bandung, 2 Juni 2021

Yang menyatakan



Hanjaya Suryalim

ABSTRAK

Nama : Hanjaya Suryalim
Program Studi : Informatika
Judul : Menguji *hyperparameter* pada Arsitektur *Neural Network* untuk Meningkatkan Akurasi Pengenalan Wajah

Pitch adalah interpretasi *fundamental frequency* f_0 aliran audio akustik yang membawa karakteristik melodi musik. Pembawaan karya musik yang baik ditentukan dari ketepatan *pitch* yang menentukan ketepatan intonasi. Maka dari itu, untuk menentukan kualitas intonasi penampilan suatu karya musik, estimasi *pitch* perlu dilakukan. Metode yang digunakan untuk melakukan estimasi *pitch* adalah *convolutional neural network* (CNN). Setelah estimasi *pitch* dilakukan, digunakan metode *dynamic time warping* (DTW) untuk melakukan pengujian kemiripan (dalam *distance*) hasil estimasi *pitch* dengan *template* rekaman dari *dataset* untuk menentukan ketepatan intonasi. *Preprocessing* yang dilakukan adalah pembulatan *pitch label*, pembuatan spektogram dari rekaman, *scaling*, dan *recolor* spektogram. Metode CNN untuk melakukan estimasi *pitch* diuji menggunakan lima lagu dari *dataset* MIR-QBSH. Pengujian CNN dilakukan dengan menerapkan empat rancangan arsitektur dengan mengombinasikan nilai *epoch*, *learning rate*, jumlah *filter* pada setiap *convolutional layer*, dan jumlah konvolusi untuk mencari kombinasi terbaik yang menghasilkan akurasi tertinggi. Berdasarkan hasil pengujian, CNN yang dibangun dapat menghasilkan rata-rata akurasi tertinggi sebesar 97.425% dengan selisih antara rata-rata akurasi dan rata-rata akurasi validasi sebesar 14.383%. Nilai *threshold* yang optimal untuk *distance* berada pada rentang 1000-1500.

Kata kunci: Suara nyanyian, estimasi *pitch*, *fundamental frequency*, pengujian kemiripan, *convolutional neural network* (CNN), *dynamic time warping* (DTW).

ABSTRACT

Name : Dionisius Pratama
Department : Informatics
Title : Application of Convolutional Neural Network to Estimate Pitch from Singing Voice Recording

Pitch is an interpretation of fundamental frequency f_0 of an acoustic audio stream that carries melodic characteristics of music. A good musical performance is determined by the accuracy of the pitch which determines intonation accuracy. Therefore, to determine intonation quality of a musical performance, it is necessary to estimate the pitch. The method used to estimate pitch is convolutional neural network (CNN). After pitch estimation is done, dynamic time warping (DTW) method is used to test the similarity (measured in distance) of pitch estimation results with the recording template from the dataset to determine intonation accuracy. The preprocessing carried out is rounding pitch label, making spectrograms from recording, scaling, and recoloring spectrogram. The CNN method for performing pitch estimation is tested using five songs from the MIR-QBSH dataset. CNN testing is done by applying four architectural designs by combining epoch values, learning rate, number of filters in each convolutional layer, and number of convolutions to find the best combination that produces the highest accuracy. Based on the test results, the CNN built can produce the highest average accuracy of 97.425% with the difference between the average accuracy and the average validation accuracy of 14.383%. The optimal threshold value for distance is in the range of 1000-1500.

Keywords: Singing voice, pitch estimation, fundamental frequency, similarity matching, convolutional neural network (CNN), dynamic time warping (DTW).

KATA PENGANTAR

Penulis mengucapkan terima kasih kepada Tuhan Yesus karena dengan karunia-Nya, penulis dapat menyelesaikan Tugas Akhir berjudul "Penerapan *Convolutional Neural Network* untuk Melakukan Estimasi *Pitch* pada Rekaman Suara Penyanyi" dengan baik. Penelitian ini disusun sebagai salah satu syarat kelulusan Program Studi Informatika di Institut Teknologi Harapan Bangsa. Lebih dari itu, penulis ingin mencoba menghubungkan ilmu yang telah didapatkan selama masa perkuliahan ke dalam keilmuan musik. Perkembangan teknologi sekarang, khususnya *deep learning*, telah memungkinkan pengolahan informasi musik yang beragam. Musik juga merupakan bagian integral dari hidup banyak orang, sehingga bagi penulis, musik menjadi objek yang menarik untuk diteliti.

Selama penyusunan Tugas Akhir ini, banyak dinamika yang dialami oleh penulis yang tidak dapat dijalani sendirian. Oleh karena itu, penulis ingin menyampaikan terima kasih yang sebesar-besarnya kepada:

1. Tuhan Yesus dan Bunda Maria, karena bimbingan dan karunia-Nya, penulis selalu mendapat pengharapan, kebahagiaan, serta pengalaman berharga selama pembuatan Tugas Akhir ini.
2. Bapak Dr. Hery Heryanto, M.Kom., selaku pembimbing utama Tugas Akhir yang senantiasa memberi dukungan, semangat, waktu bimbingan yang sangat membuka wawasan, ilmu, serta saran yang membangun kepada penulis selama pembuatan Tugas Akhir ini.
3. Bapak Hans Christian Kurniawan, M.T., selaku pembimbing pendamping Tugas Akhir yang senantiasa memberi dukungan, semangat, ilmu, saran, dan kesempatan menjadi Asisten Dosen selama penyusunan Tugas Akhir ini.
4. Bapak Yoyok Yusman Gamaliel, S.T., M.Eng., selaku penguji I Tugas Akhir yang telah memberikan pengujian serta masukan-masukan kritis kepada penulis selama pembuatan Tugas Akhir ini.
5. Bapak Dr. Tjong Wan Sen, M.T., selaku penguji II Tugas Akhir yang telah memberikan pengujian serta masukan-masukan kritis kepada penulis selama pembuatan Tugas Akhir ini.
6. Dosen dan *staff* Program Studi Informatika serta DAAK ITHB yang telah membantu menyelesaikan proses administrasi sehingga Tugas Akhir ini dapat diselesaikan.
7. Kedua orang tua yang menyediakan waktu untuk memberikan doa, semangat,

dan dukungan kepada penulis dalam menyelesaikan Tugas Akhir ini.

8. Jessica Cecilia Budianto, S.Kom., Patrick Nicholas Hadinata, S.T., yang tidak bosan-bosan menerima banyak pertanyaan soal penulisan dokumen, analisis *source code*, waktu diskusi bahkan sampai tengah malam, dan banyak hal lain yang menginspirasi penulis selama melakukan Tugas Akhir ini.
9. Januard Benedictus, Michelle Natasha Irawan, dan Christophorus Stanley. Terima kasih untuk bantuan dalam merapikan dokumen dalam bagian visualisasi data dan ide desain.
10. Kakak kelas Informatika ITHB, khususnya: Edwin, Jovin, Ivan, Axel, Jesslyn, William, Vincenlie. Terima kasih atas inspirasi, menjadi tempat bercerita, dan tips menghadapi dan memperjuangkan Tugas Akhir.
11. Teman bimbingan: Christian, Ricky, Adrian, Nicholas, Nael. Terima kasih untuk waktu bimbingan bersama yang memberikan semangat bagi penulis untuk menyelesaikan Tugas Akhir ini.
12. Teman *Discord*: Agape, Gian, Johann, Kekey, Andreas, Evan, Jacob, Jason, Davin. Terima kasih sudah menemani penulis dan mendengarkan banyak hal selama penyelesaian Tugas Akhir ini.
13. Teman seangkatan, adik dan kakak kelas yang berkuliahan di satu tempat dan berbeda tempat, yang telah memberikan dukungan, bantuan teknis, semangat, dan waktu diskusi kepada penulis selama melakukan Tugas Akhir ini.

Penulis menyadari bahwa Tugas Akhir ini masih jauh dari sempurna, baik dari segi konten maupun penulisan, karena keterbatasan waktu, pengetahuan, dan pengalaman yang dimiliki oleh penulis. Penulis menyampaikan permohonan maaf atas hal tersebut. Oleh karena itu, kritik dan saran yang membangun sangat diharapkan demi penyempurnaan penelitian sejenis di masa mendatang.

Akhir kata, semoga Tugas Akhir ini dapat membantu para pembaca yang ingin mengetahui contoh penerapan bidang keilmuan Informatika yang diintegrasikan dengan musik serta bagi siapa saja yang membutuhkan atau sekedar ingin mengetahui. Semoga apresiasi musik senantiasa membawa hidup pembaca lebih berwarna dan indah.

Bandung, 2 Juni 2021

Hormat penulis,



HANJAYA SURYALIM

DAFTAR ISI

ABSTRAK	iv
ABSTRACT	v
KATA PENGANTAR	vi
DAFTAR ISI	viii
DAFTAR TABEL	xiii
DAFTAR GAMBAR	xviii
DAFTAR ALGORITME	xxiv
DAFTAR LAMPIRAN	xxiv
BAB 1 PENDAHULUAN	1-1
1.1 Latar Belakang	1-1
1.2 Rumusan Masalah	1-3
1.3 Tujuan Penelitian	1-3
1.4 Batasan Masalah	1-4
1.5 Kontribusi Penelitian	1-4
1.6 Metode Penelitian	1-4
1.7 Sistematika Pembahasan	1-5
BAB 2 LANDASAN TEORI	2-1
2.1 Tinjauan Pustaka	2-1
2.1.1 <i>Neural Network</i>	2-1
2.1.1.1 <i>Perceptron</i>	2-1
2.1.1.2 <i>Deep Learning</i>	2-3
2.1.2 <i>Convolutional Neural Network</i>	2-5
2.1.2.1 <i>Convolution Layer</i>	2-6
2.1.2.2 <i>Pooling Layer</i>	2-13
2.1.2.3 <i>Fully Connected Layer</i>	2-14
2.1.3 <i>Backward Propagation</i>	2-15
2.1.3.1 <i>Gradient Descent</i>	2-17
2.1.3.2 <i>Cost Function</i>	2-18

2.1.4	<i>Arsitektur CNN</i>	2-23
2.1.4.1	<i>VGG-16</i>	2-23
2.1.4.2	<i>Inception</i>	2-25
2.1.5	<i>Fungsi Aktivasi</i>	2-29
2.1.5.1	<i>Rectified Linear Unit (ReLU)</i>	2-29
2.1.5.2	<i>Softmax</i>	2-30
2.1.6	<i>Weight Initialization</i>	2-31
2.1.6.1	<i>He Initialization</i>	2-31
2.1.7	<i>Regularization</i>	2-32
2.1.7.1	<i>L2</i>	2-32
2.1.7.2	<i>Batch Normalization Layer</i>	2-34
2.1.7.3	<i>Dropout Layer</i>	2-35
2.1.8	<i>Optimizers</i>	2-36
2.1.8.1	<i>Stochastic Gradient Descent</i>	2-36
2.1.8.2	<i>Nesterov Accelerated Gradient (NAG)</i>	2-39
2.1.8.3	<i>Adagrad</i>	2-42
2.1.8.4	<i>Adadelta</i>	2-43
2.1.8.5	<i>Adam</i>	2-46
2.1.9	<i>Image Preprocessing</i>	2-48
2.1.9.1	<i>Image Interpolation</i>	2-48
2.1.9.2	<i>Aspect Aware Preprocessing</i>	2-49
2.1.9.3	<i>Image Augmentation</i>	2-50
2.1.9.4	<i>LAB - colorspaces</i>	2-57
2.1.9.5	<i>Histogram Equalization (HE)</i>	2-59
2.1.9.6	<i>Adaptive Histogram Equalization (AHE)</i>	2-60
2.1.9.7	<i>Contrast Limited Adaptive Histogram Equalization (CLAHE)</i>	2-60
2.1.9.8	<i>Modified Contrast Limited Adaptive Histogram Equalization (MCLAHE)</i>	2-62
2.1.10	<i>Confusion Matrix</i>	2-62
2.1.11	<i>Pustaka</i>	2-63
2.1.11.1	<i>TensorFlow</i>	2-64
2.1.11.2	<i>Keras</i>	2-66
2.1.11.3	<i>OpenCV</i>	2-72
2.1.11.4	<i>NumPy</i>	2-73
2.1.11.5	<i>OS</i>	2-74
2.1.11.6	<i>Matplotlib</i>	2-75

2.1.11.7	<i>Imutils</i>	2-76
2.1.11.8	<i>Sklearn</i>	2-77
2.2	Tinjauan Studi	2-79
2.2.1	<i>State of the Art</i>	2-79
2.2.2	Pembahasan penelitian terkait	2-81
2.3	Tinjauan Objek	2-81
2.3.1	<i>Cropped Extended Yale Face Database</i>	2-82
2.3.2	<i>Komnet Dataset</i>	2-82
BAB 3	ANALISIS DAN PERANCANGAN SISTEM	3-1
3.1	Analisis Masalah	3-1
3.2	Kerangka Pemikiran	3-2
3.3	Analisis Urutan Proses Global	3-4
3.3.1	Pemrosesan awal	3-5
3.3.2	Proses <i>Training</i> (Pelatihan)	3-7
3.3.2.1	<i>Forward Propagation Convolutional Neural Network</i>	3-9
3.3.2.2	<i>Backward Propagation Convolutional Neural Network</i>	3-10
3.3.3	Proses <i>Testing</i> (Pengujian)	3-11
3.4	Analisis Manual	3-12
3.4.1	<i>Dataset</i>	3-12
3.4.1.1	<i>Aspect Aware Preprocessor and Image to Array preprocessor</i>	3-13
3.4.1.2	<i>Modified Contrast Limited Adaptive Histogram Equalization</i>	3-13
3.4.2	<i>Forward Propagation</i>	3-14
3.4.2.1	<i>VGG16</i>	3-14
3.4.2.2	<i>Inception</i>	3-23
3.4.3	<i>Backward Propagation</i>	3-31
3.4.3.1	<i>Stochastic Gradient Descent</i>	3-31
3.4.3.2	<i>Nesterov Accelerated Gradient</i>	3-32
3.4.3.3	<i>Adagrad</i>	3-33
3.4.3.4	<i>Adadelta</i>	3-34
3.4.3.5	<i>Adam</i>	3-36
BAB 4	IMPLEMENTASI DAN PENGUJIAN	4-1
4.1	Lingkungan Implementasi	4-1

4.1.1	Spesifikasi Perangkat Keras	4-1
4.1.2	Lingkungan Perangkat Lunak	4-1
4.2	Implementasi Perangkat Lunak	4-1
4.2.1	Penggunaan Jupyter Lab	4-1
4.2.2	Penggunaan Google Colaboratory	4-2
4.2.2.1	Class VGG16	4-3
4.2.2.2	Class Inception	4-3
4.2.3	Implementasi Penggunaan <i>Dataset</i>	4-3
4.3	Implementasi Aplikasi	4-4
4.4	Pengujian	4-11
4.4.1	Pengujian Nilai <i>Epoch</i>	4-11
4.4.2	Pengujian Nilai <i>Learning Rate</i>	4-11
4.4.3	Pengujian Jumlah Konvolusi dan <i>Filter</i>	4-12
4.4.4	Pengujian Nilai <i>Threshold</i>	4-18
4.5	Hasil Pengujian	4-18
4.5.1	Pengujian pada Arsitektur 1	4-19
4.5.1.1	Nilai <i>Epoch</i> = 50	4-19
4.5.1.2	Nilai <i>Epoch</i> = 100	4-20
4.5.1.3	Nilai <i>Epoch</i> = 150	4-22
4.5.1.4	Nilai <i>Epoch</i> = 200	4-23
4.5.1.5	Nilai <i>Epoch</i> = 250	4-25
4.5.1.6	Nilai <i>Epoch</i> = 300	4-26
4.5.1.7	Tabel Hasil Pengujian pada Arsitektur 1	4-28
4.5.2	Pengujian pada Arsitektur 2	4-35
4.5.2.1	Nilai <i>Epoch</i> = 50	4-35
4.5.2.2	Nilai <i>Epoch</i> = 100	4-37
4.5.2.3	Nilai <i>Epoch</i> = 150	4-38
4.5.2.4	Nilai <i>Epoch</i> = 200	4-40
4.5.2.5	Nilai <i>Epoch</i> = 250	4-41
4.5.2.6	Nilai <i>Epoch</i> = 300	4-43
4.5.2.7	Tabel Hasil Pengujian pada Arsitektur 2	4-44
4.5.3	Pengujian pada Arsitektur 3	4-51
4.5.3.1	Nilai <i>Epoch</i> = 50	4-51
4.5.3.2	Nilai <i>Epoch</i> = 100	4-53
4.5.3.3	Nilai <i>Epoch</i> = 150	4-54
4.5.3.4	Nilai <i>Epoch</i> = 200	4-56
4.5.3.5	Nilai <i>Epoch</i> = 250	4-57

4.5.3.6	Nilai <i>Epoch</i> = 300	4-59
4.5.3.7	Tabel Hasil Pengujian pada Arsitektur 3	4-60
4.5.4	Pengujian pada Arsitektur 4	4-67
4.5.4.1	Nilai <i>Epoch</i> = 50	4-67
4.5.4.2	Nilai <i>Epoch</i> = 100	4-69
4.5.4.3	Nilai <i>Epoch</i> = 150	4-70
4.5.4.4	Nilai <i>Epoch</i> = 200	4-72
4.5.4.5	Nilai <i>Epoch</i> = 250	4-73
4.5.4.6	Nilai <i>Epoch</i> = 300	4-75
4.5.4.7	Tabel Hasil Pengujian pada Arsitektur 4	4-76
4.5.5	Pembahasan Umum Hasil Pengujian Arsitektur CNN	4-83
4.5.6	Hasil Pengujian <i>Threshold</i> dengan Metode DTW	4-84
4.5.6.1	Pengujian pada Model Pertama	4-85
4.5.6.2	Pengujian pada Model Kedua	4-86
4.5.7	Pembahasan Hasil Pengujian <i>Threshold</i>	4-87
4.6	Analisis Kesalahan	4-98
BAB 5	KESIMPULAN DAN SARAN	5-1
5.1	Kesimpulan	5-1
5.2	Saran	5-2
DAFTAR REFERENSI	i	
LAMPIRANA		
	Nomor Not MIDI, Nama Not, dan <i>Center Frequency</i>	A-1
LAMPIRANB		
	Hasil Pengujian Arsitektur <i>Convolutional Neural Network</i>	B-1
LAMPIRANC		
	Hasil Pengujian <i>Distance</i> dengan <i>Dynamic Time Warping</i> dan <i>Equivalent Pitch Class</i>	C-1
LAMPIRAND		
	Perbandingan Hasil Estimasi <i>Pitch</i> dengan <i>Dataset</i>	D-1

DAFTAR TABEL

2.1	<i>Confusion matrix</i>	2-62
2.2	Daftar <i>method</i> yang digunakan dari pustaka TensorFlow	2-64
2.3	Daftar <i>method</i> yang digunakan dari pustaka Keras	2-66
2.4	Daftar <i>method</i> yang digunakan dari pustaka OpenCV	2-72
2.5	Daftar <i>method</i> yang digunakan dari pustaka NumPy	2-74
2.6	Daftar <i>method</i> yang digunakan dari pustaka OS	2-74
2.7	Daftar <i>method</i> yang digunakan dari pustaka matplotlib	2-75
2.8	Daftar <i>method</i> yang digunakan dari pustaka imutils	2-76
2.9	Daftar <i>method</i> yang digunakan dari pustaka sklearn	2-77
2.10	<i>State of the Art</i>	2-80
2.10	<i>State of the Art</i>	2-81
3.1	Contoh matriks citra	3-15
3.2	Contoh matriks citra dengan <i>padding</i>	3-16
3.3	Contoh nilai <i>kernel</i> 3×3	3-16
3.4	Contoh nilai <i>feature map</i> hasil konvolusi	3-17
3.5	Contoh nilai <i>feature map</i> setelah diaktivasi	3-18
3.6	Contoh <i>feature map</i> hasil perhitungan <i>batch normalization</i>	3-19
3.7	Contoh hasil perhitungan <i>max pooling</i>	3-20
3.8	Contoh inisialisasi <i>random number</i> proses <i>dropout</i>	3-20
3.9	Contoh <i>random number</i> proses <i>dropout</i>	3-21
3.10	Contoh <i>kernel</i> pada lapisan <i>dense</i>	3-21
4.1	Daftar <i>method</i> pada <i>Jupyter lab</i>	4-2
4.2	Daftar <i>method</i> pada <i>class VGG16</i>	4-3
4.3	Perincian Penggunaan <i>Dataset</i> untuk Implementasi	4-4
4.4	Hasil Pengujian (nilai rata-rata) pada Arsitektur 1	4-28
4.4	Hasil Pengujian (nilai rata-rata) pada Arsitektur 1	4-29
4.5	Hasil Pengujian (nilai maksimal) pada Arsitektur 1	4-29
4.5	Hasil Pengujian (nilai maksimal) pada Arsitektur 1	4-30
4.6	Hasil Pengujian (nilai rata-rata) pada Arsitektur 2	4-44
4.6	Hasil Pengujian (nilai rata-rata) pada Arsitektur 2	4-45
4.7	Hasil Pengujian (nilai maksimal) pada Arsitektur 2	4-45
4.7	Hasil Pengujian (nilai maksimal) pada Arsitektur 2	4-46
4.8	Hasil Pengujian (nilai rata-rata) pada Arsitektur 3	4-60

4.8	Hasil Pengujian (nilai rata-rata) pada Arsitektur 3	4-61
4.9	Hasil Pengujian (nilai maksimal) pada Arsitektur 3	4-61
4.9	Hasil Pengujian (nilai maksimal) pada Arsitektur 3	4-62
4.10	Hasil Pengujian (nilai rata-rata) pada Arsitektur 4	4-76
4.10	Hasil Pengujian (nilai rata-rata) pada Arsitektur 4	4-77
4.11	Hasil Pengujian (nilai maksimal) pada Arsitektur 4	4-77
4.11	Hasil Pengujian (nilai maksimal) pada Arsitektur 4	4-78
4.12	Hasil Pengujian <i>Distance</i> dengan Model Pertama	4-85
4.13	Hasil Pengujian <i>Threshold</i> pada Model Pertama	4-86
4.14	Hasil Pengujian <i>Distance</i> dengan Model Kedua	4-86
4.15	Hasil Pengujian <i>Threshold</i> pada Model Kedua	4-87
4.16	Hasil Perbandingan <i>Pitch</i> pada lagu Twinkle-Twinkle Little Star di Model Pertama	4-89
4.16	Hasil Perbandingan <i>Pitch</i> pada lagu Twinkle-Twinkle Little Star di Model Pertama	4-90
4.16	Hasil Perbandingan <i>Pitch</i> pada lagu Twinkle-Twinkle Little Star di Model Pertama	4-91
4.17	Hasil Perbandingan <i>Pitch</i> pada lagu Twinkle-Twinkle Little Star di Model Kedua	4-91
4.17	Hasil Perbandingan <i>Pitch</i> pada lagu Twinkle-Twinkle Little Star di Model Kedua	4-92
4.17	Hasil Perbandingan <i>Pitch</i> pada lagu Twinkle-Twinkle Little Star di Model Kedua	4-93
4.18	Hasil Perbandingan <i>Pitch</i> pada lagu Old McDonald Had a Farm di Model Pertama	4-94
4.18	Hasil Perbandingan <i>Pitch</i> pada lagu Old McDonald Had a Farm di Model Pertama	4-95
4.19	Hasil Perbandingan <i>Pitch</i> pada lagu Old McDonald Had a Farm di Model Kedua	4-96
4.19	Hasil Perbandingan <i>Pitch</i> pada lagu Old McDonald Had a Farm di Model Kedua	4-97
A	Nomor not MIDI, nama not, dan <i>center frequency</i>	A-1
A	Nomor not MIDI, nama not, dan <i>center frequency</i>	A-2
A	Nomor not MIDI, nama not, dan <i>center frequency</i>	A-3
A	Nomor not MIDI, nama not, dan <i>center frequency</i>	A-4
A	Nomor not MIDI, nama not, dan <i>center frequency</i>	A-5

C-1	Hasil pengujian <i>distance</i> pada lagu Twinkle-Twinkle Little Star di model pertama	C-1
C-1	Hasil pengujian <i>distance</i> pada lagu Twinkle-Twinkle Little Star di model pertama	C-2
C-2	Hasil pengujian <i>distance</i> pada lagu Old McDonald di model pertama	C-3
C-2	Hasil pengujian <i>distance</i> pada lagu Old McDonald di model pertama	C-4
C-3	Hasil pengujian <i>distance</i> pada lagu Happy Birthday di model pertama	C-4
C-3	Hasil pengujian <i>distance</i> pada lagu Happy Birthday di model pertama	C-5
C-3	Hasil pengujian <i>distance</i> pada lagu Happy Birthday di model pertama	C-6
C-4	Hasil pengujian <i>distance</i> pada lagu Brother John di model pertama .	C-6
C-4	Hasil pengujian <i>distance</i> pada lagu Brother John di model pertama .	C-7
C-5	Hasil pengujian <i>distance</i> pada lagu London Bridge is Falling Down di model pertama	C-8
C-5	Hasil pengujian <i>distance</i> pada lagu London Bridge is Falling Down di model pertama	C-9
C-6	Hasil pengujian <i>distance</i> pada lagu Twinkle-Twinkle Little Star di model kedua	C-9
C-6	Hasil pengujian <i>distance</i> pada lagu Twinkle-Twinkle Little Star di model kedua	C-10
C-7	Hasil pengujian <i>distance</i> pada lagu Old McDonald di model kedua .	C-11
C-7	Hasil pengujian <i>distance</i> pada lagu Old McDonald di model kedua .	C-12
C-8	Hasil pengujian <i>distance</i> pada lagu Happy Birthday di model kedua	C-12
C-8	Hasil pengujian <i>distance</i> pada lagu Happy Birthday di model kedua	C-13
C-8	Hasil pengujian <i>distance</i> pada lagu Happy Birthday di model kedua	C-14
C-9	Hasil pengujian <i>distance</i> pada lagu Brother John di model kedua .	C-14
C-9	Hasil pengujian <i>distance</i> pada lagu Brother John di model kedua .	C-15
C-10	Hasil pengujian <i>distance</i> pada lagu London Bridge is Falling Down di model kedua	C-15
C-10	Hasil pengujian <i>distance</i> pada lagu London Bridge is Falling Down di model kedua	C-16
C-10	Hasil pengujian <i>distance</i> pada lagu London Bridge is Falling Down di model kedua	C-17
D-1	Hasil Perbandingan <i>Pitch</i> pada lagu Twinkle-Twinkle Little Star di Model Pertama	D-2
D-1	Hasil Perbandingan <i>Pitch</i> pada lagu Twinkle-Twinkle Little Star di Model Pertama	D-3

D-1	Hasil Perbandingan <i>Pitch</i> pada lagu Twinkle-Twinkle Little Star di Model Pertama	D-4
D-2	Hasil Perbandingan <i>Pitch</i> pada lagu Old McDonald Had a Farm di Model Pertama	D-5
D-2	Hasil Perbandingan <i>Pitch</i> pada lagu Old McDonald Had a Farm di Model Pertama	D-6
D-2	Hasil Perbandingan <i>Pitch</i> pada lagu Old McDonald Had a Farm di Model Pertama	D-7
D-3	Hasil Perbandingan <i>Pitch</i> pada lagu Happy Birthday di Model Pertama	D-8
D-3	Hasil Perbandingan <i>Pitch</i> pada lagu Happy Birthday di Model Pertama	D-9
D-3	Hasil Perbandingan <i>Pitch</i> pada lagu Happy Birthday di Model Pertama	D-10
D-4	Hasil Perbandingan <i>Pitch</i> pada lagu Brother John di Model Pertama	D-11
D-4	Hasil Perbandingan <i>Pitch</i> pada lagu Brother John di Model Pertama	D-12
D-4	Hasil Perbandingan <i>Pitch</i> pada lagu Brother John di Model Pertama	D-13
D-4	Hasil Perbandingan <i>Pitch</i> pada lagu Brother John di Model Pertama	D-14
D-5	Hasil Perbandingan <i>Pitch</i> pada lagu London Bridge is Falling Down di Model Pertama	D-14
D-5	Hasil Perbandingan <i>Pitch</i> pada lagu London Bridge is Falling Down di Model Pertama	D-15
D-5	Hasil Perbandingan <i>Pitch</i> pada lagu London Bridge is Falling Down di Model Pertama	D-16
D-6	Hasil Perbandingan <i>Pitch</i> pada lagu Twinkle-Twinkle Little Star di Model Kedua	D-16
D-6	Hasil Perbandingan <i>Pitch</i> pada lagu Twinkle-Twinkle Little Star di Model Kedua	D-17
D-6	Hasil Perbandingan <i>Pitch</i> pada lagu Twinkle-Twinkle Little Star di Model Kedua	D-18
D-6	Hasil Perbandingan <i>Pitch</i> pada lagu Twinkle-Twinkle Little Star di Model Kedua	D-19
D-7	Hasil Perbandingan <i>Pitch</i> pada lagu Old McDonald Had a Farm di Model Kedua	D-19
D-7	Hasil Perbandingan <i>Pitch</i> pada lagu Old McDonald Had a Farm di Model Kedua	D-20
D-7	Hasil Perbandingan <i>Pitch</i> pada lagu Old McDonald Had a Farm di Model Kedua	D-21
D-7	Hasil Perbandingan <i>Pitch</i> pada lagu Old McDonald Had a Farm di Model Kedua	D-22

D-8	Hasil Perbandingan <i>Pitch</i> pada lagu Happy Birthday di Model Kedua	D-22
D-8	Hasil Perbandingan <i>Pitch</i> pada lagu Happy Birthday di Model Kedua	D-23
D-8	Hasil Perbandingan <i>Pitch</i> pada lagu Happy Birthday di Model Kedua	D-24
D-8	Hasil Perbandingan <i>Pitch</i> pada lagu Happy Birthday di Model Kedua	D-25
D-9	Hasil Perbandingan <i>Pitch</i> pada lagu Brother John di Model Kedua	D-25
D-9	Hasil Perbandingan <i>Pitch</i> pada lagu Brother John di Model Kedua	D-26
D-9	Hasil Perbandingan <i>Pitch</i> pada lagu Brother John di Model Kedua	D-27
D-9	Hasil Perbandingan <i>Pitch</i> pada lagu Brother John di Model Kedua	D-28
D-10	Hasil Perbandingan <i>Pitch</i> pada lagu London Bridge is Falling Down di Model Kedua	D-29
D-10	Hasil Perbandingan <i>Pitch</i> pada lagu London Bridge is Falling Down di Model Kedua	D-30
D-10	Hasil Perbandingan <i>Pitch</i> pada lagu London Bridge is Falling Down di Model Kedua	D-31

DAFTAR GAMBAR

2.1	Perceptron [7]	2-2
2.2	Multi Layer Perceptron [13]	2-3
2.3	Deep Neural Network	2-4
2.4	Contoh gambar	2-4
2.5	Deep Learning Schema	2-5
2.6	Deep Learning Schema	2-5
2.7	Convolution Layer[12]	2-7
2.8	Convolution Layer[14]	2-7
2.9	<i>Depth in convolutional layer</i> [14]	2-9
2.10	<i>convolution process</i> [34]	2-10
2.11	<i>convolution process</i> lanjutan [34]	2-11
2.12	<i>Enlargement convolution process</i> [34]	2-11
2.13	<i>Hierarki CNN</i> [12]	2-13
2.14	<i>Pooling Layer</i> [4]	2-13
2.15	<i>Fully Connected Layer</i> [14]	2-15
2.16	<i>Backward Propagation</i>	2-15
2.17	Contoh Gradient Descent	2-18
2.18	<i>Cross Entropy</i> [10]	2-22
2.19	<i>Convolutional Neural Network</i> [14]	2-23
2.20	VGG16 [18]	2-24
2.21	VGG16 [18]	2-25
2.22	VGG16 [17]	2-25
2.23	<i>Inception problem</i> [18]	2-26
2.24	<i>Convolution module</i>	2-27
2.25	<i>Inception module</i>	2-27
2.26	<i>Inception module architecture</i> [18]	2-28
2.27	<i>Normal Convolution</i> [18]	2-28
2.28	<i>1x1 convolution layer</i> [18]	2-29
2.29	ReLU [26]	2-30
2.30	<i>derivative of ReLU</i> [26]	2-30
2.31	<i>Dropout layer</i>	2-36
2.32	<i>Gradient Descent</i>	2-37
2.33	<i>Gradient Descent</i>	2-37
2.34	<i>Gradient Descent</i>	2-38

2.35	<i>Stochastic Gradient Descent [8]</i>	2-39
2.36	<i>Mini Batch Gradient Descent [8]</i>	2-39
2.37	<i>Local Minimum[22]</i>	2-40
2.38	<i>NAG movement[22]</i>	2-41
2.39	<i>Aspect Aware Preprocess [21]</i>	2-49
2.40	<i>rotation process [39]</i>	2-52
2.41	<i>rotation process [36]</i>	2-52
2.42	<i>shift process [36]</i>	2-53
2.43	<i>flip process [36]</i>	2-53
2.44	<i>brightness process [36]</i>	2-53
2.45	<i>zoom process [36]</i>	2-54
2.46	<i>Original Image</i>	2-54
2.47	<i>Row wised zoom</i>	2-54
2.48	<i>Column wised zoom</i>	2-54
2.49	<i>Original Image</i>	2-55
2.50	<i>Row wised zoom</i>	2-55
2.51	<i>Column wised zoom</i>	2-55
2.52	<i>Column wised zoom</i>	2-55
2.53	<i>K-times zoom</i>	2-55
2.54	<i>K-times zoom</i>	2-56
2.55	<i>Column wised zoom</i>	2-56
2.56	<i>Shear x example</i>	2-57
2.57	<i>Before Histogram Equalization [37]</i>	2-59
2.58	<i>After Histogram Equalization [37]</i>	2-59
2.59	<i>Contrast Limited Adaptive Histogram Equalization (CLAHE) [1]</i>	2-61
2.60	<i>Contrast Limited Adaptive Histogram Equalization (CLAHE) [1]</i>	2-61
2.61	<i>Cropped Yale Face Database Illumination</i>	2-82
2.62	<i>Komnet Dataset example</i>	2-83
3.1	<i>Kerangka Pemikiran</i>	3-2
3.2	<i>Flowchart urutan proses global</i>	3-5
3.3	<i>Preprocessing</i>	3-6
3.4	<i>Flowchart pelatihan sistem pengenalan wajah</i>	3-8
3.5	<i>LAB channel example</i>	3-14
3.6	<i>VGG16 architecture</i>	3-15
3.7	<i>Inception architecture</i>	3-24
4.1	<i>Tampilan GUI awal aplikasi</i>	4-5
4.2	<i>Tampilan GUI untuk pemilihan judul lagu</i>	4-6

4.3	Tampilan GUI ketika berkas yang <i>diupload</i> memiliki ekstensi yang salah	4-7
4.4	Tampilan GUI yang berisi data masukan dari pengguna	4-8
4.5	Tampilan GUI hasil pengujian: informasi nilai akurasi dan F1 <i>Score</i> rata-rata model	4-9
4.6	Tampilan GUI hasil pengujian: hasil pengujian <i>pitch</i> dan <i>distance score</i>	4-10
4.7	Tampilan GUI hasil pengujian: penjelasan singkat dari hasil pengujian yang dilakukan	4-11
4.8	Rancangan arsitektur CNN yang pertama	4-13
4.9	Rancangan arsitektur CNN yang kedua	4-14
4.10	Rancangan arsitektur CNN yang ketiga	4-16
4.11	Rancangan arsitektur CNN yang keempat	4-17
4.12	Grafik perubahan akurasi pada pengujian arsitektur 1 untuk <i>epoch</i> = 50	4-19
4.13	Grafik perubahan akurasi validasi pengujian pada arsitektur 1 untuk <i>epoch</i> = 50	4-20
4.14	Grafik perubahan akurasi pada pengujian arsitektur 1 untuk <i>epoch</i> = 100	4-21
4.15	Grafik perubahan akurasi validasi pada pengujian arsitektur 1 untuk <i>epoch</i> = 100	4-21
4.16	Grafik perubahan akurasi pada pengujian arsitektur 1 untuk <i>epoch</i> = 150	4-22
4.17	Grafik perubahan akurasi validasi pada pengujian arsitektur 1 untuk <i>epoch</i> = 150	4-23
4.18	Grafik perubahan akurasi pada pengujian arsitektur 1 untuk <i>epoch</i> = 200	4-24
4.19	Grafik perubahan akurasi validasi pada pengujian arsitektur 1 untuk <i>epoch</i> = 200	4-24
4.20	Grafik perubahan akurasi pada pengujian arsitektur 1 untuk <i>epoch</i> = 250	4-25
4.21	Grafik perubahan akurasi validasi pada pengujian arsitektur 1 untuk <i>epoch</i> = 250	4-26
4.22	Grafik perubahan akurasi pada pengujian arsitektur 1 untuk <i>epoch</i> = 300	4-27
4.23	Grafik perubahan akurasi validasi pada pengujian arsitektur 1 untuk <i>epoch</i> = 300	4-27

4.24	Grafik akurasi, akurasi validasi, dan selisihnya pada pengujian di arsitektur pertama dengan $learning rate = 0.001 (10^{-3})$	4-31
4.25	Grafik akurasi, akurasi validasi, dan selisihnya pada pengujian di arsitektur pertama dengan $learning rate = 0.0001 (10^{-4})$	4-32
4.26	Grafik akurasi, akurasi validasi, dan selisihnya pada pengujian di arsitektur pertama dengan $learning rate = 0.0002 (2 \times 10^{-4})$	4-33
4.27	Grafik akurasi, akurasi validasi, dan selisihnya pada pengujian di arsitektur pertama dengan $learning rate = 0.00001 (10^{-5})$	4-34
4.28	Grafik perubahan akurasi pada pengujian arsitektur 2 untuk $epoch = 50$	4-36
4.29	Grafik perubahan akurasi validasi pada pengujian arsitektur 2 untuk $epoch = 50$	4-36
4.30	Grafik perubahan akurasi pada pengujian arsitektur 2 untuk $epoch = 100$	4-37
4.31	Grafik perubahan akurasi validasi pada pengujian arsitektur 2 untuk $epoch = 100$	4-38
4.32	Grafik perubahan akurasi pada pengujian arsitektur 2 untuk $epoch = 150$	4-39
4.33	Grafik perubahan akurasi validasi pada pengujian arsitektur 2 untuk $epoch = 150$	4-39
4.34	Grafik perubahan akurasi pada pengujian arsitektur 2 untuk $epoch = 200$	4-40
4.35	Grafik perubahan akurasi validasi pada pengujian arsitektur 2 untuk $epoch = 200$	4-41
4.36	Grafik perubahan akurasi pada pengujian arsitektur 2 untuk $epoch = 250$	4-42
4.37	Grafik perubahan akurasi validasi pada pengujian arsitektur 2 untuk $epoch = 250$	4-42
4.38	Grafik perubahan akurasi pada pengujian arsitektur 2 untuk $epoch = 300$	4-43
4.39	Grafik perubahan akurasi validasi pada pengujian arsitektur 2 untuk $epoch = 300$	4-44
4.40	Grafik akurasi, akurasi validasi, dan selisihnya pada pengujian di arsitektur kedua dengan $learning rate = 0.001 (10^{-3})$	4-47
4.41	Grafik akurasi, akurasi validasi, dan selisihnya pada pengujian di arsitektur kedua dengan $learning rate = 0.0001 (10^{-4})$	4-48

4.42 Grafik akurasi, akurasi validasi, dan selisihnya pada pengujian di arsitektur kedua dengan $learning rate = 0.0002 (2 \times 10^{-4})$	4-49
4.43 Grafik akurasi, akurasi validasi, dan selisihnya pada pengujian di arsitektur kedua dengan $learning rate = 0.00001 (10^{-5})$	4-50
4.44 Grafik perubahan akurasi pada pengujian arsitektur 3 untuk $epoch = 50$	4-52
4.45 Grafik perubahan akurasi validasi pada pengujian arsitektur 3 untuk $epoch = 50$	4-52
4.46 Grafik perubahan akurasi pada pengujian arsitektur 3 untuk $epoch = 100$	4-53
4.47 Grafik perubahan akurasi validasi pada pengujian arsitektur 3 untuk $epoch = 100$	4-54
4.48 Grafik perubahan akurasi pada pengujian arsitektur 3 untuk $epoch = 150$	4-55
4.49 Grafik perubahan akurasi validasi pada pengujian arsitektur 3 untuk $epoch = 150$	4-55
4.50 Grafik perubahan akurasi pada pengujian arsitektur 3 untuk $epoch = 200$	4-56
4.51 Grafik perubahan akurasi validasi pada pengujian arsitektur 3 untuk $epoch = 200$	4-57
4.52 Grafik perubahan akurasi pada pengujian arsitektur 3 untuk $epoch = 250$	4-58
4.53 Grafik perubahan akurasi validasi pada pengujian arsitektur 3 untuk $epoch = 250$	4-58
4.54 Grafik perubahan akurasi pada pengujian arsitektur 3 untuk $epoch = 300$	4-59
4.55 Grafik perubahan akurasi validasi pada pengujian arsitektur 3 untuk $epoch = 300$	4-60
4.56 Grafik akurasi, akurasi validasi, dan selisihnya pada pengujian di arsitektur ketiga dengan $learning rate = 0.001 (10^{-3})$	4-63
4.57 Grafik akurasi, akurasi validasi, dan selisihnya pada pengujian di arsitektur ketiga dengan $learning rate = 0.0001 (10^{-4})$	4-64
4.58 Grafik akurasi, akurasi validasi, dan selisihnya pada pengujian di arsitektur ketiga dengan $learning rate = 0.0002 (2 \times 10^{-4})$	4-65
4.59 Grafik akurasi, akurasi validasi, dan selisihnya pada pengujian di arsitektur ketiga dengan $learning rate = 0.00001 (10^{-5})$	4-66

4.60	Grafik perubahan akurasi pada pengujian arsitektur 4 untuk <i>epoch</i> $= 50$	4-68
4.61	Grafik perubahan akurasi validasi pada pengujian arsitektur 4 untuk <i>epoch</i> = 50	4-68
4.62	Grafik perubahan akurasi pada pengujian arsitektur 4 untuk <i>epoch</i> $= 100$	4-69
4.63	Grafik perubahan akurasi validasi pada pengujian arsitektur 4 untuk <i>epoch</i> = 100	4-70
4.64	Grafik perubahan akurasi pada pengujian arsitektur 4 untuk <i>epoch</i> $= 150$	4-71
4.65	Grafik perubahan akurasi validasi pada pengujian arsitektur 4 untuk <i>epoch</i> = 150	4-71
4.66	Grafik perubahan akurasi pada pengujian arsitektur 4 untuk <i>epoch</i> $= 200$	4-72
4.67	Grafik perubahan akurasi validasi pada pengujian arsitektur 4 untuk <i>epoch</i> = 200	4-73
4.68	Grafik perubahan akurasi pada pengujian arsitektur 4 untuk <i>epoch</i> $= 250$	4-74
4.69	Grafik perubahan akurasi validasi pada pengujian arsitektur 4 untuk <i>epoch</i> = 250	4-74
4.70	Grafik perubahan akurasi pada pengujian arsitektur 4 untuk <i>epoch</i> $= 300$	4-75
4.71	Grafik perubahan akurasi validasi pada pengujian arsitektur 4 untuk <i>epoch</i> = 300	4-76
4.72	Grafik akurasi, akurasi validasi, dan selisihnya pada pengujian di arsitektur keempat dengan <i>learning rate</i> = 0.001 (10^{-3})	4-79
4.73	Grafik akurasi, akurasi validasi, dan selisihnya pada pengujian di arsitektur keempat dengan <i>learning rate</i> = 0.0001 (10^{-4})	4-80
4.74	Grafik akurasi, akurasi validasi, dan selisihnya pada pengujian di arsitektur keempat dengan <i>learning rate</i> = 0.0002 (2×10^{-4})	4-81
4.75	Grafik akurasi, akurasi validasi, dan selisihnya pada pengujian di arsitektur keempat dengan <i>learning rate</i> = 0.00001 (10^{-5})	4-82
4.76	Contoh spektrogram dengan sebaran energi tinggi	4-99
4.77	Contoh spektrogram dengan sebaran energi rendah	4-100

DAFTAR ALGORITME

Algoritme 2.1	algoritma <i>Aspect Aware Preprocessing</i>	2-49
Algoritme 2.2	<i>K-times zooming algorithm</i>	2-55
Algoritme 2.3	<i>Conversion between RGB to LAB channel</i>	2-58
Algoritme 2.4	<i>HE Algorithm [37]</i>	2-59
Algoritme 3.1	<i>CLAHE algorithm</i>	3-7
Algoritme 3.2	<i>Forward Propagation Convolutional Neural Network</i>	3-10
Algoritme 3.3	<i>Backward Propagation Convolutional Neural Network without optimizer</i>	3-11
Algoritme 3.4	<i>Backward Propagation Convolutional Neural Network with optimizer</i>	3-11
Algoritme 3.5	algoritma <i>Histogram Equalization</i>	3-13
Algoritme 3.6	Algoritma <i>MCLAHE</i>	3-13

DAFTAR LAMPIRAN

LAMPIRAN A	A-1
LAMPIRAN B	B-1
LAMPIRAN C	C-1
LAMPIRAN D	D-1

BAB 1 PENDAHULUAN

Bab ini menjelaskan landasan yang menjadi acuan bagi penulis dalam melakukan penelitian ini. Landasan tersebut didasarkan kepada fenomena masalah yang terjadi dan akan dibahas dalam penelitian setelah membaca sumber-sumber referensi.

1.1 Latar Belakang

Pada zaman modern sekarang ini, banyak sistem aplikasi yang menggunakan sistem pengenalan / identifikasi berbasis biometrik karena kemampuan pengenalan biometrik yang memiliki tingkat akurasi yang tinggi, dan juga penggunaannya yang nyaman bagi *user* [1]. Teknologi pengenalan wajah adalah satu dari sekian banyak teknologi pengenalan *biometric* yang paling banyak digunakan untuk mengklasifikasi serta membedakan identitas satu orang dengan yang lainnya. Teknologi pengenalan wajah digunakan untuk memverifikasi dan juga mengidentifikasi wajah. Sistem pengenalan wajah bertujuan untuk mengenali, mengklasifikasi orang tertentu pada gambar atau video dengan membandingkannya dengan *database*.

Aplikasi pengenalan wajah yang digunakan sekarang secara garis besar dibagi menjadi dua kelompok besar [2] berdasarkan tujuan penggunaannya :

1. **Face Identification** : Gambar wajah dapat digunakan untuk mendefinisikan identitas seseorang.
2. **Face Verification** : Sistem diberikan wajah dan estimasi identitas, tugas sistem adalah untuk menguji apakah estimasi identitas adalah tepat dibandingkan dengan wajah yang diberikan. Tujuan dari penggunaan aplikasi verifikasi wajah adalah untuk keamanan.

Meskipun sudah banyak aplikasi-aplikasi dan juga sistem yang mengaplikasikan teknologi pengenalan wajah, topik ini masih menjadi perdebatan di antara para peneliti mengenai metode yang tepat untuk meningkatkan akurasi pengenalan wajah sekaligus memiliki kompleksitas yang rendah (tingkat komputasi yang rendah). Hingga saat ini, banyak metode-metode yang bersumber dari metode tradisional seperti *Eigenface* dan *Fisherman* dengan mengaplikasikan algoritma seperti *Principal Component Analysis (PCA)*, *Linear Discriminant Analysis (LDA)* dan *Independent Component Analysis (ICA)* [2].

Metode klasifikasi yang banyak digunakan untuk teknologi pengenalan wajah adalah pendekatan *SVM* (*Support Vector Machine*) dan *LBP* (*Local Binary Pattern*). Tetapi pendekatan tradisional tersebut memiliki *bottleneck*, salah satunya adalah tidak optimal pada *dataset* berjumlah besar [2].

Untuk mengatasi *bottleneck* tersebut, maka muncul pendekatan baru yaitu dengan menggunakan *neural network / deep learning*, paling umum digunakan adalah *Convolutional Neural Network (CNN)*. *CNN* banyak digunakan untuk kepentingan pengurangan dimensi dan pengenalan identitas wajah pada gambar maupun video. *Deep Learning* mengadaptasi arsitektur jaringan syaraf, merupakan *neural network* dengan *hidden layer* lebih dari satu. Dengan menggunakan arsitektur model yang mengandung beberapa transformasi *non-linear*, *deep learning* mampu menemukan *feature* tingkat tinggi pada data gambar.

Pendekatan *Deep Learning* dengan menggunakan *CNN* memberikan hasil yang lebih optimal dibandingkan dengan metode *machine learning* konvensional. Akan tetapi metode konvensional memiliki kompleksitas yang lebih rendah dibandingkan dengan metode *deep learning*. Metode *Deep learning* yang tertera pada jurnal dengan judul *Illumination-robust face recognition based on deep convolutional neural networks architecture* memiliki akurasi yang cukup baik yaitu 99.89%, menggunakan arsitektur *Inception* dan gambar terlebih dahulu di *preprocess* dengan metode *Modified Contrast Limited Adaptive Histogram Equalization* [1].

Penelitian yang dilakukan oleh *Samar et al.* [2] dengan judul *Deep Learning Face Detection and Recognition* mengembangkan arsitektur *CNN* 15 *layer* dengan menggunakan *optimizer SGD* sehingga mencapai akurasi 99.67 % dengan *faces96 dataset*. Penelitian yang dilakukan oleh *Pranav* dalam jurnalnya yang berjudul *Design and Evaluation of a Real-Time Face Recognition System using Convolutional Neural Network* [3] meneliti pengaruh *hyperparameter tuning* pada arsitektur *CNN* sehingga menghasilkan akurasi sebesar 98.75 %.

Penelitian berikutnya yang dilakukan oleh *Musab et al.* dalam jurnalnya yang berjudul *Face Recognition based on Convolutional Neural Network* [4] meneliti tentang pengaruh penambahan *batch normalization* terhadap akurasi pengenalan wajah. *Batch normalization* membantu memitigasi dampak dari *internal covariate shift* yang terjadi pada data gambar yang mengalami pemrosesan oleh bobot pada *hidden layer*. Akurasi tertinggi yang dicapai dalam penelitian ini adalah 94.8 %. Penelitian berikutnya yang dilakukan oleh *Zhiming Xie* dalam jurnalnya yang

berjudul *A Face Recognition Method based on CNN* [5] membahas tentang pengaruh perubahan jumlah *neuron* pada *convolution layer* terhadap kenaikan akurasi. Penelitian oleh Xie menggunakan arsitektur *LeNet-5*, diperoleh akurasi tertinggi dengan penggunaan jumlah neuron pada *convolution layer* masing-masing berjumlah 36,76 dan 1024.

Berdasarkan jurnal-jurnal referensi tersebut, penelitian ini menggunakan *M-CLAHE* sebagai *preprocessing data*, lalu untuk ekstraksi *feature* akan digunakan *Inception* dan *VGG16*. Sebagai pembanding penelitian juga akan menggunakan data yang diproses dengan *M-CLAHE* dan yang tidak diproses.

Deep learning bekerja dengan dua proses yaitu *feed-forward propagation* dan *backward propagation*. *Feed-forward propagation* mempelajari fitur-fitur pada gambar dan menghasilkan model yang akan dipakai untuk memprediksi gambar. Tentunya dalam satu kali iterasinya model yang dihasilkan menghasilkan nilai *error*. Tujuan dari pelatihan ini adalah untuk meminimalkan nilai *error* dari model dengan mengubah nilai bobot pada *hidden layer*. Nilai bobot pada *hidden layer* diubah pada saat *deep learning* melakukan proses *backward propagation*. Metode yang dipakai untuk merubah nilai bobot pada *hidden layer* ditentukan oleh jenis *optimizer* yang digunakan. Penelitian kali ini bermaksud menguji jenis *optimizer* yang menghasilkan tingkat akurasi yang paling tinggi.

1.2 Rumusan Masalah

Berdasarkan latar belakang di atas, penelitian ini akan membahas beberapa masalah sebagai berikut:

1. Apakah preproses *M-CLAHE* berpengaruh pada tingkat akurasi model ?
2. Apakah penggunaan *optimizer* berpengaruh terhadap kenaikan akurasi pengenalan wajah ?
3. arsitektur *CNN* yang mana yang menghasilkan tingkat akurasi paling tinggi dalam pengenalan wajah ?

1.3 Tujuan Penelitian

Berdasarkan rumusan masalah di atas, tujuan penelitian ini adalah sebagai berikut:

1. Menguji akurasi model yang dihasilkan dengan menggunakan *dataset* yang dipreproses dengan *M-CLAHE* dengan yang tidak.
2. Membandingkan akurasi arsitektur *CNN* yang dapat digunakan dalam pengenalan wajah.

3. Menguji penggunaan *optimizer* dalam arsitektur *CNN* untuk pengenalan wajah.

1.4 Batasan Masalah

Dalam penelitian ini, ada beberapa batasan masalah, antara lain:

1. Penelitian mengukur akurasi deteksi wajah dengan membandingkan arsitektur dari model *CNN* yang dibangun berdasarkan pola *VGG-16* dan *Inception / GoogleNet*.
2. *Hyperparameter* yang diuji adalah jenis *optimizer*.
3. penelitian ini juga menguji pengaruh *preprocessing* menggunakan metode *MCLAHE* terhadap peningkatan akurasi model.
4. Data gambar wajah yang dipakai pada penelitian ini hanyalah wajah yang menghadap ke depan (*frontal face*).

1.5 Kontribusi Penelitian

Kontribusi yang diberikan dari penelitian ini adalah:

1. Menguji arsitektur *Neural Network* yang memiliki tingkat akurasi yang lebih tinggi terhadap klasifikasi pengenalan wajah.
2. Melakukan pengujian mengenai pengaruh penggunaan jenis *optimizer* yang berbeda terhadap peningkatan akurasi pengenalan wajah.

1.6 Metode Penelitian

Metode penelitian yang dilakukan dalam penelitian ini adalah sebagai berikut:

1. Studi Literatur

Penelitian ini dimulai dengan studi kepustakaan yaitu mengumpulkan referensi baik dari buku, artikel, *paper*, jurnal, makalah mengenai arsitektur *neural network, face recognition* dan juga *optimizers*.

2. *Data Sampling*

Pengambilan *sample* data di internet. *Dataset* yang dipakai pada penelitian ini adalah *Cropped Yale Face Database* dan *Komnet Dataset*.

3. Analisis Masalah

Pada tahap ini, dilakukan analisis permasalahan yang ada, batasan-batasan yang dimiliki, dan kebutuhan yang diperlukan untuk menyelesaikan permasalahan yang sudah dianalisis.

4. Perancangan dan Implementasi Algoritme

Pada tahap ini, dilakukan perancangan algoritme *convolutional neural network* beserta dengan arsitektur yang akan diteliti.

BAB 1 PENDAHULUAN

5. Pengujian

Pada tahap ini, dilakukan pengujian terhadap aplikasi *face recognition* yang telah dibangun.

6. Evaluasi Metode (Kesimpulan)

Pada tahap ini, dilakukan pengumpulan data penelitian dan mengambil suatu konklusi dari penelitian yang sudah dilakukan.

7. Dokumentasi

Pada tahap ini, dilakukan pendokumentasian hasil analisis dan implementasi secara tertulis dalam bentuk laporan skripsi.

1.7 Sistematika Pembahasan

Penelitian ini disusun berdasarkan sistematika penulisan sebagai berikut:

BAB 1 PENDAHULUAN

Bab ini berisi latar belakang, rumusan masalah, tujuan penelitian, batasan masalah, kontribusi penelitian, serta metode penelitian yang akan digunakan untuk membangun model *face recognition*.

BAB 2 LANDASAN TEORI

Bab ini berisi penjelasan dasar tentang teori yang mendukung penelitian ini, seperti *artificial neural network*, *convolutional neural network*, *histogram equalization* dan *optimizers*.

BAB 3 ANALISIS DAN PERANCANGAN SISTEM

Bab ini berisi analisis algoritme *convolutional neural network* beserta dengan jenis arsitektur yang dipakai pada penelitian kali ini.

BAB 4 IMPLEMENTASI DAN PENGUJIAN

Bab ini berisi implementasi dan pengujian arsitektur *convolutional neural network* yang digunakan untuk melakukan *face recognition* dengan melakukan *modeling convolutional neural network* dan mengujinya dengan data *testing*. Selain itu, bab ini juga menjelaskan hasil pengujian arsitektur *convolutional neural network*.

BAB 5 KESIMPULAN DAN SARAN

Bab ini berisi kesimpulan dari penelitian berdasarkan hasil pengujian yang telah dilakukan dan saran untuk penelitian lebih lanjut di masa

BAB 1 PENDAHULUAN

mendatang pada bidang *face recognition*.

BAB 2 LANDASAN TEORI

Bab ini menjelaskan dasar-dasar teori yang digunakan dalam penelitian ini. Teori tersebut merupakan teori penunjang yang diambil dari jurnal terkait dan buku-buku referensi.

2.1 Tinjauan Pustaka

Penelitian ini menggunakan teori-teori terkait yang diperlukan dalam penelitian. Pembahasan mengenai teori-teori tersebut akan dijelaskan sebagai berikut.

2.1.1 Neural Network

Neural Network memiliki prinsip kerja yang sama dengan otak manusia, memungkinkan komputer untuk memahami pola dan memecahkan masalah masalah di dunia *Artificial Intelligent*. *machine learning* maupun *deep learning* [13].

Neural Network dikenal juga dengan sebutan *Artificial Neural Network (ANN)* atau *Simulated Neural Network(SNN)* adalah *subset* dari *machine learning* dan pusat dari algoritma *deep learning*. Nama dan struktur kerja *Neural Network* mirip dengan cara kerja otak manusia, meniru bagaimana cara kerja otak manusia mentransfer sinyal dari satu *neuron (node)* ke *neuron (node)* lainnya [13].

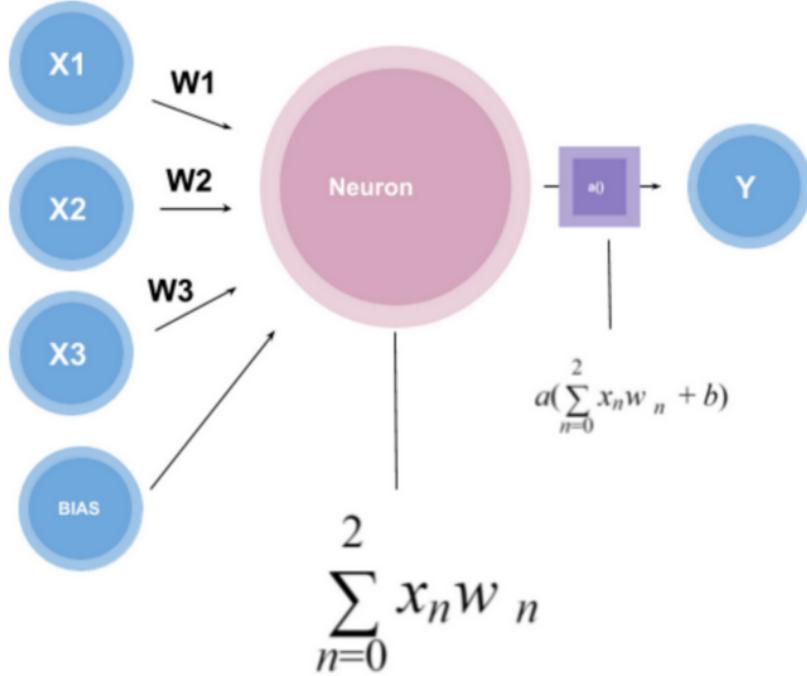
Artificial Neural Network (ANN) terdiri dari lapisan-lapisan *node*. Lapisan-lapisan *node* tersebut terbagi lagi menjadi tiga kategori : *input*, *hidden layer(s)*, dan *output*. Setiap *node (artificial neuron)* terkoneksi ke setiap *node* lainnya dan memiliki *weight* dan *threshold* yang terasosiasi secara khusus untuk *node* tersebut. Jika nilai keluaran suatu *node* melebihi nilai *thresholdnya*, maka *node* tersebut akan mengirimkan data ke jaringan *neuron* berikutnya, jika tidak melebihi nilai *threshold*, maka *node* tidak akan megirimkan data [13].

2.1.1.1 Perceptron

Perceptron adalah *unit* dasar dari *neural network*. *Perceptron* terdiri dari nilai *input*, *weight*, *bias*, *weighted sum* dan fungsi aktivasi. Seperti yang terlihat pada gambar 2.1.

Perceptron bekerja dengan cara mengambil nilai-nilai masukan lalu mengalikannya dengan masing-masing *weight (weighted sum)* lalu menjumlahkannya dengan *bias*. Lalu hasil akhir akan di lewatkan kepada fungsi aktivasi pada *node* tersebut dan

akan menghasilkan nilai keluaran.[7]



Gambar 2.1 Perceptron [7]

Cara kerja Neural Network

Setiap *node* memiliki model *regresinya* sendiri, terdiri dari *input*, *weight*, *bias* dan *output*.

$$\sum_{i=1}^m w_i x_i + bias = w_1 x_1 + w_2 x_2 + \dots + w_i x_i + bias \quad (2.1)$$

Keterangan :

- m : total *neuron*
- i : urutan *neuron*
- w : bobot *neuron*
- x : nilai *feature*
- $bias$: nilai bias

Nilai bobot di-*assign* berdasarkan tingkat korelasi *variable* terhadap nilai keluaran, semakin tinggi nilai bobot mengartikan bahwa *variable* tersebut semakin

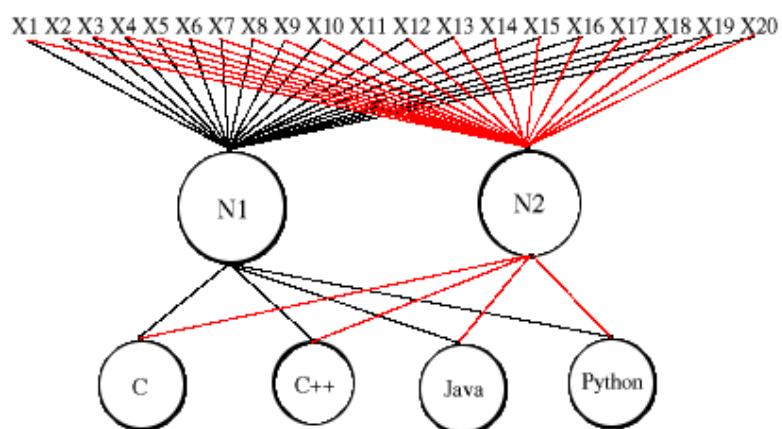
berpengaruh terhadap suatu nilai keluaran dibandingkan dengan *input* yang lainnya [7].

Setiap *input* nantinya dikalikan dengan nilai bobotnya masing-masing lalu dijumlahkan. Hasilnya akan menjadi *parameter* untuk *activation function* di *node* tersebut yang akan menentukan nilai keluarannya [7].

$$output = f(x) = 1[\sum w_i x_i + bias \geq 0] \quad (2.2)$$

$$output = f(x) = 0[\sum w_i x_i + bias < 0] \quad (2.3)$$

Jika hasil dari penjumlahan *input* dan nilai bobotnya melebihi nilai *threshold*-nya, maka *node* akan teraktifasi dan akan melewatkkan data ke lapisan berikutnya di dalam jaringan. Nilai keluaran dari suatu jaringan *node* akan menjadi nilai masukan di jaringan *node* lainnya. Proses melewati data dari satu jaringan ke jaringan lainnya pada *neural network* disebut ***feedforward network***. Karena terdiri lebih dari satu *node*, maka *feedforward network* disebut juga ***Multi Layer Perceptron (MLP)***, dan pada umumnya fungsi aktivasi dari setiap *node* adalah fungsi *non-linear* [7].

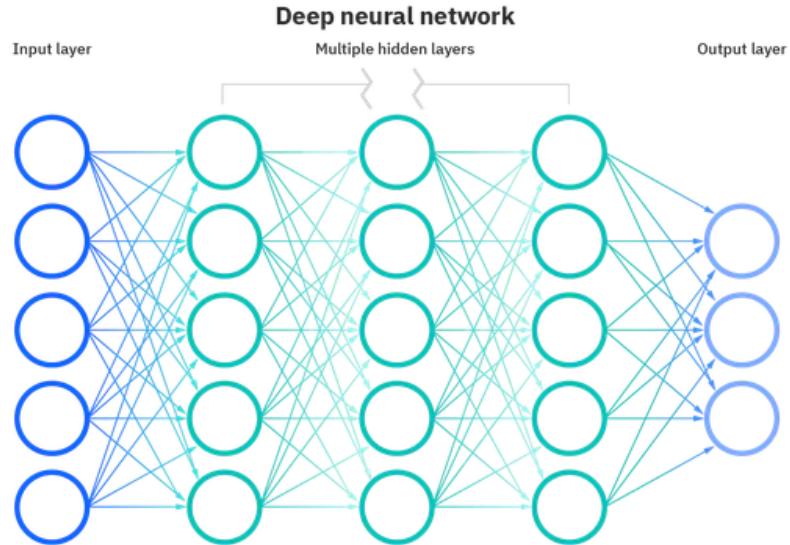


Gambar 2.2 Multi Layer Perceptron [13]

2.1.1.2 Deep Learning

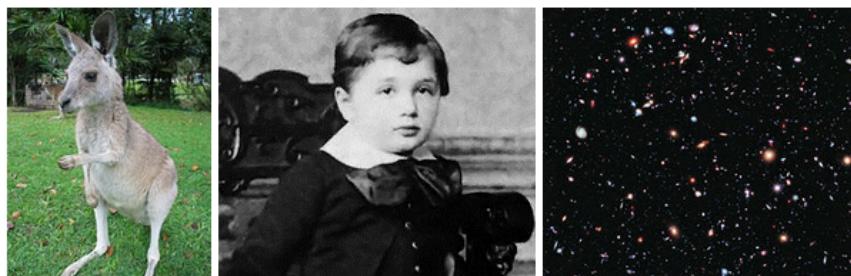
Perbedaan antara *neural network* dan *deep neural network* terletak pada jumlah *hidden layer*-nya. Pada *neural network* hanya terdapat satu *hidden layer*, sementara

pada *deep neural network* terdapat lebih dari satu *hidden layer* [12].

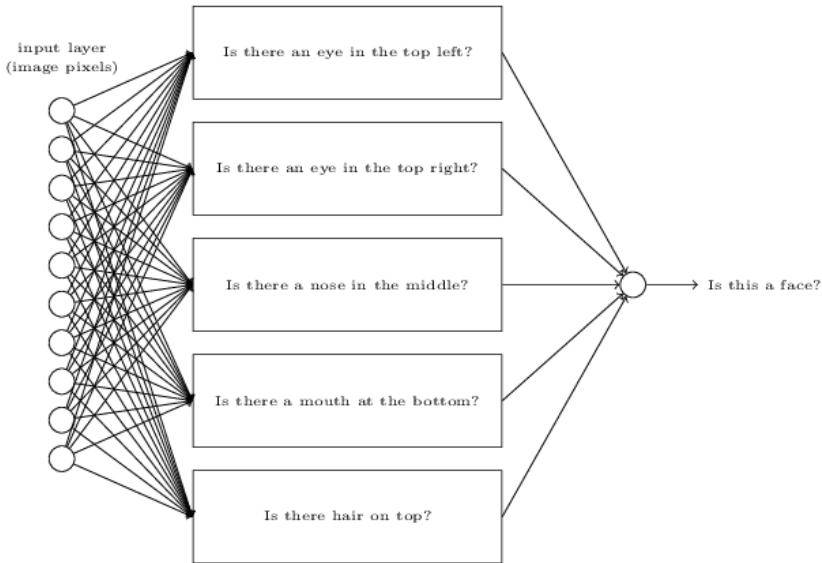


Gambar 2.3 Deep Neural Network

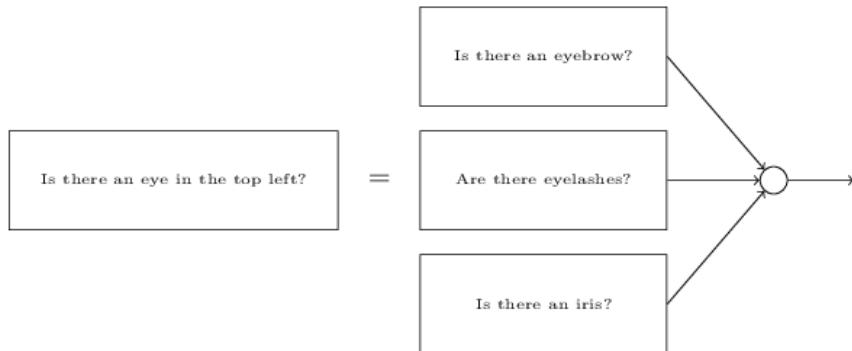
Asumsikan hendak membangun model untuk memprediksi apakah pada gambar 2.4 terdapat wajah atau tidak, *Neural network* akan mengekstraksi fitur-fitur pada gambar yang pada akhirnya akan dipelajari oleh *network* untuk mengklasifikasi apakah pada gambar terdapat wajah atau tidak [12].



Gambar 2.4 Contoh gambar



Gambar 2.5 Deep Learning Schema



Gambar 2.6 Deep Learning Schema

Secara heuristik jika sebagian besar dari fitur hasil *threshold*-nya adalah positif, maka kemungkinan besar *network* akan mengklasifikasi suatu gambar terdapat wajah (perhatikan gambar 2.5). Masing-masing fitur dapat didekomposisi lebih jauh untuk mendapatkan hasil yang lebih akurat (perhatikan gambar 2.6). Fitur ini bisa di *break down* lebih jauh lagi, membentuk *multiple layer* pada *hidden layer neural network*-nya, dengan *layer* yang paling awal untuk menyaring fitur yang sederhana namun *pixel*-nya berjumlah besar dan *layer* yang terakhir untuk menyaring fitur yang lebih detail yang bisa saja dapat dijawab oleh *pixel* yang jumlahnya jauh lebih sedikit [12].

2.1.2 Convolutional Neural Network

Convolutional Neural Network (CNN) atau yang biasa disebut juga *ConvNets*. Diperkenalkan pada tahun 1980 oleh Yann Lecun. Versi awal dari *CNN* dikenal dengan nama *LeNet*, dapat mengenali angka numerik yang ditulis oleh tangan.

Convolutional Neural Network terdiri dari lapisan berlapis dari *artificial neurons*. *Artificial Neurons* adalah imitasi kasar dari saraf / *neuron* secara biologis, yang pada dasarnya adalah fungsi matematis untuk menghitung jumlah bobot dari sejumlah nilai masukan dan hasil keluarannya dikenakan fungsi aktivasi [11].

Convolutional neural network, atau disingkat sebagai *CNN*, adalah pengembangan dari *multilayer perceptron* yang berupa *neural network* khusus untuk memproses data yang memiliki topologi seperti *grid*. Contoh data yang dapat diproses adalah *time-domain* data yang dapat dianggap sebagai *grid* satu dimensi yang mengambil sampel pada interval waktu dan data gambar yang dapat dianggap sebagai *grid* piksel dua dimensi. Nama *convolutional neural network* menunjukkan bahwa jaringan menggunakan operasi matematika yang disebut konvolusi, yang merupakan jenis operasi linear khusus. Jaringan konvolisional adalah *neural network* yang menggunakan operasi konvolusi sebagai ganti dari penerapan matriks umum pada setidaknya satu dari lapisannya [29].

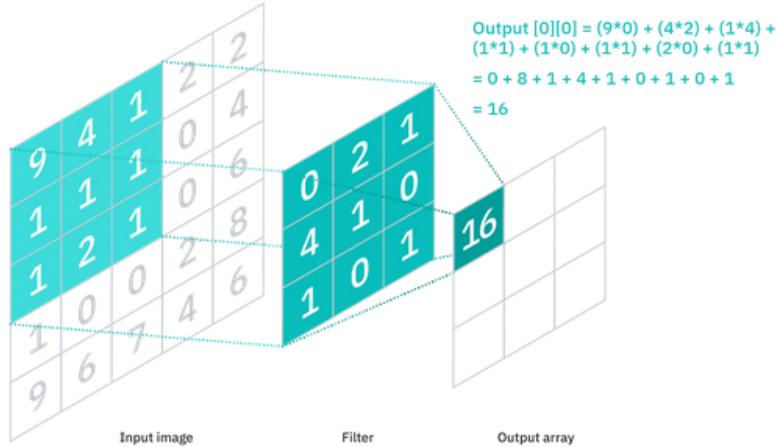
Dalam bentuk yang paling umum, konvolusi adalah operasi pada dua fungsi argumen yang bernilai *real*. Ada dua buah argumen yang dibutuhkan untuk melakukan operasi konvolusi: masukan dan *kernel*. Keluaran hasil konvolusi disebut sebagai *feature map*. Dalam aplikasi pembelajaran mesin, masukan berupa larik multidimensi dan textitkernel biasanya berupa larik multidimensi dari parameter yang diadaptasi oleh algoritme pembelajaran. Larik multidimensi ini disebut juga sebagai *tensor*. Setiap elemen masukan dan *kernel* harus disimpan secara terpisah. *Kernel* dapat belajar dengan memperbaharui nilainya pada tahap pelatihan, yaitu *backward propagation*. Ukuran *kernel* ditentukan oleh peneliti. Pemilihan ukuran *kernel* bergantung kepada fitur yang ingin dipelajari dari masukan [29].

Ada tiga jenis lapisan yang membentuk arsitektur *convolutional neural network*, yaitu: *convolutional layer*, *max pooling layer*, dan *dense/fully connected layer*.

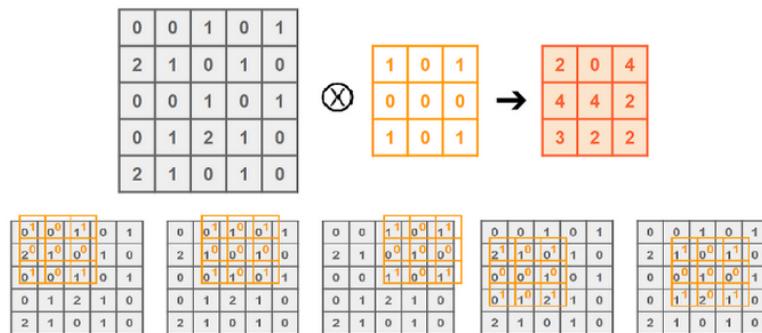
2.1.2.1 Convolution Layer

Convolution layer adalah elemen yang membedakan *CNN* dengan arsitektur *neural network* lainnya. Layar konvolusi secara otomatis mengekstraksi fitur-fitur tanpa campur tangan dari luar. Terdapat beberapa komponen, di antaranya ada masukan data, *filter*, dan *map*. Asumsikan data masukannya berupa gambar berwarna, artinya data masukan berupa matrix tiga dimensi yang memiliki *width*, *height* dan *depth*. Terdapat juga *filter* (*kernel*) yang berjalan untuk mendekripsi apakah terdapat fitur

tertentu. Proses ini yang disebut dengan *convolution* [12].



Gambar 2.7 Convolution Layer[12]



Gambar 2.8 Convolution Layer[14]

Feature detector / kernel pada umumnya menggunakan larik dua dimensi (3×3), walaupun pada prakteknya bisa saja ukurannya dinamis. *Kernel* tersebut kemudian akan diaplikasikan kepada seluruh *input field* (perhatikan gambar 2.9) dan dilakukan operasi *dot product* antara *input pixel* dengan *filter*. Selanjutnya *filter* akan bergerak sesuai dengan nilai *strides* pada fungsi. Nilai keluaran dari proses ini disebut *feature map (convolved feature)* [12]. Penjelasan mengenai formula perhitungan konvolusi dapat dilihat pada persamaan 2.4.

Bobot pada *feature detector / kernel* akan tetap meskipun dia bergerak ke seluruh *input field (parameter sharing)*. Bobot *kernel* akan di-update selama proses *training*. Beberapa bobot akan mengalami perubahan melalui proses *backpropagation* dan *gradient descent*. [12]. Beberapa contoh dari pendekatan yang ada dapat dilihat pada sub bab 2.1.6.

$$v(i, j) = \sum_n \sum_m w_{(n,m)} x_{(i-n, j-m)} \quad (2.4)$$

Keterangan :

- x : citra masukan
 w : nilai matriks *kernel*
 $v(i, j)$: nilai piksel x, y pada citra keluaran
 i : koordinat x pada citra.
 j : koordinat y pada citra.
 n : variabel iterasi untuk lebar citra.
 m : variabel iterasi untuk tinggi citra.
-

Ada beberapa parameter yang harus diinisiasi terlebih dahulu sebelum memulai *training neural network* [12].

- ***Stride***

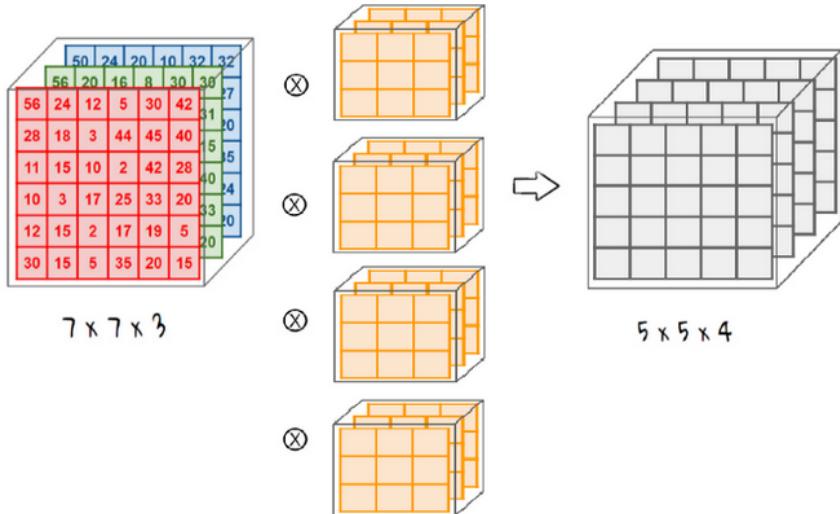
jumlah *pixel* yang dilewati ketika *kernel* bergerak.

- ***Padding***

Digunakan ketika dimensi *filter* tidak sesuai jumlahnya dengan dimensi *input image*. Terdapat *Valid padding* atau biasa dikenal dengan istilah *no padding*, membuang *convolution* terakhir jika dimensi *kernel* tidak sesuai. Terdapat juga *Same padding*, memastikan *output layer* mempunyai ukuran dimensi yang sama dengan *input layer* [12].

- **Jumlah filter**

mempengaruhi kedalaman dari dimensi nilai keluarannya. Tiga *distinct filter* akan menghasilkan tiga *feature map*. Ini artinya akan menghasilkan nilai keluaran dengan dimensi *depth* bernilai tiga.



Gambar 2.9 Depth in convolutional layer[14]

Gambar 2.9 menunjukkan bahwa empat *kernel* menghasilkan keluaran *tensor* dengan empat *channel*.

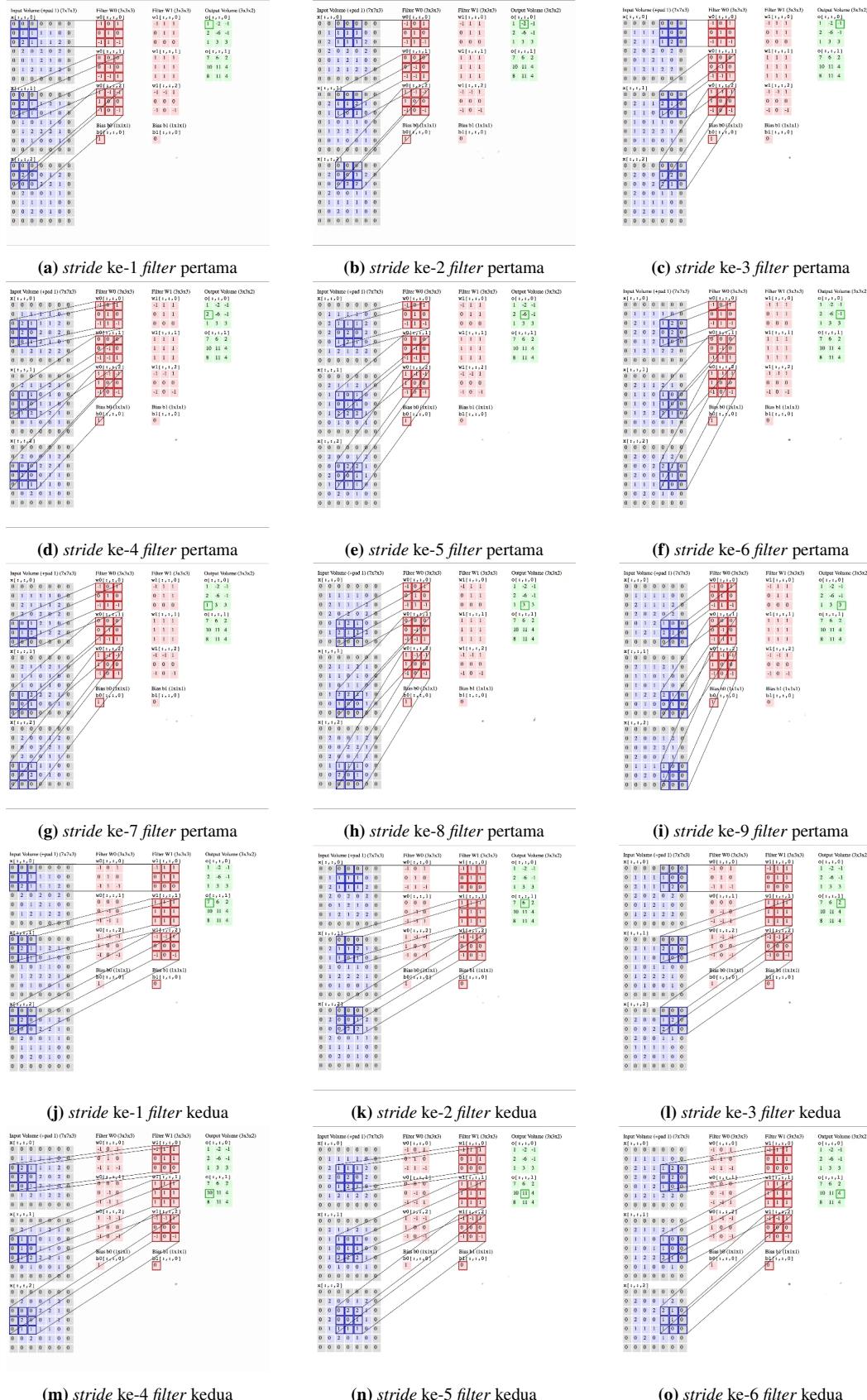
Gambar 2.10 dan 2.11 menggambarkan ilustrasi proses konvolusi. Pada contoh di atas, dilakukan proses konvolusi citra dengan dimensi $7 \times 7 \times 3$ dengan *kernel* berdimensi 3×3 yang berjumlah dua. Operasi konvolusi dijalankan dengan nilai *strides* sama dengan dua dan *valid padding*. Nilai *strides* adalah jumlah *pixel* yang dilewati *kernel* ketika bergerak.

Gambar 2.12 adalah perbesaran dari gambar 2.10 pada bagian 2.10a. Pada gambar tersebut dapat dilihat bahwa satu *kernel* diaplikasikan kepada semua *channel* citra masukan. Operasi ini menggunakan nilai *bias* sama dengan satu. Operasi konvolusi berlangsung pada setiap *channel*, hasil akhirnya akan dijumlahkan lalu ditambah dengan nilai *bias*. Nilai tersebut akan menjadi nilai pertama pada *channel* pertama *tensor* keluarannya. Bila dijabarkan lebih lanjut secara matematis proses untuk mendapatkan nilai pada keluaran *tensor*, dapat dibagi menjadi tiga bagian (karena ada tiga *kernel*). Menggunakan persamaan 2.4, bagian pertama menghasilkan nilai keluaran sebagai berikut

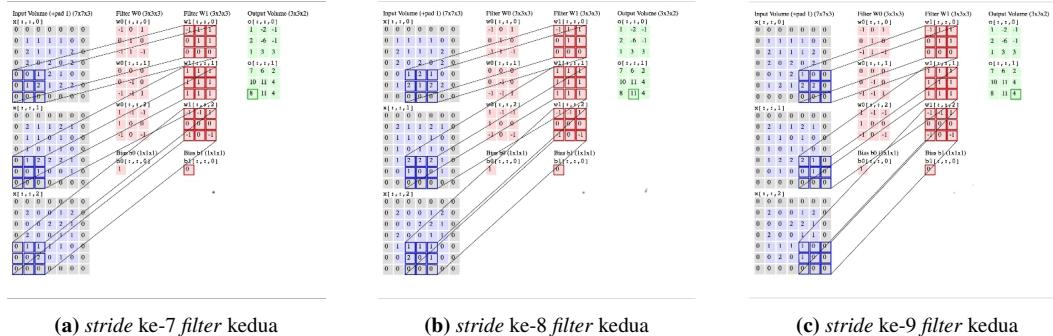
$$(0 \times -1) + (0 \times 0) + (0 \times 1) + (0 \times 0) + (1 \times 1) + (1 \times 0) + (0 \times -1) + (2 \times 1) + (1 \times -1) = 2$$

Bagian kedua menghasilkan nilai keluaran sebagai berikut

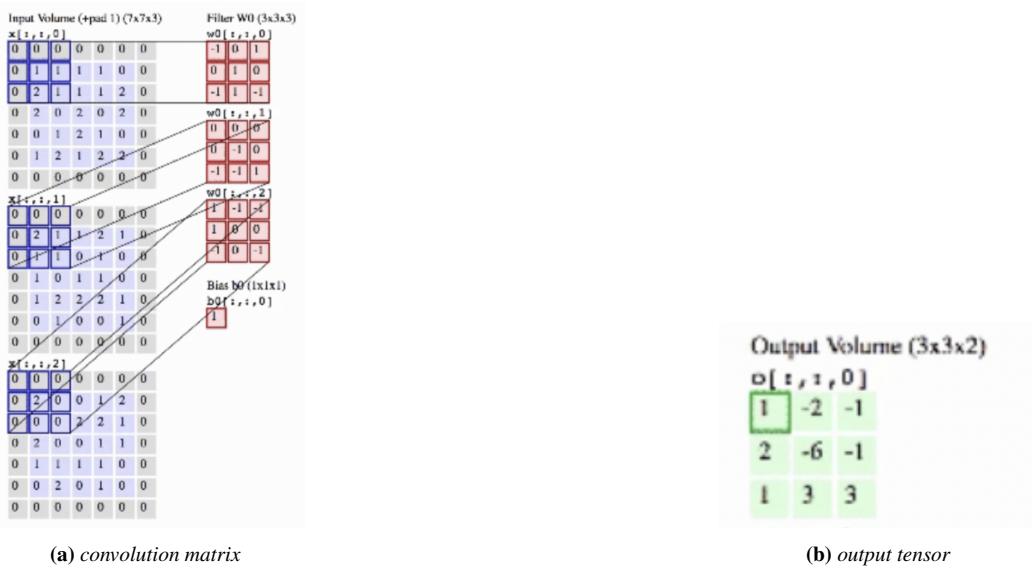
BAB 2 LANDASAN TEORI



Gambar 2.10 convolution process [34]



Gambar 2.11 convolution process lanjutan [34]



Gambar 2.12 Enlargement convolution process [34]

$$(0 \times 0) + (0 \times 0) + (0 \times 0) + (0 \times 0) + (2 \times -1) + (1 \times 0) + (0 \times -1) + (1 \times -1) + (1 \times 1) = -2$$

Bagian ketiga menghasilkan nilai keluaran sebagai berikut

$$(0 \times 1) + (0 \times -1) + (0 \times -1) + (0 \times 1) + (2 \times 0) + (0 \times 0) + (-1 \times 0) + (0 \times 0) + (0 \times -1) = 0$$

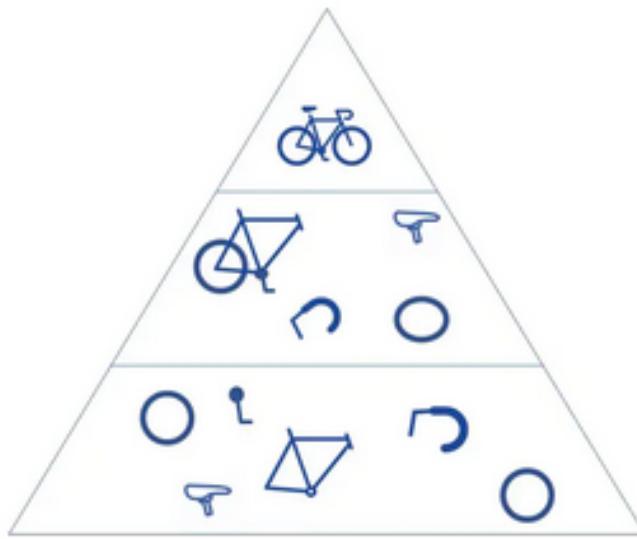
Nilai-nilai yang diperoleh lalu dijumlahkan dan ditambah dengan nilai *bias* pada sistem. Apabila melihat gambar 2.12, dapat dilihat bahwa nilai *bias* bernilai 1, maka nilai total pada *channel* pertama *output tensor* adalah

$$2 - 2 + 0 + 1 = 1$$

Proses ini berlangsung hingga seluruh *image field* terkonvolusi, hasil keluarannya adalah berupa satu *channel* pada *output tensor*. Selanjutnya beralih kepada *filter* yang kedua. Proses serupa berjalan dan menghasilkan satu *channel* baru pada *output tensor* [34].

Setelah proses konvolusi selesai, *CNN* akan mengaplikasikan *activation layer* atau fungsi aktivasi untuk mentransformasi *feature map* atau nilai keluaran dari *block* konvolusi. Tujuannya adalah memperkenalkan *non-linearity* pada model. Ada banyak jenis fungsi aktivasi yang tersedia seperti yang tertera pada section 2.1.5

Lapisan konvolusi ini yang menyebabkan struktur *CNN* memiliki hierarki, dimana lapisan akhir dapat melihat *pixel* dari lapisan sebelumnya.[12]

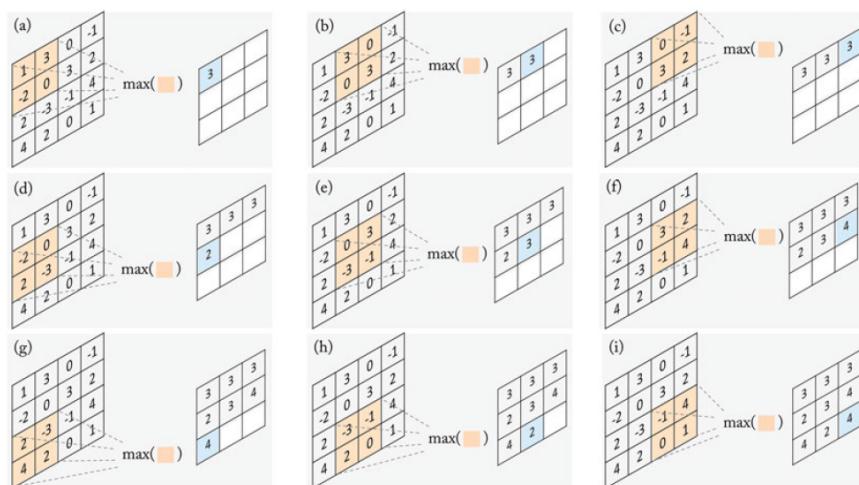


Gambar 2.13 Hierarki CNN[12]

Sebagai contoh, pada gambar 2.13 hendak mengidentifikasi apakah terdapat sepeda atau tidak. Pikirkan bahwa sepeda adalah gabungan dari beberapa bagian seperti roda, pedal, *frame*, dan lain-lain. Setiap bagian pada sepeda adalah *lower level pattern* pada *neural network* dan kombinasi bagiannya adalah *higher level pattern*. Lapisan konvolusi mengkonversi gambar menjadi bilangan numerik, membantu *neural network* dalam menginterpretasikan dan mengekstraksi pola yang berhubungan [12].

2.1.2.2 Pooling Layer

Lapisan *pooling* diletakkan di antara lapisan-lapisan konvolusi. Lapisan *pooling* mengukur nilai maksimum atau rata-rata dari fitur-fitur daerah data masukan.[5].



Gambar 2.14 Pooling Layer[4]

Grafik di atas menggambarkan bagaimana *max pooling* beroperasi. Pada jaringan, setiap pemetaan fitur yang diletakan pada lapisan *pooling* dijadikan sampel, dan jumlah fitur yang dipetakan tidak berubah, tetapi ukuran setiap *feature map* akan menjadi semakin kecil. Tujuan untuk menggunakan lapisan *pooling* adalah untuk meminimalisasi jumlah kalkulasi dan menolak perubahan karena *microdisplacement* dengan mempertahankan fitur paling penting dari data.[5].

Pooling layer disebut juga *downsampling*, melakukan *dimensionality reduction*, mengurangi jumlah *parameter* pada data masukan. Mirip dengan *convolution*, pada *pooling* juga terdapat *filter* yang bergeber ke seluruh *image field*. Namun perbedaannya dengan *convolution layer* adalah tidak adanya bobot pada *kernel*-nya[12]. Dengan adanya *pooling layer*, akan banyak informasi yang hilang, tetapi ini merupakan keuntungan bagi *CNN*. Mengurangi kompleksitas, meningkatkan efisiensi dan mengurangi resiko *overfitting*.

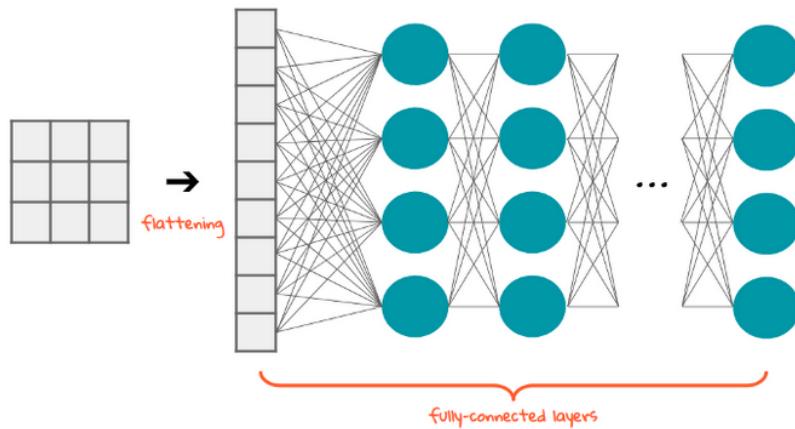
2.1.2.3 Fully Connected Layer

Pixel hasil keluaran dari proses *convolution* dan *pooling* bukanlah *pixel* yang pada mulanya merupakan *pixel* yang bersebelahan, oleh karena itu sebelumnya perlu dilakukan proses *flattening*, yaitu proses yang mengkonversi data menjadi larik satu dimensi. Setelah *pixel* data menjadi data yang *flat*. Proses selanjutnya adalah mengirimkan data ke *fully connected layer*. Lapisan ini memiliki fungsi mengklasifikasikan berdasarkan fitur yang diekstraksi pada *layer* sebelumnya. Fungsi aktivasi yang digunakan bisa beragam di antara *fully connected layer* yang ada, namun pada *output layer* yang biasa digunakan adalah fungsi aktivasi *softmax*, karena menghasilkan probabilitas keseluruhan kelas klasifikasi pada satu *data point*-nya [13]. Perhitungan matematis dari *fully connected layer* dapat dilihat pada persamaan 2.5.

$$y = f(W^T x + b) \quad (2.5)$$

Keterangan :

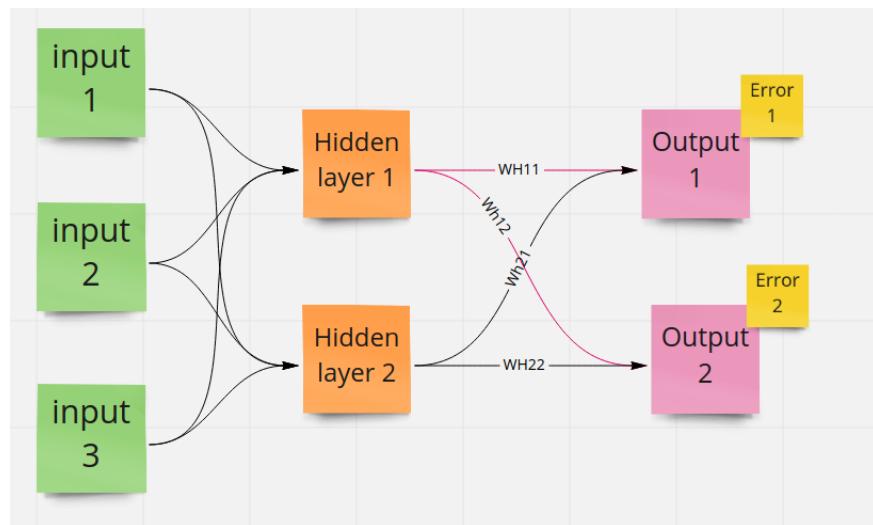
- | | |
|-------|--|
| W^T | : matriks yang berisi bobot sambungan antar <i>layer</i> |
| f | : fungsi aktivasi |
| x | : vektor yang dihasilkan dari <i>feature map</i> |
| b | : vektor <i>bias</i> |
-



Gambar 2.15 Fully Connected Layer[14]

2.1.3 Backward Propagation

Backpropagation pada *Neural Network* bertujuan untuk melakukan pembaruan nilai bobot pada *hidden layer*. Besarnya perubahan nilai bobot pada *hidden layer* bergantung kepada nilai *error* yang dihasilkan oleh *output node*.



Gambar 2.16 Backward Propagation

$$EH_n = WH11 * error1 + \dots + WH1n * errorn \quad (2.6)$$

$$\begin{bmatrix} EH1 \\ EH2 \end{bmatrix} = \begin{bmatrix} WH11 & WH21 \\ WH12 & WH22 \end{bmatrix} \begin{bmatrix} error1 \\ error2 \end{bmatrix} \quad (2.7)$$

$$EH = W^T \cdot E \quad (2.8)$$

Keterangan :

EH : Error Hidden Layer

WH : Weight Hidden Layer

Perhatikan gambar 2.16 di atas, nilai *error* yang dihasilkan adalah hasil selisih dari nilai prediksi dengan nilai sesungguhnya. Pada persamaan 2.6 kita bisa mengetahui bahwa total *error* pada suatu *hidden layer* adalah total dari hasil perkalian antara semua bobot *hidden layer* dengan nilai *error* nilai keluarannya [25].

Penulisan formula *error* pada *hidden layer* di atas dapat dituliskan dalam notasi vektor seperti pada persamaan 2.7. Rumusan yang lebih sederhana lagi dapat dilihat pada persamaan 2.8.

Pada kasus nyata, akan ada banyak ragam *cost function* yang bisa dipakai. Misalkan apabila menggunakan *mean square error* seperti pada persamaan 2.11, maka saat hendak melakukan *backward propagation* kita perlu mencari *derivation* atau turunan dari fungsi tersebut, yaitu pada persamaan 2.16 dan 2.14. Sehingga persamaan *gradient descentnya* dapat dilihat pada persamaan 2.19 dan 2.21. Dari persamaan diatas dapat diturunkan rumusan untuk merubah nilai bobot pada proses *backward propagation* yaitu sebagai berikut

$$\Delta_w = \alpha * Error * af' \cdot x^T \quad (2.9)$$

BAB 2 LANDASAN TEORI

Keterangan :

Δ_w : perubahan bobot

α : *learning rate*

$a f'$: *derivative of activation function*

x^T : *transpose of x (input value)*

Pada persamaan 2.9 kita perlu mencari turunan dari *activation function* karena kita perlu untuk mengubah bobot pada *hidden layer*, dimana nilai keluaran dari *hidden layer* tersebut merupakan hasil keluaran dari fungsi aktivasi yang dipakai [25].

Backpropagation pada *convolutional neural network* bertujuan untuk melakukan pembaruan bobot pada *kernel* yang digunakan untuk melakukan deteksi fitur. Pembaruan ini dilakukan agar *kernel* dapat melakukan deteksi fitur dengan lebih baik dengan mengeksekusi suatu metode *weight optimizer* sehingga dapat diperoleh nilai *loss* seminimal mungkin [25].

2.1.3.1 Gradient Descent

Gradient adalah ukuran yang menunjukkan pengaruh perubahan suatu *variable* terhadap nilai keluarannya. Menurut Lex Fridman (MIT), *gradient* adalah ukuran seberapa besar nilai keluaran dari suatu fungsi berubah jika nilai masukannya dirubah [8]. Dalam *Deep Learning*, *Gradient* adalah pengaruh perubahan nilai bobot terhadap nilai *error*. *Gradient Descent* adalah suatu proses *iterative*/berulang yang membawa pada nilai minimum suatu fungsi [8].

$$\theta^1 = \theta^0 - \alpha \nabla J(\theta) \quad (2.10)$$

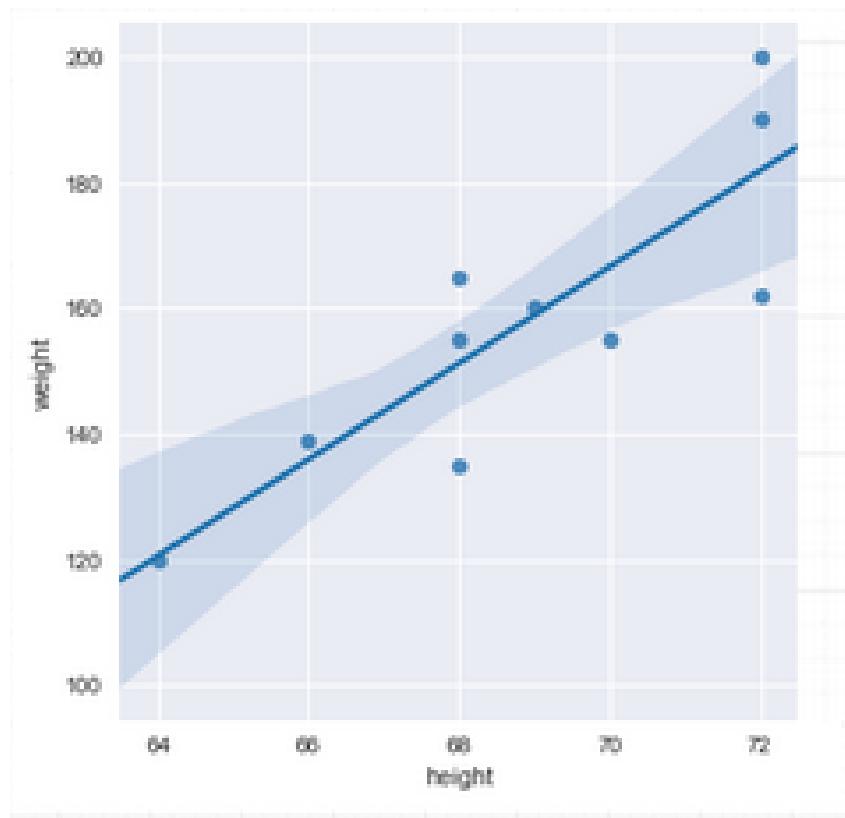
Keterangan :

θ^1 : posisi berikutnya

θ^0 : posisi sekarang

α : *learning rate*

$\nabla J(\theta)$: arah menuju kenaikan tercepat. (Turunan parsial *cost function* terhadap *gradient* dan *bias*).



Gambar 2.17 Contoh Gradient Descent

Sebagai contoh, untuk memprediksi berat badan berdasarkan tinggi badan, dan sudah terdapat data seperti yang terlihat pada gambar 2.17. Langkah pertama adalah dengan menginisiasi nilai *gradient* (m) dan *intercept*(b) secara *random* sebagai kondisi awal $Y = mx + b$. Garis tersebut adalah model yang akan kita pakai untuk memprediksi, *gradient descent* membantu mengoptimalkan model dengan meminimalkan nilai *error*-nya. *Gradient Descent* bekerja dengan formula seperti pada persamaan 2.10.

2.1.3.2 Cost Function

Cost Function adalah fungsi yang digunakan untuk mengukur performa suatu model algoritma *machine learning*. Pada *machine learning*, *cost function* digunakan untuk memahami perbedaan antara nilai yang diprediksi dan nilai sebenarnya. Biasanya *cost function* digunakan pada algoritma *supervised learning* yang menggunakan teknik optimasi[10].

$$J_{m,b} = \frac{1}{N} \sum_{i=1}^N (Error_i)^2 \quad (2.11)$$

Keterangan :

J : Cost Function

N : jumlah neuron

Error : perbedaan antara nilai yang diprediksi dengan nilai yang sebenarnya

Gradient Descent adalah salah satu contoh optimasi, dalam proses optimasinya *gradient descent* menggunakan *cost function*. Jika kita meninjau kembali persamaan 2.10, dapat dilihat bahwa kita melakukan turunan parsial pada *cost function* yang kita gunakan.

$$\frac{\partial J}{\partial m} = 2.Error \cdot \frac{\partial}{\partial m} Error \quad (2.12)$$

$$\frac{\partial J}{\partial b} = 2.Error \cdot \frac{\partial}{\partial b} Error \quad (2.13)$$

Seperti pada persamaan 2.12 dan 2.13 dapat dilihat bahwa *cost function* diturunkan secara parsial terhadap *gradient* dan *bias*. Nilai *error* sesungguhnya adalah selisih antara nilai yang diprediksi dengan nilai yang sesungguhnya.

$$\frac{\partial}{\partial m} \text{Error} = \frac{\partial}{\partial m} (mX + b - Y) \quad (2.14)$$

$$\frac{\partial}{\partial m} \text{Error} = X \quad (2.15)$$

$$\frac{\partial}{\partial b} \text{Error} = \frac{\partial}{\partial b} (mX + b - Y) \quad (2.16)$$

$$\frac{\partial}{\partial b} \text{Error} = 1 \quad (2.17)$$

Dengan menggabungkan persamaan 2.10, 2.13, 2.17, 2.12, 2.15 diperoleh rumusan lengkap dari formula *gradient descent*.

$$\frac{\partial J}{\partial m} = \text{Error} * X * \alpha \quad (2.18)$$

$$m^1 = m^0 - \text{Error} * X * \alpha \quad (2.19)$$

$$\frac{\partial J}{\partial b} = \text{Error} * \alpha \quad (2.20)$$

$$b^1 = b^0 - \text{Error} * \alpha \quad (2.21)$$

BAB 2 LANDASAN TEORI

Keterangan :

m^0 : gradient awal

m^1 : gradient baru

b^0 : bias awal

b^1 : bias baru

α : learning rate

X : nilai sekarang

$Error$: perbedaan antara nilai yang diprediksi dengan nilai yang sebenarnya

Pada *Deep learning*, *cost function* yang banyak digunakan adalah menggunakan *mean square error*, seperti yang dapat dilihat pada contoh persamaan 2.11. Ini dikarenakan lebih mudah untuk mendapatkan nilai turunan dari garis regresinya, juga dengan menggunakan *square error*, data yang *error* diperjelas, sehingga terlihat perbedaannya dengan nilai yang ditargetkan [9].

1. **Loss Function** : mengacu kepada error dari *single training data*.
2. **Cost Function** : mengacu kepada rata rata dari keseluruhan *loss function* pada keseluruhan *training set*.

Pada dasarnya *cost function* menunjukkan seberapa akurat model memberikan prediksi untuk setiap nilai *gradient* dan *bias* [9]. Ada beberapa jenis *cost function*:

1. *Regression Cost Function*

Model regresi berurusan dengan nilai kontinu, contohnya seperti gaji karyawan, prediksi peminjaman, harga mobil, harga rumah. Perhitungan *error*-nya adalah hasil selisih dari nilai prediksi dengan nilai yang sebenarnya. Beberapa contoh *regression cost function* adalah :

- *Mean Error*

masing-masing *error* pada *data point* dihitung, lalu dijumlah dan dihitung rata-ratanya. Hasilnya sendiri bisa saja negatif atau positif. Pada saat tahap penjumlahan *data point*, bisa saja saling *cancel* satu sama lain. Oleh karena itu *Mean Error* tidak efektif untuk dijadikan *cost function*.

- *Mean Square Error*

untuk mengatasi *cancelling* pada *Mean Error*, pada *MSE* hasil *error*-nya dikuadratkan. *MSE* adalah rata-rata dari jumlah kuadrat *error* tiap *data point*. Disebut juga dengan *L2 Loss*. Karena perhitungan *error*-nya dikuadratkan, *MSE* membantu mempinaltikan deviasi kecil, tetapi jika

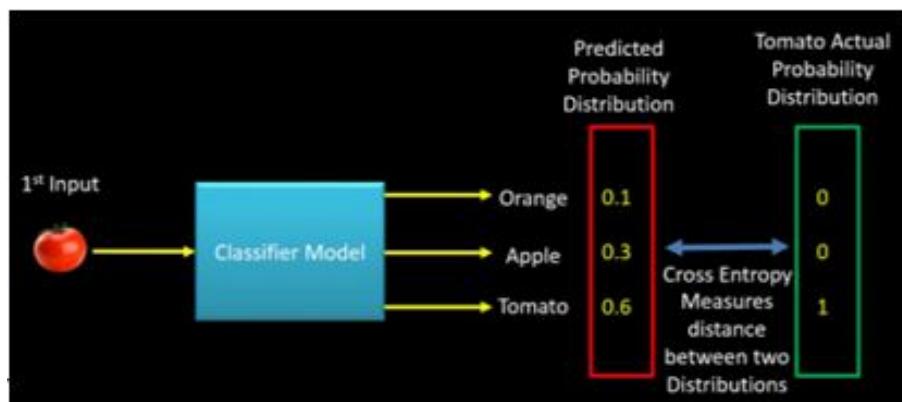
dataset memiliki *outlier*, *MSE* akan membesarkannya. Karena itu, *MSE* memiliki kekurangan yaitu tidak *robust* terhadap *outlier*.

- *Mean Absolute Error*

pendekatan *MAE* adalah dengan memutlakan *error* pada setiap *datapoint*. Disebut juga *L1 Loss*, sistem ini *robust* terhadap *outlier*.

2. Classification Cost Function

Cost function pada kasus klasifikasi berbeda dengan *cost function* yang digunakan pada regresi, yang paling umum digunakan adalah *Cross Entropy*. Sebagai contoh, pada klasifikasi kelas (*Orange*, *Apple*, *Tomato*), maka model *machine learning* akan menampilkan distribusi probabilitas terhadap tiga kelas tersebut. *Output* = $[P(\text{Orange}), P(\text{Apple}), P(\text{Tomato})]$. Jika nilai masukan adalah tomat, maka ekspektasinya adalah nilai distribusi probabilitas cenderung ke arah tomat, jika tidak maka perlu ada penyesuaian bobot dari arsitektur *neural network*. Dengan kata lain, *Cross Entropy* adalah cara untuk mengukur jarak antara dua distribusi probabilitas [10]



Gambar 2.18 *Cross Entropy* [10]

$$loss(y, p) = -[y_1, y_2, y_n] \begin{bmatrix} \log(p_1) \\ \log(p_2) \\ \log(p_3) \end{bmatrix} \quad (2.22)$$

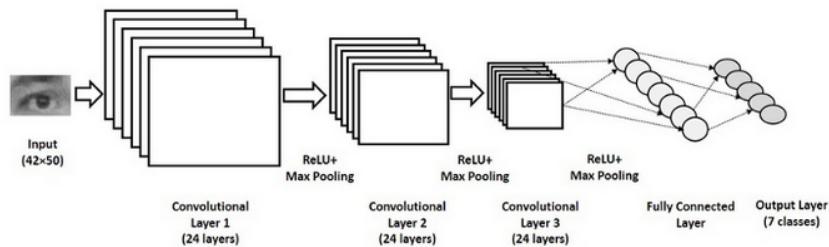
Keterangan :

y : probabilitas target

p : probabilitas yang dihasilkan

persamaan 2.22 di atas adalah formula untuk menghitung *loss* dari satu *data point*. Pada *real case*, yang kita hitung ada *cost function* dari sistem, yang artinya kita harus menghitung *loss* dari semua *data point*. Untuk formula perhitungan *cost function* akan bergantung kepada pemilihan peneliti, baik *mean square error* ataupun *mean absolute error*. Namun *mean square error* banyak digunakan karena kemudahan penurunan saat *back propagation* [10].

2.1.4 Arsitektur CNN



Gambar 2.19 Convolutional Neural Network [14]

CNN mengklasifikasi gambar dengan cara mengekstraksi fitur-fitur yang terdapat pada gambar melalui *convolution layer*, hasil keluarannya nanti akan dibuat menjadi *flat*. Selanjutnya *feature map* yang sudah menjadi *flat* (lariuk satu dimensi) akan dikirim ke *fully connected layer* yang akan berujung pada *output layer* yang memiliki fungsi aktivasi *softmax*. *Output* yang dihasilkan berupa probabilitas masing masing kelas klasifikasi. Perbedaan arsitektur pada CNN mengakibatkan perubahan cara ekstraksi fitur pada gambar.

2.1.4.1 VGG-16

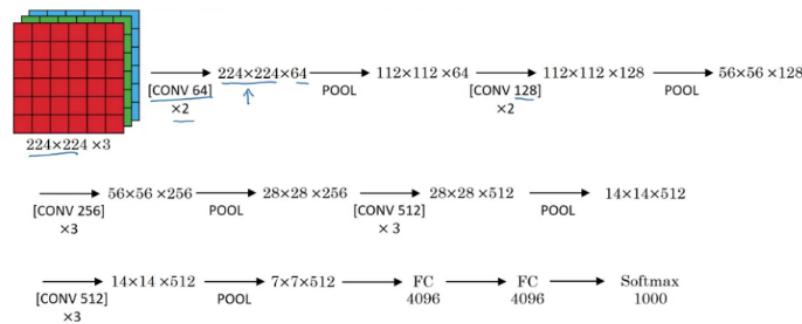
Ide dari *VGG16* adalah menggunakan *network* yang lebih sederhana dibandingkan dengan pendahulunya (*AlexNet*). Yaitu dengan menggunakan layer konvolusi 3×3 dengan *stride* sebesar satu dan juga *same padding*. Di akhir setiap proses konvolusi, digunakan *Max Pooling* dengan nilai sama dengan 2 dan *stride* sebesar 2[17].

Dimensi masukan data pada arsitektur *VGG16* adalah $224 \times 224 \times 3$. Gambar masukan kemudian dikirimkan kepada *convolution layer* dengan *filter* 64 sebanyak 2 kali. Selanjutnya dikirimkan kepada *convolution layer* dengan *filter* 128 sebanyak 2 kali. Selanjutnya, kembali dikirimkan kepada *convolution layer*

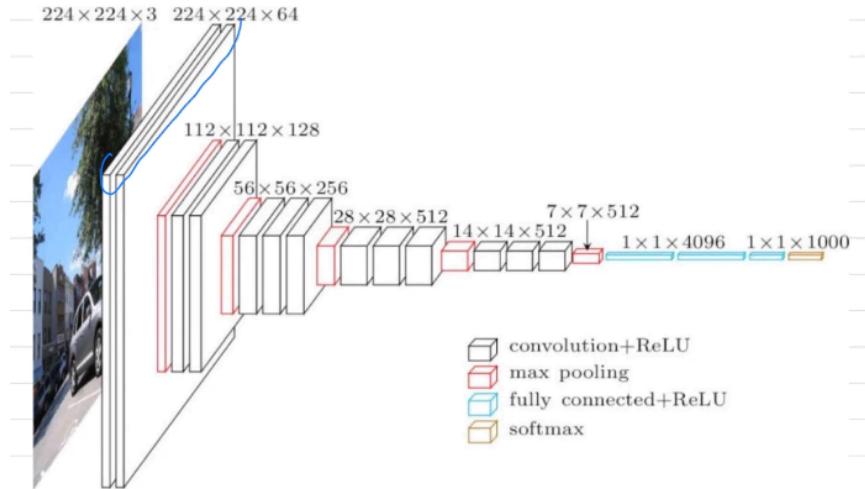
dengan *filter* 256 sebanyak 3 kali. Selanjutnya dikirimkan kembali kepada *convolution layer* dengan *filter* 512 sebanyak 3 kali. Lalu dikirimkan kepada *convolution layer* dengan *filter* 512 sebanyak 3 kali. Sebelum pada akhirnya dijadikan *flat array* dan dikirimkan ke *fully connected layer / Dense Layer* dengan *filter* 4096 sebanyak 2 kali. Lalu *output layer* dengan aktivasi *softmax*.

Data masukan pada sistem memiliki tiga *channel*, lalu data masukan tersebut dikirimkan kepada 64 *kernel* dengan dimensi 3×3 . yang terjadi disini adalah *kernel* yang sebenarnya dilewati adalah $64 \times 3 \times 3 \times 3$. [34]. Pada iterasi pertama terdapat tiga *kernel* yang dilewati pada tiga *channel* gambar. Hasil konvolusi masing masing *channel* akan dijumlahkan menjadi nilai baru pada *output channel*. Begitu seterusnya, sehingga hasil akhirnya adalah 64 *channel*.

Pada akhir setiap *block convolution*, dapat ditambahkan *dropout layer* ataupun *batch normalization* sebagai *reguralization* dan mengurangi resiko *overfitting*[20]. Berdasarkan pembahasan oleh Dr. Adrian Rosebrock pada buku [20], pada penelitian ini diputuskan untuk mengaplikasikan *batch normalization* dan *dropout* pada arsitektur VGG16.



Gambar 2.20 VGG16 [18]



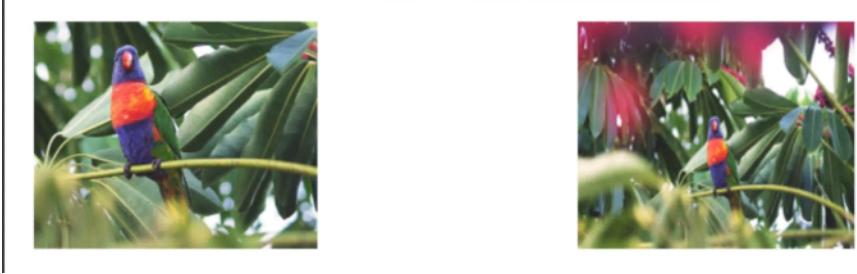
Gambar 2.21 VGG16 [18]

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224×224 RGB image)					
conv3-64	conv3-64 LRN	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 conv1-256	conv3-256 conv3-256 conv1-256	conv3-256 conv3-256 conv3-256 conv3-256
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
FC-4096	FC-4096				
FC-4096	FC-4096				
FC-1000	FC-1000				
soft-max	soft-max				

Gambar 2.22 VGG16 [17]

2.1.4.2 Inception

Ide utama dari arsitektur *inception* adalah, kita menggunakan lebih dari satu *filter* dalam sekali konvolusinya. *Filter* ini berjalan secara *parallel* dan *feature map* hasil dari setiap *filter* konvolusi akan di *concat* menjadi satu kesatuan *feature map*[18].

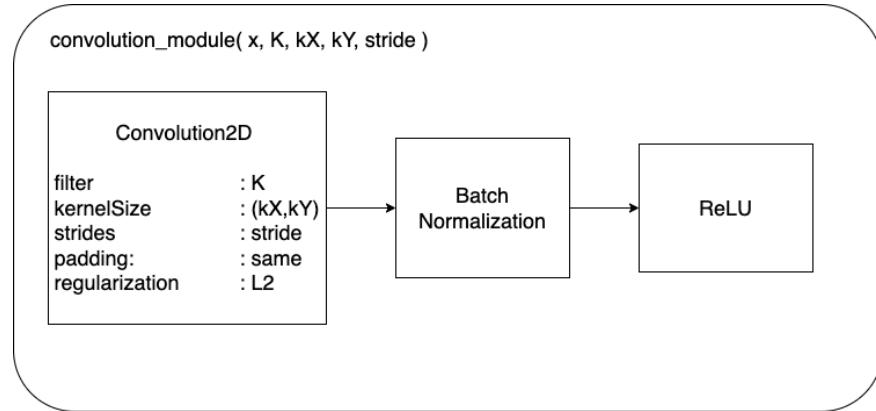


Gambar 2.23 *Inception problem* [18]

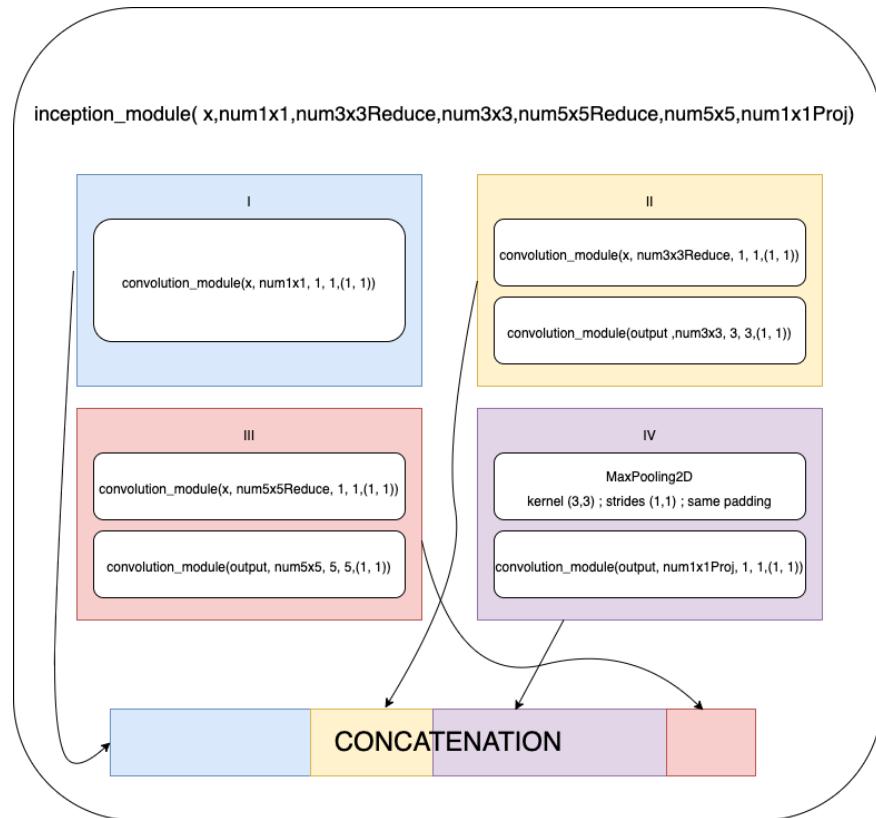
Dapat dilihat pada gambar 2.23, arsitektur *Inception / GoogleNet* berusaha untuk memecahkan masalah ini. Bayangkan jika menggunakan satu jenis *filter* yang ukurannya kecil, *filter* tersebut dalam satu kali *pass* dapat mencakup keseluruhan gambar burung pada gambar di sebelah kanan, akan tetapi untuk gambar di sebelah kiri, *filter* tersebut tidak cukup memuat seluruh tubuh burung dalam satu kali *pass*. Apabila menggunakan lebih dari satu ukuran *filter* (ukuran *filter*-nya beragam), masalah ini bisa dipecahkan. Atas dasar ini, maka *Inception* menggunakan beragam jenis *filter* [18].

Pada arsitektur *Inception* terdapat dua jenis *module* (kumpulan beberapa *layer* yang dipakai berulang), yaitu *convolution module* dan *inception module*. Pada gambar 2.24 dapat dilihat ilustrasi *convolution module*, merupakan sebuah fungsi yang menerima parameter berupa banyaknya *filter*, ukuran *filter* dan besar nilai *strides*. Di dalam fungsi *convolution module* terdapat proses konvolusi dua dimensi dengan *same padding* yang dilanjutkan oleh *batch normalization* lalu diakhiri oleh fungsi aktivasi berupa *ReLU*. Keluaran dari fungsi ini tentunya adalah *feature map*. Inisialisasi nilai bobot pada *kernel convolution* menggunakan teknik regularisasi *L2*. Beberapa *convolution module* menyusun suatu *inception module*. Dalam arsitektur *Inception* terdapat tujuh *inception module* dan tiga *convolution module* yang dipakai. Proses *concatenation feature map* berlangsung pada *inception module* [18].

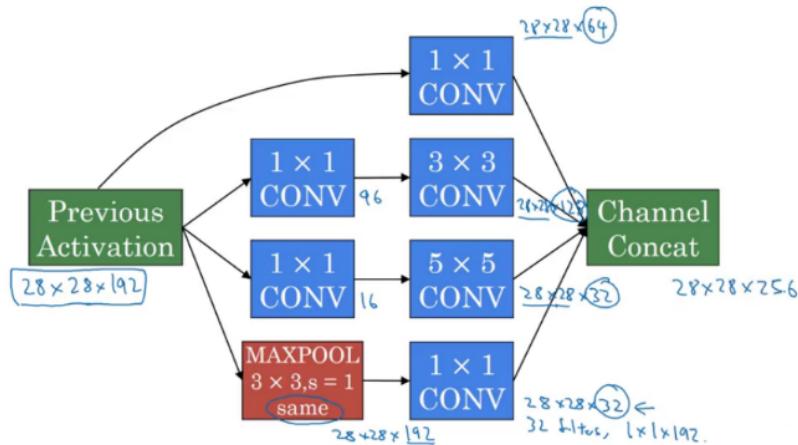
BAB 2 LANDASAN TEORI



Gambar 2.24 Convolution module



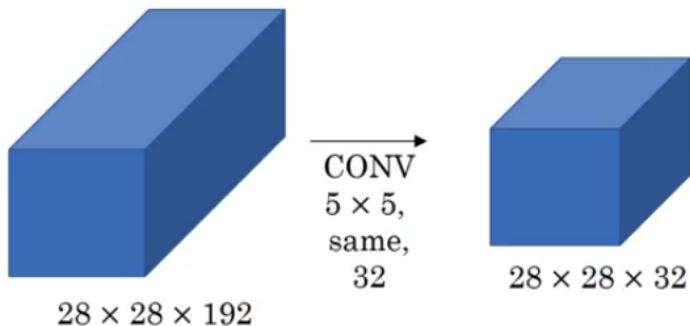
Gambar 2.25 Inception module



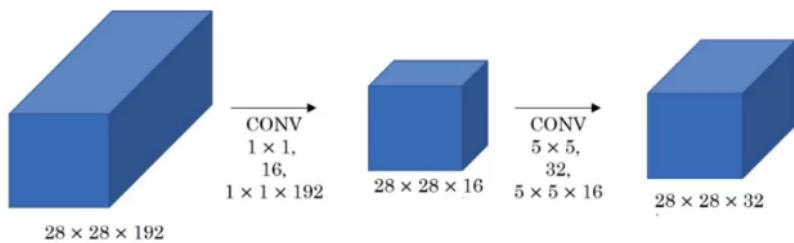
Gambar 2.26 Inception module architecture [18]

Pada gambar 2.25 dapat dilihat bahwa digunakan konvolusi berdimensi 1×1 . Fungsi dari konvolusi ini adalah untuk mengurangi jumlah parameter yang digunakan saat melakukan ekstraksi fitur. Seperti yang sudah dijelaskan pada subbab 2.1.2, satu filter pada proses konvolusi akan diaplikasikan kepada seluruh *channel* masukan, sehingga satu *filter* konvolusi akan menghasilkan satu *channel output* berupa *feature map*. Pada gambar 2.27, dapat dilihat suatu *filter* konvolusi berdimensi $5 \times 5 \times 32$ dengan *same padding* hendak memproses *tensor input* dengan dimensi $28 \times 28 \times 192$. Pada kasus ini besar biaya komputasi yang dibutuhkan adalah $5 \times 5 \times 192 \times 28 \times 28 \times 32$, karena filter akan dikonvolusi ke seluruh *channel* pada *input tensor*, lalu dikalikan kembali oleh dimensi keluaran yaitu $28 \times 28 \times 32$. Sehingga total biaya komputasi yang dibutuhkan adalah

$$5 \times 5 \times 192 \times 28 \times 28 \times 32 = 120422400$$



Gambar 2.27 Normal Convolution [18]



Gambar 2.28 1×1 convolution layer [18]

Jika dilalui terlebih dahulu *filter* 1×1 , maka jumlah parameternya adalah

$$(28 \times 28 \times 16 \times 1 \times 1 \times 192) + (28 \times 28 \times 32 \times 5 \times 5 \times 16) = 12443648$$

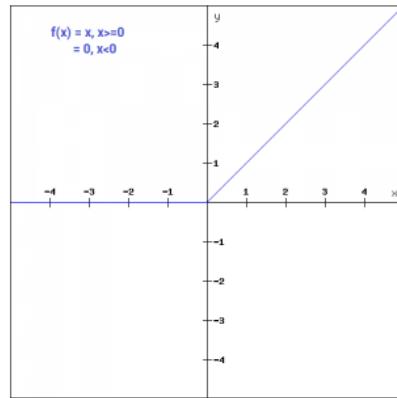
Dapat dilihat bahwa penggunaan 1×1 convolution membantu mengurangi biaya komputasi. Teknik ini banyak dipakai oleh arsitektur yang rumit seperti *Squeeze Net*, *ResNet* dan *Inception*.

2.1.5 Fungsi Aktivasi

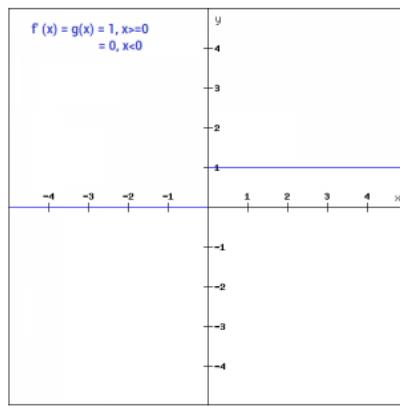
Tujuan dari fungsi aktivasi adalah memperkenalkan model *neural network* pada fungsi *non-linear*. Apabila kita tidak mengaplikasikan fungsi aktivasi pada *neural network* maka hasilnya akan sama dengan *linear regression* biasa. Terdapat banyak jenis fungsi aktivasi, namun yang dipakai pada penelitian ini hanyalah *Rectified Linear Unit (ReLU)* dan juga *softmax*.

2.1.5.1 Rectified Linear Unit (*ReLU*)

Fungsi aktivasi ini tidak mengaktifkan semua *neuron* secara bersamaan, *neuron* yang hasil keluarannya bernilai dibawah 0 tidak akan diaktifkan [26].



Gambar 2.29 ReLU [26]



Gambar 2.30 derivative of ReLU [26]

Namun apabila melihat grafik turunan dari ReLU (gambar 2.30, bisa dilihat bahwa *neuron* yang hasil keluarannya negatif, memiliki nilai *gradient* 0. Ini artinya ReLU akan menghasilkan *dead neuron* yaitu neuron yang bobotnya tidak akan pernah diupdate saat *backpropagation*. Sementara sisanya akan diupdate dengan faktor konstan [26].

2.1.5.2 Softmax

Dideskripsikan sebagai kombinasi dari banyak fungsi *Sigmoid*. Nilai keluaran dari fungsi *sigmoid* adalah dalam rentang 0 hingga 1. *Sigmoid* biasanya hanya digunakan pada *binary classification*, sementara *Softmax* digunakan pada *multi class classification*. Nilai keluaran dari *Softmax* adalah probabilitas dari satu *data point* terhadap semua kemungkinan kelas [26].

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^K e^{z_k}} \quad (2.23)$$

BAB 2 LANDASAN TEORI

Keterangan :

$\sigma(z)_j$: softmax probability of class j

e : sigmoid probability

Persamaan *loss function*-nya adalah sebagai berikut

$$\text{Loss} = -\log(\phi_i) \quad (2.24)$$

Keterangan :

Loss : nilai *loss*

ϕ_i : hasil fungsi *Softmax* yang menunjukkan probabilitas objek ke- i

2.1.6 Weight Initialization

Weight atau bobot memiliki peranan penting untuk koneksi antara unit-unit dalam *neural network*. Dapat diinisialisasi secara random dan diubah saat *backpropagation* dalam rangka untuk mengurangi *loss* [31].

Terdapat beberapa metode dalam menginisialisasi bobot *kernel*. Terdapat *Normal Initialization*, *Uniform Initialization*, *Xavier Normal*, *Xavier Uniform*, *He Normal*, *He Uniform* [31]. Pada penelitian kali ini digunakan *He initialization* karena metode ini mendukung fungsi aktivasi *ReLU*.

2.1.6.1 He Initialization

Ketika menggunakan fungsi aktivasi *ReLU*, hasil standard deviasi dari *activation layer* dekat dengan nilai akar dari 2 jika menggunakan inisialisasi *xavier*. Untuk mengatasi ini, maka diciptakanlah metode baru yaitu *He Initialization* [31].

$$W = N(\mu, \sigma) \quad (2.25)$$

$$\sigma = \sqrt{\frac{2}{f_{in}}} \quad (2.26)$$

$$W = U(a, b) \quad (2.27)$$

$$a = -\sqrt{\frac{6}{f_{in} + f_{out}}} \quad (2.28)$$

$$b = \sqrt{\frac{6}{f_{in} + f_{out}}} \quad (2.29)$$

Keterangan :

W : nilai bobot, ada di antara a hingga b

f_{in} : nilai masukan

f_{out} : nilai keluaran

2.1.7 Regularization

Optimizer memiliki tujuan untuk mengurangi nilai *loss* sementara *regularization* memiliki tujuan untuk mengeneralisasi model agar tetap menghasilkan nilai akurasi tinggi meskipun menggunakan *dataset* yang berbeda dari data latihnya [32]. Penelitian ini menggunakan tiga metode *regularization*: L2, *batch normalization*, *image augmentation* dan *dropout*.

2.1.7.1 L2

L2 mengestimasi *mean* dari data untuk menghindari *overfitting* dan membantu mengurangi *loss* pada *gradient descent*. Regularisasi L2 digunakan untuk menghitung angka (disebut penalti) yang ditambahkan ke nilai *loss* untuk melakukan penalti kepada bobot dan *bias* yang bernilai besar dari model yang dibangun. Bobot besar mungkin menunjukkan bahwa sebuah neuron mencoba

BAB 2 LANDASAN TEORI

menghafal elemen data. Padahal, akan lebih baik jika lebih banyak neuron yang berkontribusi pada keluaran model, daripada hanya menggunakan sedikit neuron saja [35].

Nilai penalti regularisasi L_2 dihitung dari jumlah kuadrat bobot dan bias. Pendekatan nonlinear ini akan cenderung melakukan penalti kepada bobot dan *bias* yang lebih besar daripada yang lebih kecil. Hal ini membuat regularisasi L_2 lebih umum digunakan karena tidak memengaruhi nilai parameter kecil secara substansial dan memungkinkan model untuk menghasilkan bobot yang tidak terlalu besar dengan memberi penalti berat pada nilai yang relatif lebih besar [35]. Persamaan 2.30 dan 2.31 masing-masing digunakan menghitung penalti bobot dan *bias* dalam regularisasi L_2 [35]. Dalam penelitian ini, akan digunakan konstanta regularisasi λ dengan nilai $0.01 (10^{-2})$ untuk arsitektur *VGG16* dan $0.0005 (5 \times 10^{-4})$ untuk arsitektur *Inception*.

$$L_{2w} = \lambda \sum_m w_m^2 \quad (2.30)$$

Keterangan :

w : nilai bobot (*weight*)

L_{2w} : hasil regularisasi untuk bobot w

λ : konstanta regularisasi

$\sum_m w_m^2$: jumlah kuadrat bobot w dari sekumpulan *neuron m*

$$L_{2b} = \lambda \sum_n b_n^2 \quad (2.31)$$

Keterangan :

b : nilai *bias*

L_{2b} : hasil regularisasi untuk *bias b*

λ : konstanta regularisasi

$\sum_n b_n^2$: jumlah kuadrat *bias b* dari sekumpulan *bias n*

2.1.7.2 Batch Normalization Layer

adalah proses *scaling* ulang data numerik tanpa mendistorsi bentuknya. Secara umum ketika kita hendak mengirimkan data ke dalam model *machine learning*, kita cenderung mengubahnya dalam skala yang berimbang, tujuannya memastikan model dapat tergeneralisasi secara tepat. Pada *Deep learning*, implementasi *Batch Normalization* adalah dengan menambahkan satu lapisan baru yang bertugas untuk standarisasi dan normalisasi nilai masukan sehingga *neural network* dapat bekerja lebih cepat [15].

Batch Normalization digunakan untuk memitigasi *Internal Covariate Shift*, yaitu perubahan distribusi nilai masukan karena perubahan *parameter*. *Deep learning* memiliki *layer* yang berlapis-lapis dapat dipastikan bahwa nilai *parameter* akan terus menerus mengalami perubahan di setiap *passing layer*-nya. Perubahan *parameter* akan berpengaruh terhadap perubahan nilai masukan pada *layer* berikutnya [15]. *Batch Normalization* bekerja dengan cara demikian:

1. Normalisasi input

proses mengubah data untuk memiliki *mean* = 0 dan *standard deviasi* = 1.

$$\mu = \frac{1}{m} \sum h_i \quad (2.32)$$

$$\sigma = \left[\frac{1}{m} \sum (h_i - \mu)^2 \right]^{\frac{1}{2}} \quad (2.33)$$

$$h_{i(norm)} = \frac{(h_i - \mu)}{\sigma + \epsilon} \quad (2.34)$$

$$h_i = \gamma h_{i(norm)} + \beta \quad (2.35)$$

Keterangan :

μ	: batch normalization mean
m	: jumlah <i>neuron</i> pada <i>hidden layer</i> tertentu
σ	: standard deviation
ϵ	: Smoothing Term
γ	: rescaling parameter
β	: shifting parameter
h_i	: hidden layer ke i

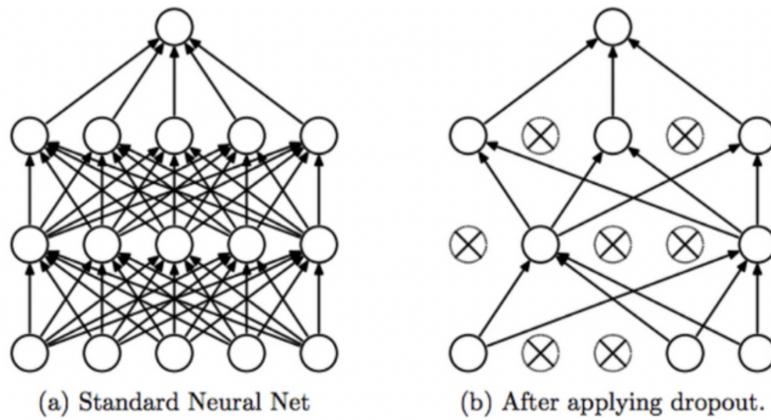
Pada proses normalisasi *input*, langkah pertama adalah menghitung *mean* dengan persamaan 2.32. Langkah selanjutnya adalah menentukan nilai standard deviasi dengan persamaan 2.33. Kemudian, menormalisasi nilai *hidden layer* berdasarkan persamaan 2.34.

2. Rescaling and Offsetting

Pada tahap ini digunakan persamaan 2.35. *Gamma* digunakan untuk *rescaling model* terhadap *mean* yang baru, sementara *beta* digunakan untuk *shifting model* terhadap *variance* yang baru. *Batch Normalization* bertindak sebagai regularisasi [15].

2.1.7.3 Dropout Layer

Dropout bekerja dengan cara menonaktifkan beberapa fitur secara acak dengan parameter dengan probabilitas p yang ditetapkan di awal inisialisasi seperti pada gambar 2.31. Fitur dinonaktifkan agar arsitektur tidak terlalu banyak beradaptasi dengan data pelatihan, sehingga memungkinkan terjadinya *generalization* yang lebih baik. *Dropout* secara acak mengurangi jumlah *neuron* yang saling berhubungan dalam *neural network*. Pada setiap iterasi pelatihan, setiap *neuron* memiliki peluang untuk dikeluarkan (*dropout*). Teknik ini meminimalkan *overfitting*, karena setiap *neuron* menjadi cukup mandiri, *neuron* dalam lapisan mempelajari nilai bobot yang tidak berdasarkan kerja sama *neuron* tetangganya. Nilai probabilitas ada dalam rentang 0 hingga 1. Semakin tinggi probabilitasnya semakin kecil kemungkinan suatu fitur dinonaktifkan. *Dropout layer* juga bekerja untuk mencegah terjadinya *co-adaptation*, yaitu suatu peristiwa dimana adanya sebagian *neuron* yang dominan sehingga setiap iterasinya menjadi semakin kuat, menyebabkan *neuron* sisanya tidak berpartisipasi.



Gambar 2.31 Dropout layer

Proses perhitungan untuk mendapatkan satu nilai seperti pada persamaan 2.36 dimulai dengan melakukan inisialisasi nilai *random* dengan rentang 0 sampai 1. Nilai probabilitas akan menentukan berapa jumlah *neuron* yang akan dinon-aktifkan pada satu iterasinya.

$$O_{(i)} = \begin{cases} 0 & \text{for } rn > p \\ O_{(i)} & \text{for } rn \leq p \end{cases} \quad (2.36)$$

Keterangan :

$O_{(i)}$: matriks keluaran posisi ke-*i*

p : nilai probabilitas

rn : nilai *random* dari 0 sampai 1

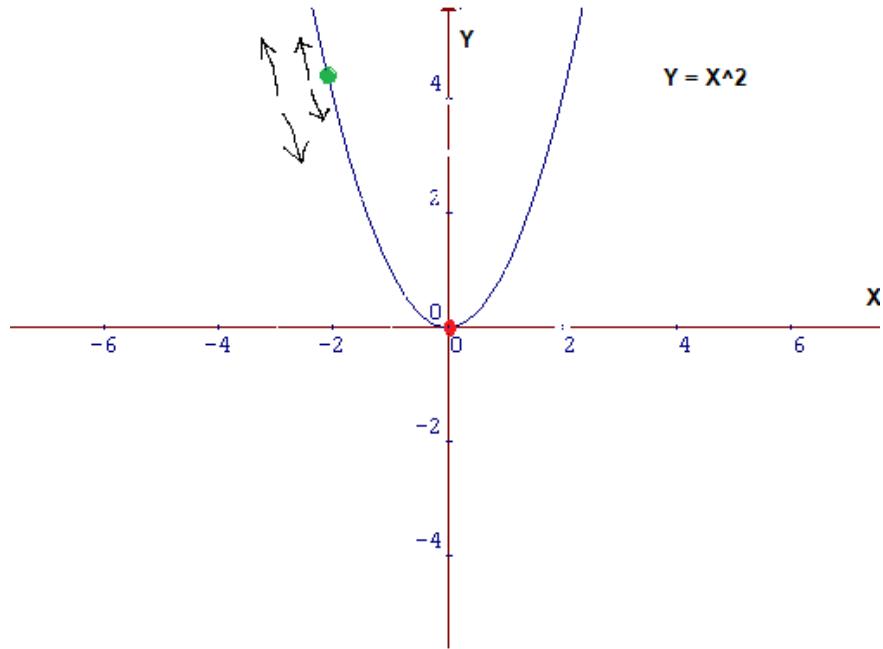
2.1.8 Optimizers

Optimizers adalah salah satu metode untuk meminimalisasi nilai keluaran dari *cost function*. Merupakan fungsi matematika yang *dependent* terhadap *learnable parameter* dari model seperti *bias* dan *weight*. *Optimizer* mempelajari bagaimana caranya mengubah *weight* dan *bias* dari *neural network* untuk mengurangi *losses* [26].

2.1.8.1 Stochastic Gradient Descent

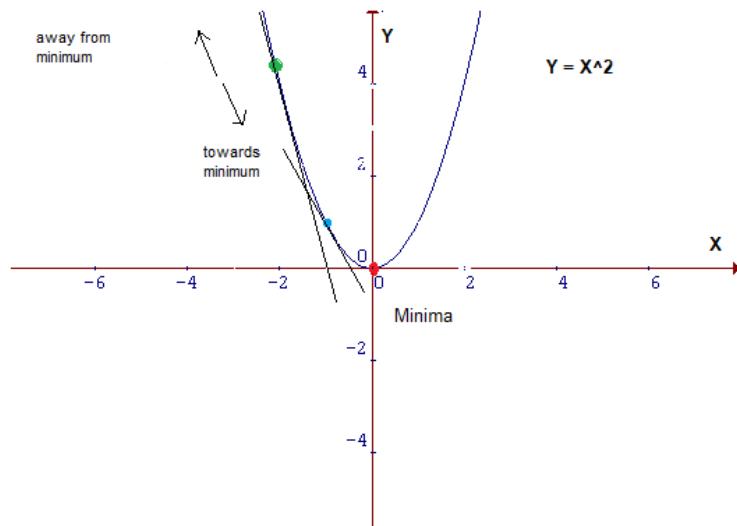
Seperti yang sudah dijelaskan pada subbab 2.1.3, *gradient* adalah ukuran yang menunjukkan pengaruh dari perubahan suatu *variable* terhadap nilai keluarannya. Dalam *Deep Learning*, *Gradient* adalah pengaruh perubahan nilai bobot terhadap

nilai *error*. *Gradient Descent* adalah suatu proses *iterative*/berulang yang membawa kepada nilai minimum suatu fungsi [8].

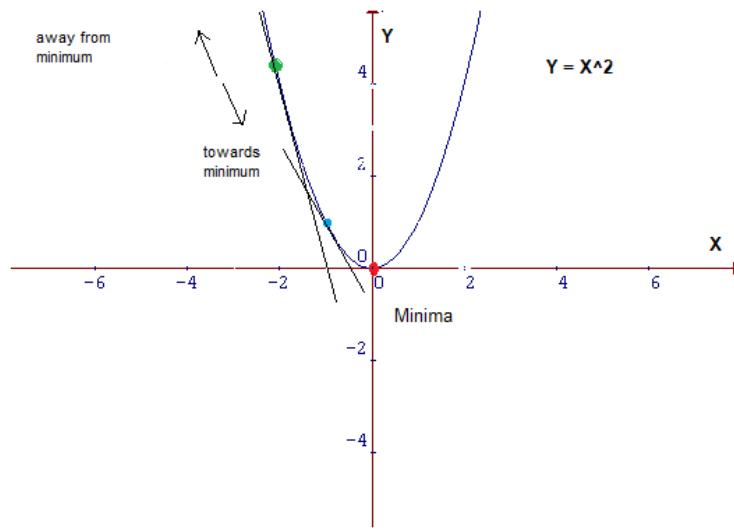


Gambar 2.32 Gradient Descent

Misalkan posisi hijau adalah posisi awal, kita hendak menuju *local minimum* — titik merah. Perlu diketahui sebelumnya, perlu bergerak ke arah mana dan seberapa besar langkah yang perlu diambil. *Gradient Descent* membantu kita mengetahui 2 hal itu dengan menggunakan *derivative* atau turunan fungsi. Turunan dari suatu fungsi dalam matematika adalah *slope* atau kemiringan suatu titik [8].



Gambar 2.33 Gradient Descent



Gambar 2.34 Gradient Descent

Slope / kemiringan tersebut membantu dalam mebentuk besarnya *step*(dalam kasus ini *learning rate*) setiap *epoch*-nya. Bila melihat grafik 2.34, dapat disimpulkan bahwa kemiringan titik hijau lebih curam dibandingkan dengan titik biru. Ini artinya *learning rate* titik hijau lebih besar dibandingkan dengan titik biru. Untuk persamaan *gradient descent* bisa dilihat pada section 2.1.3.[8].

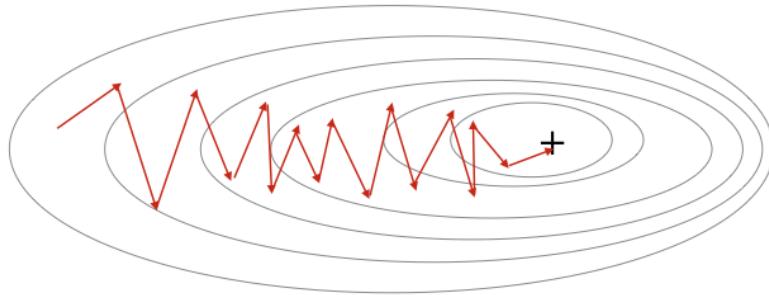
Gradient descent memiliki beberapa jenis:

1. *Vanilla Gradient Descent*

disebut juga dengan *Batch Gradient Descent*. Menghitung *error* pada setiap contoh *training dataset*. Tetapi model akan di *update* setelah seluruh *data training* dievaluasi. Satu kali prosesnya disebut *training epoch*.

2. *Stochastic Gradient Descent*

Model akan di-*update* setiap satu data latih selesai dievaluasi. Keuntungan dari metode *SGD* ini adalah prosesnya lebih cepat dibandingkan dengan *Batch Gradient Descent*, perkembangan model juga detil karena perubahan dilakukan per satu *data point*. Kelemahan dari metode ini adalah *high computational cost* dan banyaknya *noise* yang muncul pada *gradient* karena *frequent update* yang menyebabkan *error rate* nya melompat dan tidak berkurang secara bertahap [8].

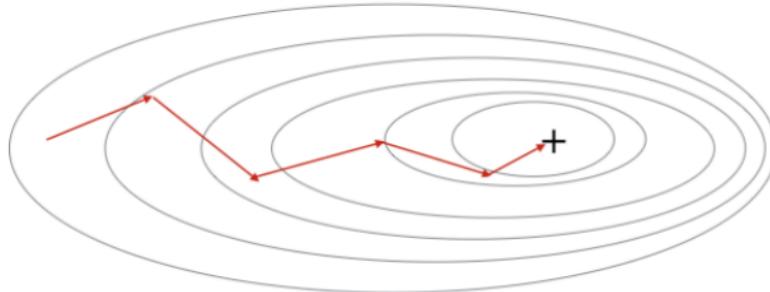


Stochastic Gradient Descent

Gambar 2.35 Stochastic Gradient Descent [8]

3. Minibatch Gradient Descent

Metode ini merupakan gabungan antara *Stochastic Gradient Descent* dan *Batch Gradient Descent*. Pada metode ini model memperbaharui nilainya setiap *batch*-nya. Keuntungan dari metode ini adalah komputasinya efisien dan konvergensinya stabil. Kekurangan pada metode ini adalah dibutuhkan agar data satu *batch*-nya berada pada *memory* agar bisa diproses [8].



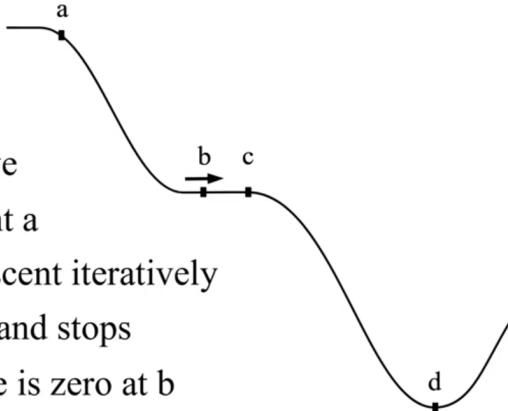
Mini Batch Gradient Descent

Gambar 2.36 Mini Batch Gradient Descent [8]

2.1.8.2 Nesterov Accelerated Gradient (NAG)

Gradient Descent memiliki kelemahan yaitu model bisa terjebak pada *local minimum*.

Consider this curve
initialize x at point a
apply gradient descent iteratively
 x reaches point b and stops
because derivative is zero at b



Gambar 2.37 Local Minimum[22]

Hal ini terjadi karena *gradient* bernilai 0 pada titik b sehingga model akan berhenti belajar (tidak ada perubahan bobot saat *backpropagation*). Masalah *local minimum* ini bisa diselesaikan dengan menggunakan *momentum*.

$$v_t = \gamma v_{t-1} + \alpha \nabla J(\theta) \quad (2.37)$$

$$\theta_{t+1} = \theta_t - v_t \quad (2.38)$$

Keterangan :

v : *velocity*

α : *learning rate*

θ : posisi

$\nabla J(\theta)$: turunan parsial *cost function*

Apabila nilai *gamma* sama dengan 0, maka persamaan diatas akan sama saja dengan *gradient descent* biasa. Pada persamaan 2.37, ditambahkan *velocity*. Nilai dari *velocity* akan bertambah besar apabila model bergerak ke arah minimum. Sebagai hasilnya kita mencapai kovergen lebih cepat [22]. Biasanya nilai dari *gamma* adalah rentang antara 0.8 hingga 0.9.

Akan tetapi kita butuh algoritma yang lebih pintar, tidak hanya mempercepat tapi bisa juga memperlambat. Seolah-olah bola yang turun di lembah yang berbukit.

Kita perlu algoritma yang mempercepat juga memperlambat bola agar bisa tinggal diam di titik minimum. Lahirlah metode *Nesterov Accelerated Gradient*.

$$v_t = \gamma v_{t-1} + \alpha \nabla J(\theta - \gamma v_{t-1}) \quad (2.39)$$

$$\theta_{t+1} = \theta_t - v_t \quad (2.40)$$

Keterangan :

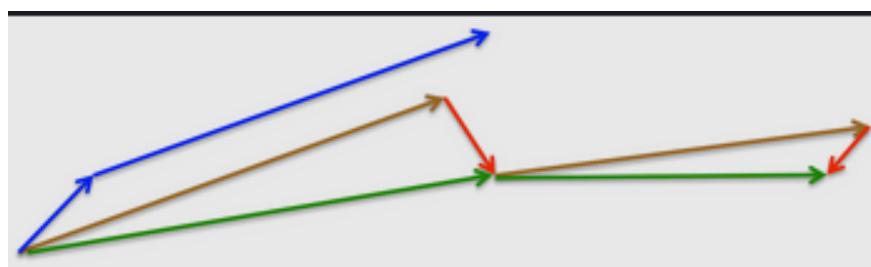
v : *velocity*

α : *learning rate*

θ : posisi

$\nabla J(\theta)$: turunan parsial *cost function*

Pergerakan NAG bisa dideskripsikan seperti gambar 2.38. Ketika momentum menghitung *gradient* dan kemudian mulai meng-update nilainya, nilainya melompat seperti pada garis biru. Perubahan yang dibuat oleh NAG mengikuti nilai akumulasi *gradient* sebelumnya (garis coklat), lalu kemudian membuat koreksi (garis merah). Garis hijau adalah hasil koreksi dari nilai perubahan *gradient* oleh NAG yang telah dikoreksi. Antisipasi perubahan ini yang mencegah perubahan yang terlalu cepat dan secara signifikan meningkatkan performa *Neural Network* [26].



Gambar 2.38 NAG movement[22]

$$v_t = m * v_0 * -\alpha * g \quad (2.41)$$

$$w_t = w_0 + v_t \quad (2.42)$$

$$\alpha_t = \alpha_0 * \frac{1}{1 + d * i} \quad (2.43)$$

Keterangan :

w_t	: bobot baru
w_0	: bobot lama
v_t	: <i>velocity</i> baru
v_0	: <i>velocity</i> lama
m	: <i>momentum</i>
g	: <i>gradient</i>
α_t	: <i>learning rate</i> baru
α_0	: <i>learning rate</i> lama
d	: nilai <i>decay</i>
i	: nilai <i>iteration</i>

Pada pustaka *keras*, persamaan NAG adalah seperti pada persamaan 2.41 dan 2.42. *Keras* mengubah nilai *learning rate* setiap iterasi sesuai dengan persamaan 2.43.

2.1.8.3 Adagrad

Algoritme *Adagrad* mengadaptasi *learning rate* pada *parameter*, mengaplikasikan perubahan kecil (*learning rate* kecil) pada *parameter* yang terjadi secara sering, dan perubahan besar (*learning rate* besar) pada *parameter* yang jarang. Karena ini, *Adagrad* cocok untuk menangani *sparse data* [22].

Adagrad menggunakan *learning rate* yang berbeda untuk setiap parameter untuk setiap *step*-nya.

$$g_{t,i} = \nabla_{\theta} J(\theta_{t,i}) \quad (2.44)$$

$$\theta_{t+1,i} = \theta_{t,i} - \alpha \cdot g_{t,i} \quad (2.45)$$

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\alpha}{\sqrt{G_{t,ii} + \epsilon}} \cdot g_{t,i} \quad (2.46)$$

$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{G_t + \epsilon}} \times g_t \quad (2.47)$$

Keterangan :

g_t : gradient pada saat step ke t

$g_{t,i}$: Turunan parsial dari fungsi terhadap parameter θ di step ke t

$G_{t,ii}$: matrix diagonal, dimana setiap diagonal elementnya (i,i) adalah jumlah kuadrat dari gradient hingga tahap ke t.

ϵ : smoothing term

Pada persamaan 2.45, disini *SGD* meng-update setiap *parameter theta* setiap step t. *Adagrad* meng-update *learning rate* setiap step-nya untuk setiap *parameter* berdasarkan perhitungan *gradient* sebelumnya.

Salah satu keuntungan dari *Adagrad* adalah mengeliminasi keperluan *learning rate tuning*. Pada implementasi umumnya menggunakan nilai 0.01. Kelemahan dari *Adagrad* adalah akumulasi dari nilai *gradient*-nya yang semakin hari semakin membesar, mengakibatkan *learning rate* menjadi semakin kecil sehingga pada akhirnya *neural network* tidak belajar [22].

2.1.8.4 *Adadelta*

Adadelta adalah *extension* dari *Adagrad*, diciptakan dengan tujuan mengatasi *learning rate* yang berkurang secara monoton. *Adagrad* menjumlahkan semua *gradient* sebelumnya, ini mengakibatkan nilai *learning rate* akan berkurang terus

menerus. Apabila nilai *learning rate* menjadi 0 pada suatu waktu tertentu, maka model tidak dapat mempelajari pola lain dari *training data*. Ide dasar dari *Adadelta* adalah menjumlahkan beberapa porsi dari *square gradient* sebelumnya dengan mengimplementasikan *decay average of square gradient* seperti pada persamaan 2.48. Dimana *gamma* adalah *decay constant* [23].

Decay constant memiliki efek yang sama dengan *alpha* pada metode momentum, yang mengatur seberapa besar pengaruh akumulasi *gradient* sebelumnya terhadap perubahan *learning rate*. Setelah didapatkan nilai akumulasi *gradient*, gunakan persamaan 2.49 (mirip dengan *Adagrad*) yaitu untuk mencari nilai *square root*-nya. Selanjutnya mengubah formula *update* dari *Adagrad* yaitu yang sebelumnya menggunakan akumulasi *gradient* dengan menggunakan *Root Mean Square* seperti pada persamaan 2.50 dan 2.51 [23].

Karena unit untuk memperbarui nilai pada *adadelta* berebeda, karenanya persamaan *decay average of square gradient* perlu ditulis ulang seperti pada persamaan 2.52. Dengan demikian dapat disimpulkan bahwa rumus *decaying average* yang baru yaitu pada persamaan 2.53. Selanjutnya adalah menghitung *Root Mean Square (RMS)*-nya [23].

Melihat pada persamaan 2.53, tidak diketahui secara pasti kuantitas *delta theta*-nya. Oleh karena itu digunakan akumulasi dari *update* sebelumnya (persamaan 2.54) [23].

$$E[g^2]_t = \gamma E[g^2]_{t-1} + (1 - \gamma) g_t^2 \quad (2.48)$$

$$RMS[g]_t = \sqrt{E[g^2]_t + \epsilon} \quad (2.49)$$

$$\Delta\theta_t = -\frac{\alpha}{RMS[g]_t} g_t \quad (2.50)$$

$$\theta_{t+1} = \theta_t + \Delta\theta_t \quad (2.51)$$

$$E[\Delta\theta^2]_t = \gamma E[\Delta\theta^2]_{t-1} + (1 - \gamma) \Delta\theta_t^2 \quad (2.52)$$

$$\Delta\theta_t = -\frac{RMS[\Delta\theta]_t}{RMS[g]_t} g_t \quad (2.53)$$

$$\Delta\theta_t = -\frac{RMS[\Delta\theta]_{t-1}}{RMS[g]_t} g_t \quad (2.54)$$

$$\alpha_t = \frac{\alpha}{RMS[g]_t} \quad (2.55)$$

Keterangan :

E : *running average*

g : *gradient*

γ : Konstanta yang melimit perubahan *gradient*

α : *learning rate*

θ : posisi

G : *diagonal matrix*

RMS : *Root Mean Square*

2.1.8.5 Adam

Adam adalah singkatan dari *Adaptive Moment Estimation*, merupakan perpaduan antara metode momentum dan *RMSprop*. Pada dasarnya Adam mengambil keuntungan dari metode momentum yaitu *smoothening* dan dari metode *RMSprop* adalah *learning rate decay*. Seperti yang dapat kita lihat pada persamaan 2.57 dan 2.57, menunjukan persamaan *velocity* untuk *smoothening* baik untuk bobot dan juga *bias*. Demikian juga persamaan 2.58 dan 2.59 merupakan persamaan *learning rate decay*. Adam memadukan kedua persamaan ini untuk meng-*update* nilai bobot dan *bias*-nya seperti pada persamaan 2.60 dan 2.61. Setiap iterasinya *learnable parameter* nilainya akan ter-*update*, mengakibatkan nilai *velocity* dan *learning rate decay* berubah seperti pada persamaan 2.62, 2.63, 2.65, 2.64 [27].

$$V_{dw} = \beta_1 V_{dw} + (1 - \beta_1) \frac{\partial L}{\partial w} \quad (2.56)$$

$$V_{db} = \beta_1 V_{db} + (1 - \beta_1) \frac{\partial L}{\partial b} \quad (2.57)$$

$$S_{dw} = \beta_2 S_{dw} + (1 - \beta_2) \left(\frac{\partial L}{\partial w} \right)^2 \quad (2.58)$$

$$S_{db} = \beta_2 S_{dw} + (1 - \beta_2) \left(\frac{\partial L}{\partial b} \right)^2 \quad (2.59)$$

$$W_t = W_{t-1} - \frac{\alpha * V_{dw}}{\sqrt{S_{dw}} + \epsilon} \quad (2.60)$$

$$b_t = b_{t-1} - \frac{\alpha * V_{db}}{\sqrt{S_{db}} + \epsilon} \quad (2.61)$$

$$V'_{dw} = \frac{V_{dw}}{1 - \beta_1^t} \quad (2.62)$$

$$V'_{db} = \frac{V_{db}}{1 - \beta_1^t} \quad (2.63)$$

$$S'_{dw} = \frac{S_{dw}}{1 - \beta_2^t} \quad (2.64)$$

$$S'_{db} = \frac{S_{db}}{1 - \beta_2^t} \quad (2.65)$$

Keterangan :

V_{dw}	: velocity on respect of w
V_{dw}'	: update of velocity on respect of w
V_{db}	: velocity on respect of b
V_{db}'	: update of velocity on respect of b
β_1	: learnable parameter 1
β_1'	: update of learnable parameter 1
β_2	: learnable parameter 2
β_2'	: update of learnable parameter 2
∂L	: partial derivative dari Loss function
S_{dw}	: perubahan posisi (mirip θ) terhadap bobot
S_{dw}'	: pertambahan dari perubahan posisi (mirip θ) terhadap bobot
S_{db}	: perubahan posisi (mirip θ) terhadap bias
S_{db}'	: pertambahan dari perubahan posisi (mirip θ) terhadap bias

2.1.9 *Image Preprocessing*

Image preprocessing adalah langkah-langkah yang diambil untuk *formatting* gambar sebelum mereka digunakan saat melatih model. *Image Augmentation* adalah manipulasi yang diaplikasikan pada gambar untuk menciptakan versi yang berbeda dari suatu konten yang sama. Tujuannya adalah memperkenalkan model terhadap contoh *training* yang lebih luas. Perbedaan antara *Image preprocessing* dengan *Image augmentation* adalah *preprocessing* digunakan pada *training* dan *testing* sementara *augmentation* hanya digunakan pada *training*. *Image preprocessing* diperlukan untuk membersihkan data gambar yang hendak dipakai saat melatih model, misalnya membuat gambar memiliki dimensi ukuran yang sama. Pada penelitian ini, *preprocessing* yang dipakai adalah *Modified Contrast Limited Adaptive Histogram Equalization*. Pada dasarnya *MCLAHE* adalah *CLAHE* yang dilanjut oleh *Gaussian blur*. Proses ini ditujukan untuk mengurangi *noise contrast* yang dihasilkan melalui proses *CLAHE* [1].

2.1.9.1 *Image Interpolation*

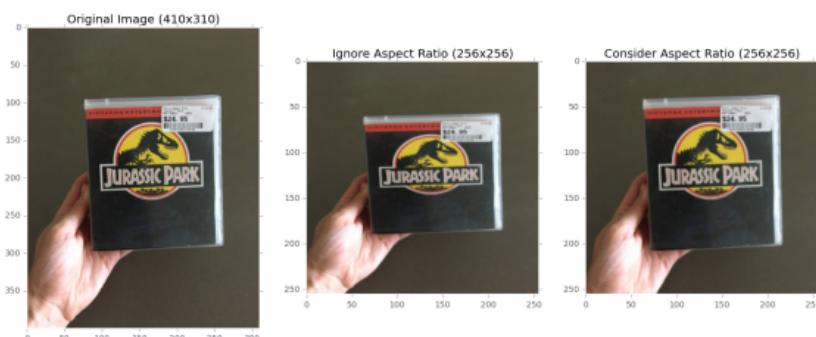
Interpolation adalah proses untuk mengestimasi nilai *pixel* pada lokasi yang tidak diketahui dengan menggunakan nilai *pixel* yang sudah diketahui sebelumnya [39]. Salah satu metode untuk interpolasi adalah *nearest neighbor interpolation* [39], pada metode ini asumsikan hendak memperbesar gambar yang semula berukuran 500×500 menjadi 750×750 . Asumsikan terdapat *imaginary image* dengan dimensi 750×750 kemudian gambar tersebut diperkecil hingga ukuran 500×500 ,

tentunya *pixel* penyusun *imaginary image* ini akan lebih padat dibanding dengan gambar semula. Langkah berikutnya adalah meletakan *imaginary image* di atas gambar semula, nilai terdekat dari *original pixel* terhadap *pixel* dari *imaginary image* itu dijadikan nilai *pixel*-nya.

Terdapat banyak metode interpolasi, pada penelitian kali ini yang dipakai adalah metode interpolasi *nearest neighbor* dari pustaka keras. Metode ini menduplikasi *pixel* terdekat. Asumsikan *pixel* awal dari gambar adalah *abcd*. Setelah proses *zoom* selesai, maka *pixel* akhir dari gambar menjadi *aaabcccc*.

2.1.9.2 Aspect Aware Preprocessing

Aspect ratio adalah hubungan proporsional antara tinggi dan lebar pada gambar. Ada beberapa kasus dalam pembangunan model *CNN* yang tidak mempedulikan *aspect ratio* tetapi tetap mendapatkan nilai akurasi yang tinggi. Tetapi untuk kasus *dataset* yang menantang lebih baik memperhatikan *aspect ratio* saat *resizing* gambar. Pada gambar 2.39 dapat dilihat perbedaan antara teknik *resize* yang memperhatikan *aspect ratio* dan yang tidak.



Gambar 2.39 Aspect Aware Preprocess [21]

Algoritma 2.1 menjelaskan langkah-langkah untuk melakukan *resizing* dengan memperhatikan *aspect ratio*.

ALGORITME 2.1 algoritma Aspect Aware Preprocessing

- 1: Mencari dimensi yang lebih besar, apakah *width* atau *height*
- 2: Apabila dimensi *height* lebih besar, maka *resize width* gambar lalu hitung *delta height* dengan cara membagi dua hasil pengurangan antara *current height* dengan *target height*. Apabila dimensi *width* lebih besar, maka *resize height* gambar lalu hitung *delta width* dengan cara membagi dua hasil pengurangan antara *current width* dengan *target width*.

3: *Crop* gambar dengan formula

$$i = i[dH : h - dH, dW : w - dW] \quad (2.66)$$

Keterangan :

i : gambar

dH : *delta height*

h : *current height*

dW : *delta weight*

w : *weight*

4: *Resize* hasil gambar yang sudah di-*crop* dengan menggunakan metode *cv2.resize*

Pada langkah ke-lima kita perlu *re-grab* lebar dan tinggi gambar lalu menggunakan *deltas* untuk memotong pusat gambar [21]. Hasil gambar setelah dipotong mungkin akan memiliki *pixel* yang berlebih dari target yang sudah ditentukan sebelumnya karena masalah pembulatan, karenanya dilakukan *resize* menggunakan metode dari pustakan *opencv* [21].

2.1.9.3 *Image Augmentation*

Image augmentation adalah teknik untuk mengaplikasikan transformasi pada gambar yang menghasilkan salinan gambar-gambar baru. Dimana gambar-gambar baru tersebut memiliki perbedaan pada aspek tertentu dari gambar asalnya tergantung dari teknik *augmentasi* yang dipakai. Mengaplikasikan variasi pada gambar tidak mengubah kelas dari objek tersebut, hanya menghasilkan sudut pandang baru terhadap objek. Teknik ini tidak hanya memperbesar ukuran *dataset*, tetapi juga terlibat dalam menambah variasi dari *dataset* yang mengakibatkan model tergeneralisasi lebih baik. Terdapat beberapa teknik *augmentasi* diantaranya adalah *rotation*, *shifts*, *flips*, *brightness* dan *zoom* [36]. Transformasi dari koordinat-koordinat gambar melalui proses *augmentation* atau bisa disebut juga *affine transformation* dapat dilihat pada persamaan berikut 2.67 [39].

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = T \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} t_{11} & t_{12} \\ t_{21} & t_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Keterangan :

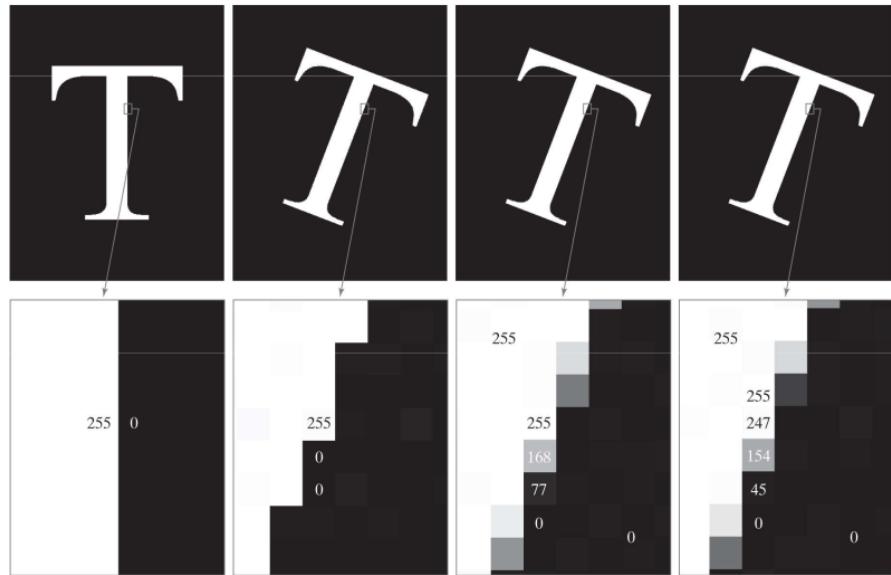
- x' : new x coordinate
 - y' : new y coordinate
 - x : x coordinate
 - y : y coordinate
 - T : affine matrix
-

1. Rotation

Merotasi gambar pada rentang 0 hingga 360 derajat. Ketika gambar dirotasi, beberapa *pixel* pada gambar akan bergeser keluar *frame* gambar sehingga posisi awal *pixel* menjadi kosong. *Pixel* yang kosong ini dapat diisi dengan teknik interpolasi, terdapat beberapa metode yang dapat digunakan. Penjelasan mengenai teknik interpolasi dapat dibaca pada subbab 2.1.9.1 Pada penelitian kali ini digunakan metode dari pustaka keras yang menggunakan teknik interpolasi tersendiri, penjelasannya dapat dibaca pada subbab 2.1.11.

$$\begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (2.67)$$

Persamaan *affine transformation* yang dipakai saat hendak merotasi gambar dapat dilihat pada persamaan 2.67. *Pixel* pada gambar nantinya akan dioperasikan dengan *matrix affine transformation* di atas sehingga menghasilkan gambar baru yang sudah mengalami rotasi.



Gambar 2.40 *rotation process* [39]



Gambar 2.41 *rotation process* [36]

2. Shift

Proses menggeser *pixel* pada gambar, bisa berlangsung secara *horizontal* maupun *vertical*. Proses ini berlangsung dengan menambahkan nilai konstan kepada semua *pixel*. *Matrix affine transformation* untuk proses ini adalah pada persamaan 2.68 [39].

$$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \quad (2.68)$$

$$x' = x + x_t \quad (2.69)$$

$$y' = y + y_t \quad (2.70)$$

Dengan mengoperasikan perkalian vektor antara koordinat mula-mula dengan *affine matrix* maka dapat disederhanakan persamaan *shifting* atau translasi menjadi seperti pada persaman 2.69 dan 2.70 [39].



Gambar 2.42 *shift process* [36]

3. Flips

Proses *flipping* gambar baik secara sumbu x maupun sumbu y. *affine matrix* ketika hendak *mirroring* terhadap sumbu x adalah pada persamaan 2.71. Ketika hendak *mirroring* terhadap sumbu y maka menggunakan persamaan 2.72 [39].

$$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad (2.71)$$

$$\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \quad (2.72)$$



Gambar 2.43 *flip process* [36]

4. Brightness

Merubah tingkat pencahayaan pada gambar. *Brightness* pada gambar dapat diatur dengan menambah atau mengurangi *pixel* penyusun gambar tersebut. Terdapat beberapa metode untuk mengatur intensitas *pixel* pada gambar, namun pada penelitian kali ini digunakan metode *MCLAHE*. Untuk lebih jelasnya dapat dibaca pada subbab 2.1.9.8



Gambar 2.44 *brightness process* [36]

5. Zoom

Melakukan *zoom in* atau *zoom out* pada gambar. Proses *zoom* sendiri terdapat 3 metode yaitu :



Gambar 2.45 *zoom process* [36]

(a) *Pixel replication (nearest neighbor interpolation)*

Metode ini hanya mereplikasi *pixel* tetangga sebanyak n baik berdasarkan kolomnya maupun barisnya.

misalkan terdapat gambar berdimensi 2×2 seperti pada gambar 2.46

1	2
3	4

Gambar 2.46 *Original Image*

Proses *zoom* pada baris adalah menyalin *pixel* ke kolom sampingnya.

2.46

1	1	2	2
3	3	4	4

Gambar 2.47 *Row wised zoom*

Langkah selanjutnya adalah menyalin *pixel* ke baris selanjutnya.

1	1	2	2
1	1	2	2
3	3	4	4
3	3	4	4

Gambar 2.48 *Column wised zoom*

Kelebihan dari metode ini adalah prosesnya yang sederhana sehingga biaya komputasinya rendah. Kelemahan dari metode ini adalah hasil keluarannya *blurry*.

(b) *Zero order hold method*

Adalah metode *zooming*, dikenal juga dengan istilah *zoom twice* karena hanya bisa melakukan proses *zooming* dua kali. Proses ini mengambil dua baris *pixel* yang berdampingan, menjumlahkan mereka lalu membagi hasilnya dengan dua, lalu hasil akhirnya dibulatkan ke bawah. Hasil akhir diletakan di antara dua elemen tersebut. Pertama proses dilakukan pada kolom selanjutnya pada baris. Proses perhitungan nilai dapat dilihat pada gambar 2.50 dan gambar 2.51. Pada gambar 2.50 dapat dilihat bahwa hasil penjumlahan $1 + 2$ adalah 3 dan pembagian 3 dengan 2 adalah 1.5. Pembulatannya ke bawah adalah 1, maka nilai yang diletakan di antara 1 dan

2 adalah nilai 1.

1		2
3		4

Gambar 2.49 Original Image

1	1	2
3	3	4

Gambar 2.50 Row wised zoom

1	1	2
2	2	3
3	3	4

Gambar 2.51 Column wised zoom

Kelebihan dari metode ini adalah hasil keluarannya tidak *blurry*, tetapi hanya bisa melakukan *zoom* dengan *power of 2*.

(c) K-Times Zooming

ALGORITME 2.2 K-times zooming algorithm

- 1: Menentukan nilai K (*zooming factor*).
- 2: Mengambil dua buah *pixel* yang bersebelahan, lalu mengurangi nilai *pixel* yang bernilai lebih besar sejumlah *pixel* yang bernilai kecil. Hasil keluaran ini dinamakan *OP*.
- 3: Membagi nilai *OP* dengan *K*. Lalu hasil pembagian ini ditambahkan kepada *pixel* yang nilainya lebih kecil. Lalu hasilnya diletakan di antara kedua *pixel* tersebut.
- 4: Menambahkan nilai *OP* kepada *pixel* hasil perhitungan proses nomor 3 dan hasilnya diletakan di samping *pixel* tersebut. Proses ini dilakukan berulang sejumlah *K-1*.
- 5: Setelah proses tersebut selesai, urutkan nilai *pixel* antara secara *ascending* apabila *pixel* pada sebelah kiri lebih kecil dan *descending* apabila *pixel* sebelah kiri lebih besar.
- 6: Ulangi langkah berikut untuk baris dan kolom.

15		30		15
30		15		30

Gambar 2.52 Column wised zoom

15	20	25	30	20	25	15
30	20	25	15	20	25	30

Gambar 2.53 K-times zoom

15	20	25	30	25	20	15
30	25	20	15	20	25	30

Gambar 2.54 *K-times zoom*

15	20	25	30	25	20	15
20	21	21	25	21	21	20
25	22	22	20	22	22	25
30	25	20	15	20	25	30

Gambar 2.55 *Column wised zoom*

Pada gambar di atas dapat dilihat proses *zooming* dengan metode *k-times* dengan $K = 3$. Dapat dilihat pada gambar awal yaitu gambar 2.52, *pixel* awal adalah 15 dan 30, maka *OP*-nya bernilai 15, hasil pembagian dengan K adalah 5. Karena $K = 3$, maka proses perhitungan akan dilakukan sebanyak $K - 1 = 2$. Pada gambar 2.53 dapat dilihat bahwa hasil *pixel* yang telah diproses diletakan di antara dua *pixel* awal. Pada gambar 2.54, dapat dilihat hasil *pixel* tersebut disusun secara *ascending / descending*. Pemrosesan pada kolom juga dilakukan dengan hal yang sama. Hasil akhir dari proses ini dapat dilihat pada gambar 2.55.

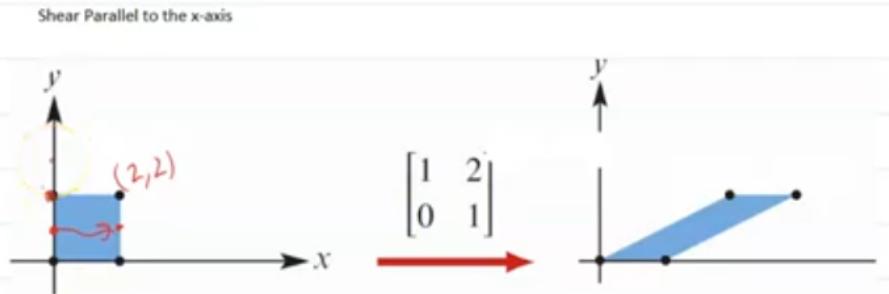
Kelebihan dari metode ini adalah hasilnya tidak *blurry* dan proses *zoom* tidak terbatas pada *power of 2*. Kelemahan dari metode ini adalah perlunya proses sortir yang membutuhkan daya komputasi lebih tinggi.

6. Shearing

Shifting setiap point dengan nilai yang proporsional terhadap jaraknya terhadap sumbu x dan sumbu y. Proses *shear* dapat berlangsung terhadap sumbu x (persamaan 2.73) maupun sumbu y (persamaan 2.74) [38].

$$\begin{bmatrix} 1 & \alpha \\ 0 & 1 \end{bmatrix} \quad (2.73)$$

$$\begin{bmatrix} 1 & 0 \\ \beta & 1 \end{bmatrix} \quad (2.74)$$



Gambar 2.56 Shear x example

$$n = \begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 2 \end{bmatrix}$$

$$n = \begin{bmatrix} 4 \\ 2 \end{bmatrix}$$

Keterangan :

n : new point

Gambar 2.56 adalah contoh proses *shearing* terhadap sumbu x dengan skala bernilai dua. Proses yang dilakukan adalah transformasi linear pada setiap koordinat *pixel* dengan matrix transformasi. Contoh perhitungan matrix dapat dilihat pada persamaan 2.75 [38].

2.1.9.4 LAB - colorspace

Lab color spaces memiliki tiga dimensi, yaitu:

1. *Lightness (L)*

mengandung intensitas yang terdapat pada gambar. Pada *RGB color spaces* terdapat tiga *channel*, masing-masing *channel* sudah ter-encoding data mengenai *brightness* gambar. Pada *LAB - colorspace* data *brightness* dipisahkan kedalam bentuk *channel* tersendiri yaitu *lightness channel*.

2. *a*

channel ini mengandung komponen warna dari hijau (#00FF00) hingga magenta (#FF00FF)

3. *b*

channel ini mengandung komponen warna dari biru (#0000FF) hingga kuning

(#FFFF00)

ALGORITME 2.3 Conversion between RGB to LAB channel

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.412453 & 0.357580 & 0.180423 \\ 0.212671 & 0.715160 & 0.072169 \\ 0.019334 & 0.119193 & 0.950227 \end{bmatrix} \cdot \begin{bmatrix} R \\ G \\ B \end{bmatrix} \quad (2.75)$$

$$L = 116 * Y^{\frac{1}{3}} - 16 \quad (2.76)$$

$$L = 903.3 * Y \quad (2.77)$$

$$a = 500(f(X) - f(Y)) + delta \quad (2.78)$$

$$b = 200(f(Y) - f(Z)) + delta \quad (2.79)$$

$$f(t) = t^{\frac{1}{3}} \quad (2.80)$$

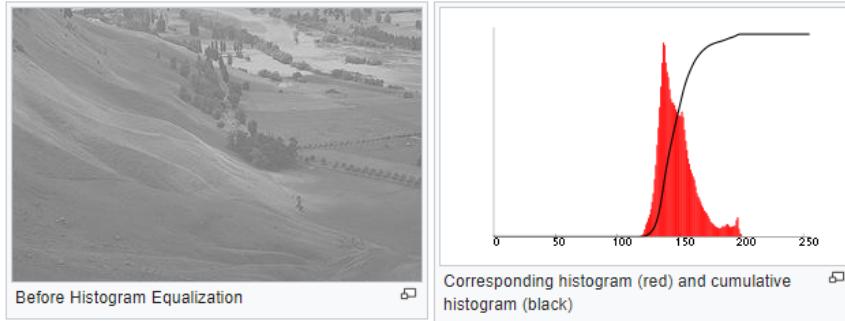
$$f(t) = 7.787t + \frac{16}{116} \quad (2.81)$$

- 1: mengonversi RGB ke dalam bentuk *floating point* pada rentang 0 hingga 1. Seperti pada persamaan 2.75
 - 2: Selanjutnya nilai X dibagi dengan nilai total keseluruhan X. Nilai Z pun demikian.
 - 3: Selanjutnya untuk perhitungan nilai *channel L*, apabila nilai Y > 0.008856 maka menggunakan persamaan 2.76. Bila tidak maka menggunakan persamaan 2.77.
 - 4: Selanjutnya untuk menentukan nilai pada *channel a* dan *b* digunakan persamaan 2.78 dan 2.79 untuk fungsi dengan ketentuan menggunakan persamaan 2.80 jika nilai masukan lebih besar dari 0.008856. Jika tidak demikian maka fungsi menggunakan persamaan 2.81. Nilai *delta* menggunakan nilai 128 jika gambar masukan adalah 8 bit. Jika bukan 8 bit maka nilainya 0.
 - 5: Apabila gambar merupakan gambar 8 bit maka nilai L perlu disesuaikan menjadi L * 255/100.
-

Algoritma 2.3 merupakan algoritma mengonversi gambar dari *RGB channel* menjadi *LAB channel*.

2.1.9.5 Histogram Equalization (HE)

Histogram adalah representasi grafis dari distribusi intensitas gambar atau dengan kata lain jumlah intensitas masing-masing *pixel* dalam gambar [37].



Gambar 2.57 Before Histogram Equalization [37]

Histogram equalization adalah metode tradisional untuk meningkatkan atau memperkuat *contrast* pada gambar digital, yang memodifikasi histogram distribusi *pixel* (*grayscale*) kedalam bentuk *uniform distribution*. Teknik ini bertujuan untuk merubah gambar agar memiliki distribusi *histogram* yang lebih merata.



Gambar 2.58 After Histogram Equalization [37]

ALGORITME 2.4 HE Algorithm [37]

- 1: Menghitung frekuensi masing-masing *pixel* pada gambar.
- 2: Menghitung *Probability Distribution Function (PDF)* untuk masing-masing *pixel*.

$$P_i = \frac{n_i}{N} \quad (2.82)$$

Keterangan :

P_i : Probability Distribution Function pixel ke i

n_i : frekuensi pixel ke i

N : total keseluruhan pixel

- 3: menghitung *Cumulative Distribution Function* dengan cara menghitung jumlah *PDF pixel* dengan total *PDF pixel* sebelumnya.
 - 4: Mengkalikan nilai *CDF* dengan total jenis *pixel* yang ada, misalkan pada gambar hanya terdapat 255 jenis *pixel*. Maka nilai *CDF* akan dikalikan dengan 255.
 - 5: Membulatkan hasil ke bilangan bulat terdekat dengan operasi matematika dasar. Bilangan bulat inilah *pixel* yang baru dari hasil *histogram equalization*.
-

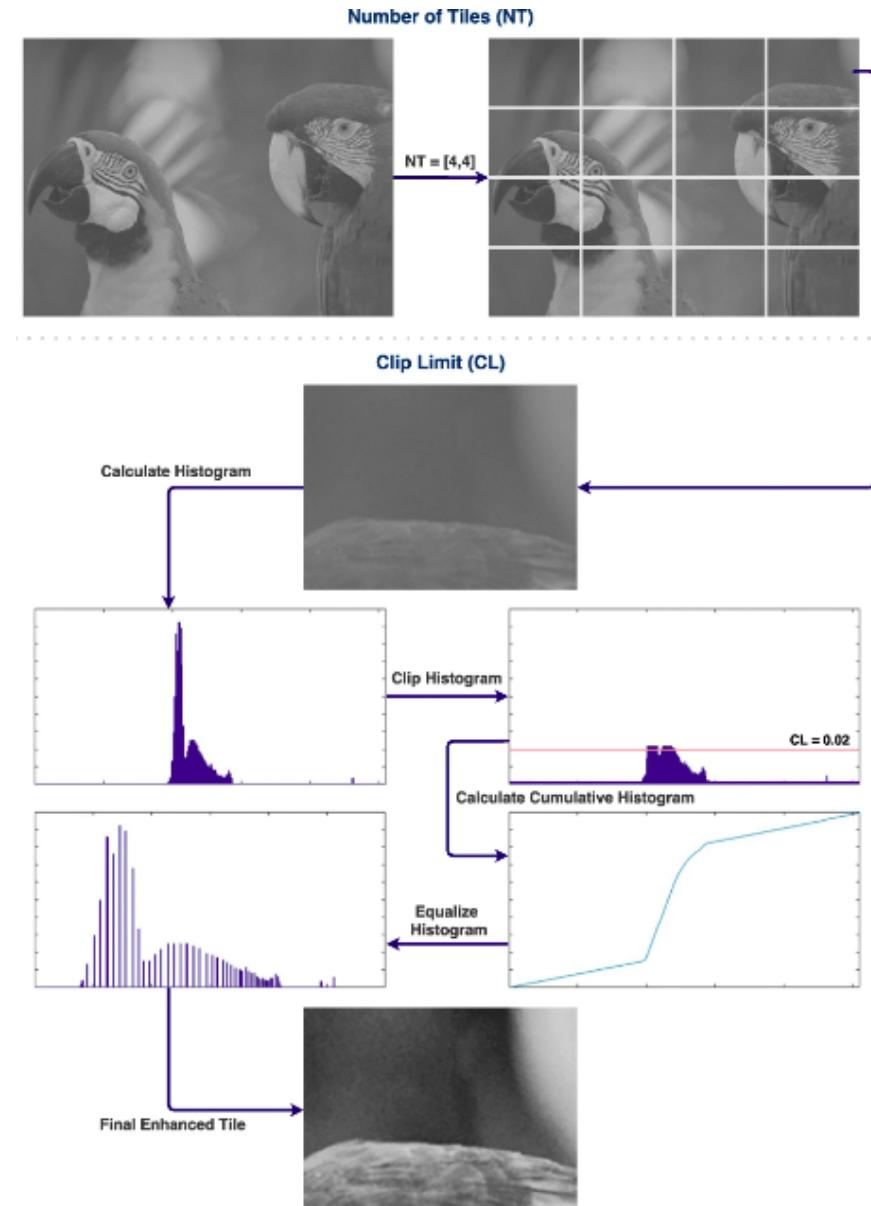
Tetapi metode *Histogram Equalization / HE* ini memiliki beberapa kelemahan, diantaranya adalah potensi hilangnya *detail* pada gambar dan beberapa daerah pada gambar menjadi lebih terang dibanding sebelumnya [1].

2.1.9.6 Adaptive Histogram Equalization (AHE)

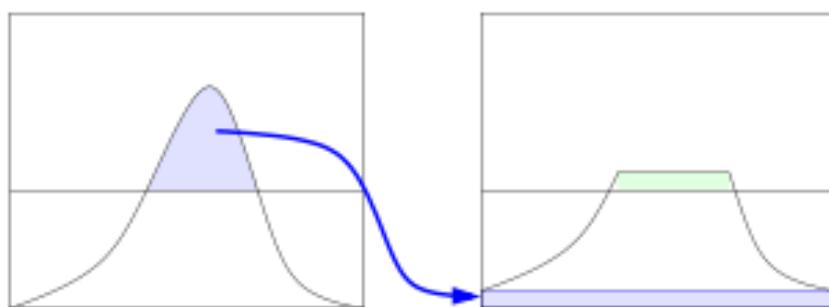
Metode yang dikembangkan untuk mengatasi kelemahan pada *Histogram Equalization* adalah *Adaptive Histogram Equalization*. *AHE* pada dasarnya mirip dengan proses konvolusi, terdapat filter berukuran $M \times N$ yang digeser keseluruh *field* pada gambar. Proses *Histogram Equalization* berlangsung pada *filter* tersebut. Tetapi *AHE* juga memiliki beberapa kelemahan diantaranya, daya komputasi yang tinggi dan amplifikasi *noise* pada daerah yang *homogen* [1].

2.1.9.7 Contrast Limited Adaptive Histogram Equalization (CLAHE)

Contrast Limited Adaptive Histogram Equalization (CLAHE) adalah perkembangan dari metode *AHE*. *AHE* cenderung *overamplify contrast* pada daerah *homogen*. *CLAHE* mengatasi masalah ini dengan membatasi amplifikasinya dengan metode *clip limit*. Algoritma dari *CLAHE* sendiri mirip dengan metode *AHE* yaitu dengan menggunakan *kernel* yang dilalui ke seluruh *field* gambar. Proses *histogram equalization* berlangsung hanya pada *kernel* tersebut. Namun perbedaannya adalah pada saat perhitungan *PDF*, apabila nilai *PDF* melebihi *clip limit* yang sudah ditentukan, maka nilai lebihnya akan ditampung dan pada akhir proses akan dibagi merata ke seluruh *pixel* yang terlibat. Dengan metode ini, histogram sebelum menghitung *Cumulative Distribution Function (CDF)*, melimitasi *contrast* yang berlebih sehingga *noise amplification* dapat teratas [1].



Gambar 2.59 Contrast Limited Adaptive Histogram Equalization (CLAHE) [1]



Gambar 2.60 Contrast Limited Adaptive Histogram Equalization (CLAHE) [1]

Seperti terlihat pada gambar 2.60, *clip limit* membatasi nilai distribusi frekuensi *Histogram Equalization* saat menghitung *Probability Distribution Function*

(PDF). Nilai lebihnya akan didistribusikan merata keseluruh nilai pada *filter* tertentu. Namun metode ini masih menghasilkan *noise* [1].

2.1.9.8 *Modified Contrast Limited Adaptive Histogram Equalization (MCLAHE)*

Pada dasarnya *Modified Contrast Limited Adaptive Histogram Equalization / MCLAHE* adalah *CLAHE* yang dilanjutkan oleh proses *Gaussian Blurring*. Hal ini dilakukan untuk menghindari masalah *noise* yang dihasilkan akibat proses *CLAHE* [1].

2.1.10 *Confusion Matrix*

Confusion matrix adalah sebuah metrik yang digunakan untuk mengevaluasi model klasifikasi secara akurat. *Confusion matrix* memvisualisasikan hasil kinerja suatu algoritme atau metode komputasi. Setiap baris matriks mewakili hasil kelas yang diprediksi, sementara setiap kolom mewakili hasil kelas dalam kondisi sebenarnya. *Confusion matrix* digunakan untuk pengujian yang memiliki dua kelas atau lebih [33]. Tabel 2.1 merupakan contoh *confusion matrix*.

Tabel 2.1 *Confusion matrix*

		Keadaan Data Sebenarnya	
		TRUE	FALSE
Hasil Prediksi	TRUE	TP (<i>True Positive</i>) disebut juga <i>correct result</i>	FP (<i>False Positive</i>) disebut juga <i>unexpected result/false alarm</i>
	FALSE	FN (<i>False Negative</i>) disebut juga <i>missing result</i>	TN (<i>True Negative</i>) disebut juga <i>correct rejection</i>

Confusion matrix memiliki empat komponen sebagai berikut [33].

1. TP (*True Positive*) merupakan kondisi di mana data bernilai positif pada keadaan sebenarnya dan bernilai positif juga pada hasil prediksi.
2. FP (*False Positive*) merupakan kondisi di mana data bernilai negatif pada keadaan sebenarnya, namun bernilai positif pada hasil prediksi.
3. TN (*True Negative*) merupakan kondisi di mana data bernilai negatif pada keadaan sebenarnya dan bernilai negatif juga hasil prediksi.
4. FN (*False Negative*) merupakan kondisi di mana data bernilai positif pada keadaan sebenarnya, namun bernilai negatif pada hasil prediksi.

Dengan menggunakan komponen yang telah disebutkan di atas, dapat dibuat persamaan untuk mengevaluasi model yang dibangun sebagai berikut [33].

1. *Accuracy/ACC*: Menghitung nilai prediksi yang benar.

$$ACC = \frac{TP + TN}{TP + FP + TN + FN} \quad (2.83)$$

2. *True Positive Rate/TPR (Recall/Sensitivity)*: Menghitung nilai prediksi positif pada saat data tersebut dalam keadaan sebenarnya adalah positif.

$$TPR = \frac{TP}{TP + FN} \quad (2.84)$$

3. *Precision*: Menghitung nilai prediksi positif yang benar.

$$Precision = \frac{TP}{TP + FP} \quad (2.85)$$

4. *F1 Score*: representasi *harmonic mean* dari perbandingan *precision P* dan *recall R*. Nilainya memiliki rentang 0-1. Nilai 1 menunjukkan *classifier* yang dibangun sudah sempurna.

$$F = \frac{2 \times P \times R}{P + R} \quad (2.86)$$

2.1.11 Pustaka

Pada bagian ini akan dijelaskan *library* yang digunakan. *Library* yang digunakan adalah *tensorflow*, *keras*, *opencv*, *numpy*, *os*, *matplotlib*, *imutils* dan *sklearn*. Untuk lebih jelasnya akan dibahas pada subbab berikut ini.

2.1.11.1 TensorFlow

TensorFlow adalah pustaka yang digunakan untuk pembelajaran mesin. TensorFlow memiliki sumber daya komprehensif dan fleksibel serta komunitas yang memungkinkan para peneliti melakukan penelitian mutakhir dalam bidang pembelajaran mesin dan dapat membangun aplikasi yang didukung pembelajaran mesin. Tabel 2.2 akan memberikan penjelasan terkait *method* yang digunakan dari pustaka ini.

Tabel 2.2 Daftar *method* yang digunakan dari pustaka TensorFlow

No.	Method	Masukan	Keluaran	Keterangan
1	keras. optimizers.SGD	learning_rate: float momentum: float nesterov: boolean	keras. optimizers	<i>Optimizer</i> yang mengimplementasikan algoritma <i>Stochastic Gradient Descent</i> apabila nesterov bernilai <i>false</i> dan algoritma <i>Nesterov Accelerated Gradient</i> ketika nesterov bernilai <i>true</i> . Ketika nilai <i>momentum</i> bernilai 0 maka <i>update rule</i> nya adalah seperti pada persamaan 2.87, sementara saat <i>momentum</i> bernilai lebih dari 0 maka <i>update rule</i> seperti pada persamaan 2.88 dan 2.89. Sementara apabila <i>nesterov</i> berinal <i>true</i> maka <i>update rule</i> akan mengikuti persaman 2.90 dan 2.91.

Tabel 2.2 Daftar *method* yang digunakan dari pustaka TensorFlow

No.	Method	Masukan	Keluaran	Keterangan
2	keras.optimizers.Adam	learning_rate: float	keras.optimizer	<i>Optimizer</i> yang mengimplementasikan algoritma Adam. parameter <i>learning rate</i> digunakan sebagai inisiasi <i>learning rate</i> awal dari algoritma. Pada penelitian ini, hanya menguji nilai <i>learning rate</i> , karenanya nilai β_1, β_2 dan <i>epsilon</i> mengikuti settingan <i>default</i> yaitu 0.9, 0.999 dan 0.0000001. Untuk formula perhitungan dapat melihat persamaan 2.60 dan 2.61 .
3	keras.optimizers.Adagrad	learning_rate: float initial_accumulator_value: float momentum: float epsilon: float name: string	keras.optimizer	<i>Optimizer</i> yang mengimplementasikan algoritma Adagrad. Persamaan matematikanya dapat dilihat pada persamaan 2.47 . <i>initial accumulator value</i> adalah nilai awal untuk variable G pada persamaan 2.47 .
4	keras.optimizers.Adadelta	learning_rate: float rho: float epsilon: float name: string	keras.optimizer	<i>Optimizer</i> yang mengimplementasikan algoritma Adadelta. Persamaan matematikanya dapat dilihat pada persamaan 2.54 .

Tabel 2.2 Daftar *method* yang digunakan dari pustaka TensorFlow

No.	Method	Masukan	Keluaran	Keterangan
5	keras.initializers.HeNormal	seed: Integer	keras initializer	mengaplikasikan initializer pada model <i>neural network</i> . Seed adalah <i>seeder</i> untuk nilai randomnya.
6	keras.regularizers.l2	-	keras regularizer	mengaplikasikan algoritma regularisasi l2 pada model <i>neural network</i>

2.1.11.2 Keras

Keras adalah pustaka yang menyediakan antarmuka Python untuk *artificial neural network*. Keras bertindak sebagai antarmuka untuk pustaka TensorFlow. Tabel 2.3 akan memberikan penjelasan terkait *method* yang digunakan dari pustaka ini.

Tabel 2.3 Daftar *method* yang digunakan dari pustaka Keras

No.	Method	Masukan	Keluaran	Keterangan
1	backend.K.image_data_format	-	status channel pada gambar, <i>channel_first</i> atau <i>channel_last</i>	Mengembalikan nilai status channel.

BAB 2 LANDASAN TEORI

Tabel 2.3 Daftar *method* yang digunakan dari pustaka Keras

No.	Method	Masukan	Keluaran	Keterangan
2	Conv2D	filters: int kernel_size: int kernel_initializer: string bias_initializer: string padding: string activation: string input_shape: tuple kernel_regularizer: string	tensor	Melakukan operasi konvolusi dua dimensi, <i>stride</i> ke arah samping kanan.
3	Batch Normalization	tensor	tensor	Lapisan yang menormalkan masukannya. <i>Batch normalization</i> menerapkan transformasi yang mempertahankan rata-rata keluaran mendekati 0 dan deviasi standar keluaran mendekati 1.
4	MaxPooling2D	pool_size: int	tensor	Operasi <i>max pooling</i> untuk data temporal 2D. Menurunkan sampel representasi masukan dengan mengambil nilai maksimum di atas <i>window</i> yang ditentukan oleh pool_size. <i>Window</i> digeser dengan <i>stride</i> .

Tabel 2.3 Daftar *method* yang digunakan dari pustaka Keras

No.	Method	Masukan	Keluaran	Keterangan
5	AveragePooling2D	pool_size: int	tensor	Operasi <i>average pooling</i> untuk data temporal 2D. Menurunkan sampel representasi masukan dengan mengambil nilai rata rata di atas <i>window</i> yang ditentukan oleh pool_size. <i>Window</i> digeser dengan <i>stride</i> .
6	regularizers.l2	l2: float	tensor	Regulator yang menerapkan penalti regularisasi L2.
7	Dropout	rate: float	tensor	Menetapkan unit masukan ke nilai 0 secara acak dengan frekuensi <i>rate</i> di setiap langkah selama waktu pelatihan, yang membantu mencegah <i>overfitting</i> .
8	Dense	units: int activation: string kernel_regularizer: string	tensor	Membuat <i>fully connected layer</i> sebagai lapisan keluaran.
10	summary	-	-	Menampilkan ringkasan model jaringan yang dibuat.
11	compile	optimizer: string loss: string metrics: list	-	Mengkonfigurasi model dengan nilai <i>loss</i> dan metrik pengukuran lainnya untuk proses pelatihan.

Tabel 2.3 Daftar *method* yang digunakan dari pustaka Keras

No.	Method	Masukan	Keluaran	Keterangan
12	<code>fit</code>	X_train: array y_train: array batch_size: int epochs: int validation_split: float callbacks: tuple	History object	Melatih model untuk sejumlah <i>epoch</i> (iterasi pada <i>dataset</i>).
13	<code>evaluate</code>	X_test: array y_test: array	Scalar test loss	Mengembalikan nilai <i>loss</i> dan nilai metrik untuk model dalam mode pengujian. Perhitungan dilakukan per <i>batch</i> .
14	<code>predict</code>	X_test: array	NumPy array	Menghasilkan prediksi keluaran pada sekumpulan data baru.
15	<code>to_categorical</code>	y: array num_classes: int	int [][]	Mengonversi vektor kelas (bilangan bulat) menjadi matriks kelas biner.
16	<code>save</code>	filepath: string include_optimizer: boolean	-	Menyimpan arsitektur model, bobot, dan konfigurasi pelatihan dalam satu berkas atau <i>folder</i> .
17	<code>load_model</code>	filepath: string custom_objects: dict	Keras model	Memuat model yang disimpan menggunakan <i>method save</i> .
18	<code>Sequential</code>	-	Container model	Menghasilkan Container yang dapat menampung <i>linear stack layer</i> dari keras Model.

Tabel 2.3 Daftar *method* yang digunakan dari pustaka Keras

No.	Method	Masukan	Keluaran	Keterangan
19	ModelCheckpoint	filepath : string monitor: string verbose: integer save_best_only: boolean save_weights_only: boolean mode: string save_freq: string	Keras callback	Memungkinkan <i>model.fit()</i> method untuk save model atau <i>weight</i> terbaik ketika <i>training</i>
20	Activation	Keras Activation Function atau string	Keras Activation Function	Menentukan jenis fungsi aktivasi yang dipakai oleh suatu <i>layer</i> .
21	preprocess.image.img_to_array	img: PIL image instance	Numpy array	Mengkonversi PIL image menjadi Numpy array.
22	utils.vis_utils.plot_model	model: Keras model to_file: string show_shapes: boolean show_dtype: boolean show_layer_names: boolean rankdir: string expand_nested: boolean dpi: integer	dot formatted image	Mengkonversi keras model ke dalam bentuk gambar berformat dot
23	Flatten	tensor	reshaped tensor	Mengubah input tensor kedalam bentuk <i>flat</i> (array 1 dimensi).

Tabel 2.3 Daftar *method* yang digunakan dari pustaka Keras

No.	Method	Masukan	Keluaran	Keterangan
24	Image. ImageData Generator	featurewise_center: boolean samplewise_center: boolean featurewise_std_ normalization: boolean samplewise_std_ normalization: boolean zca_whitening: boolean zca_epsilon: float rotation_range: integer width_shift_range: float height_shift_range: float shear_range: float zoom_range: float channel_shift_range: float fill_mode: string cval: float horizontal_flip: boolean vertical_flip: boolean validation_split: float	batches of tensor images.	Menghasilkan sekumpulan gambar tensor yang sudah di augmentasi. Peraturan augmentasi ditentukan lewat parameter seperti <i>horizontal_flip</i> , <i>width_shift_range</i> , <i>zoom_range</i> , etc. Pemrosesan <i>zoom</i> menggunakan nilai <i>fill_mode = nearest</i> . Ini artinya apabila <i>pixel</i> pada gambar bernilai abcd, maka nilai <i>pixel</i> setelah <i>zoom</i> adalah aaaaaaabccccccccc.

Tabel 2.3 Daftar *method* yang digunakan dari pustaka Keras

No.	Method	Masukan	Keluaran	Keterangan
25	Input	shape: integer batch_size: integer name: string dtype: float sparse: boolean	tensor	Menginisiasi Tensor Keras, sama seperti <i>Sequential</i> hanya saja pada Input kita bisa mendesain model secara parallel, tidak hanya 1 arah.
26	Concatenate	axis: integer	tensor	menggabungkan beberapa list input tensor.

2.1.11.3 OpenCV

Open Source Computer Vision Library adalah salah satu *open source library* pada bidang *computer vision* dan *machine learning*. OpenCV memiliki lebih dari 2500 algoritma yang telah dioptimasi. OpenCV dapat digunakan secara gratis untuk kepentingan akademik atau penggunaan secara komersial. Daftar metode dari *OpenCV* yang digunakan dalam penelitian ini dapat dilihat pada tabel 2.4

Tabel 2.4 Daftar *method* yang digunakan dari pustaka OpenCV

No.	Method	Masukan	Keluaran	Keterangan
1	resize	src :numpy array dszie :tuple integer interpolation: string	numpy array	mengubah dimensi dari gambar, bisa saja hanya lebarnya, tingginya, atau keduanya. Bisa juga mempertahankan aspek rasio dari gambar.
2	imread	path :string flag :string	numpy array	membaca data gambar dari suatu <i>directory file</i> kedalam numpy array dan disimpan dalam variable.

Tabel 2.4 Daftar *method* yang digunakan dari pustaka OpenCV

No.	Method	Masukan	Keluaran	Keterangan
3	imwrite	filename :string image :numpy array	boolean	saving image kedalam file di path yang sudah kita tetapkan, akan mengembalikan boolean true jika proses sukses.
4	cvtColor	src :string code :integer	numpy array	Mengkonversi warna dari suatu gambar kedalam bentuk format warna yang lain, misalnya <i>grayscale</i> , <i>HSV</i> , etc.
5	gaussianBlur	src :string ksize :tuple of integer borderType: integer	numpy array	Mengaplikasikan algoritma gaussian blur pada gambar, dengan melewati kernel berdimensi ksize pada gambar src.
6	equalizeHist	img :numpy array	numpy array	Mengaplikasikan algoritma <i>Histogram Equalization</i> kepada gambar img.
7	createCLAHE	clipLimit: float tileGridSize: tuple of integer	opencv clahe object	Menghasilkan instance opencv clahe untuk mengaplikasikan algoritma <i>Contrast Limited Adaptive Histogram Equalization</i> kepada gambar img, dengan menggunakan metode <i>apply</i> pada instance opencv clahe yang terbentuk.

2.1.11.4 NumPy

NumPy adalah pustaka untuk bahasa pemrograman Python yang memberikan dukungan untuk melakukan fungsi dan operasi matematika tingkat tinggi. NumPy

BAB 2 LANDASAN TEORI

membantu pemrosesan pengenalan wajah. Metode metode Numpy yang dipakai pada penelitian ini dapat dilihat pada table 2.5.

Tabel 2.5 Daftar *method* yang digunakan dari pustaka NumPy

No.	Method	Masukan	Keluaran	Keterangan
1	array	object :array object	numpy array	mengkonversi array object menjadi numpy array.
2	unique	object :array object	numpy array	mengembalikan nilai nilai yang unik yang terdapat pada object array

2.1.11.5 OS

Pustaka OS memberikan kita akses kepada beragam metode / fungsi yang berbasiskan sistem, banyak diantaranya yang berguna untuk manipulasi *file* dan *directory*. Metode metode OS yang dipakai pada penelitian ini dapat dilihat pada table 2.6.

Tabel 2.6 Daftar *method* yang digunakan dari pustaka OS

No.	Method	Masukan	Keluaran	Keterangan
1	listdir	path :string	list of string	Mengembalikan daftar dari nama nama <i>file</i> ataupun <i>directory</i> yang terdapat pada <i>directory path</i> yang kita lewati sebagai parameter pada fungsi <i>listdir</i> .
2	mkdir	path :string	-	Menciptakan <i>directory</i> pada <i>path</i> yang dilewati sebagai parameter pada fungsi.

Tabel 2.6 Daftar *method* yang digunakan dari pustaka OS

No.	Method	Masukan	Keluaran	Keterangan
3	path.sep	path :string	list of string	mirip dengan fungsi <i>split</i> , memisahkan <i>string path</i> yang dilewati sebagai parameter pada fungsi berdasarkan <i>path separator</i> yaitu /
4	path.exist	path : string	boolean	Memverifikasi bahwa <i>string path</i> yang dilewati pada parameter fungsi benar benar <i>exist</i> . Mengembalikan <i>logic true</i> jika <i>exist</i> .

2.1.11.6 Matplotlib

Pustaka Matplotlib adalah pustaka komprehensif untuk membuat visualisasi statis, animatif dan interaktif pada Python. Metode metode matplotlib yang dipakai pada penelitian ini dapat dilihat pada table 2.7.

Tabel 2.7 Daftar *method* yang digunakan dari pustaka matplotlib

No.	Method	Masukan	Keluaran	Keterangan
1	pyplot.figure	-	-	Pemanggilan fungsi ini bersifat <i>optional</i> . Fungsi ini menandakan bahwa kita hendak membuat satu atau beberapa plot pada 1 figure.
2	pyplot.plot	xpoints: array of integer ypoints: array of integer fmt: string label: string	-	Plotting x terhadap y pada figure yang disediakan.
3	pyplot.title	title: string	-	Memberikan judul pada figure plot.

Tabel 2.7 Daftar *method* yang digunakan dari pustaka matplotlib

No.	Method	Masukan	Keluaran	Keterangan
4	pyplot.xlabel	label: string	-	Memberikan judul kepada <i>x-axis</i> pada figure.
5	pyplot.ylabel	label: string	-	Memberikan judul kepada <i>y-axis</i> pada figure.
6	pyplot.legend	-	-	Menampilkan informasi mengenai label plot kurva yang terdapat pada figure.
7	pyplot.show	-	-	Menampilkan figure yang sudah disetting sebelumnya.

2.1.11.7 Imutils

Pustaka imutils berisi kumpulan fungsi-fungsi praktis untuk melakukan pemrosesan dasar atau sederhana pada gambar seperti *translation*, *resize*, *skeletonize* dan *displaying*. Untuk dapat menggunakan pustaka ini, perlu menginstall *Numpy* dan *OpenCV* sebelumnya. Metode-metode imutils yang dipakai pada penelitian ini dapat dilihat pada tabel 2.8.

Tabel 2.8 Daftar *method* yang digunakan dari pustaka imutils

No.	Method	Masukan	Keluaran	Keterangan
1	resize	img : numpy array width: integer	numpy array	Memanipulasi dimensi dari gambar, tetapi bedanya dengan <i>cv2.resize</i> adalah metode ini mempertahankan <i>aspect ratio</i> .

Tabel 2.8 Daftar *method* yang digunakan dari pustaka imutils

No.	Method	Masukan	Keluaran	Keterangan
2	path.list_images	path : string	list of string	Mirip dengan fungsi <i>os.listdir</i> , fungsi ini bekerja secara rekursif, mendapatkan daftar <i>path images</i> dalam <i>string path</i> yang dilewati sebagai parameter pada fungsi ini.

2.11.8 Sklearn

Sklearn adalah singkatan dari *Sci-kit learn* merupakan *open source library* yang mendukung *supervised learning* dan *unsupervised learning*. Sklearn juga memiliki metode untuk melakukan *model fitting*, *data preprocessing*, *model selection*, *evalution etc.* Metode metode sklearn yang dipakai pada penelitian ini dapat dilihat pada table 2.9.

Tabel 2.9 Daftar *method* yang digunakan dari pustaka sklearn

No.	Method	Masukan	Keluaran	Keterangan
1	preprocessing.LabelBinarizer	neg_label: integer pos_label: integer sparse_output: boolean	Label Binarizer instance	Mengembalikan <i>instance of labelBinarizer</i> . Kita bisa mengaplikasikan algoritma <i>one hot encoding</i> dengan menggunakan method <i>fit_transform</i> pada array data.

Tabel 2.9 Daftar *method* yang digunakan dari pustaka sklearn

No.	Method	Masukan	Keluaran	Keterangan
2	model_selection. train_test_split	arrays: arrays test_size: float train_size: float random_state: integer stratify: reference shuffle: boolean	splitting list	Membagi data arrays dari parameter yang dilewati ke fungsi menjadi 2 daftar sesuai dengan porsi <i>test_size</i> dan <i>train_size</i> yang ditetapkan pada parameter fungsi. Parameter <i>stratify</i> memungkinkan pembagian data masing masing kelas proporstinya sama.
3	metrics. classification_ report	y_true: arrays y_pred: arrays target_names: arrays	print out text report	Menampilkan hasil test dari model yang sudah dilatih sebelumnya. Menampilkan informasi mengenai <i>precision</i> , <i>recall</i> , <i>f1-score</i> untuk masing masing datapoint serta <i>accuracy</i> rata rata dari seluruh data.

$$w = w - \alpha * g \quad (2.87)$$

$$v = m * v - \alpha \quad (2.88)$$

$$w = w + v \quad (2.89)$$

$$v = m * v - \alpha * g \quad (2.90)$$

$$w = w + m * v - \alpha * g \quad (2.91)$$

Keterangan :

w : bobot

v : *velocity*

α : *learning rate g* : *gradient*

m : *momentum*

2.2 Tinjauan Studi

Pada bagian ini, akan dijelaskan perbandingan dari berbagai penelitian terkait dengan tema penelitian ini. Perbandingan penelitian mengacu kepada jurnal referensi yang telah dikumpulkan dan dibaca oleh penulis.

2.2.1 State of the Art

BAB 2 LANDASAN TEORI

Tabel 2.10 State of the Art

No	Peneliti	Judul	Objektif	Hasil
1	Bendjilali Ridha Ilyas, Beladgham Mohammed, Merit Khaled and Taleb-Ahmed Abdelmalik (2020)	Illumination-robust face recognition based on deep convolutional neural networks architecture [1]	Melakukan evaluasi apakah penggunaan <i>Clip Limited Adaptive Histogram Equalization</i> dapat meningkatkan akurasi pengenalan wajah ?	Dihasilkan akurasi model 99.89% dengan menggunakan <i>MCLAHE</i> dan arsitektur <i>CNN</i> berupa <i>Inception / GoogleNet</i> .
2	Samar S. Mohammed, Wael A. Mohamed, A.T. Khalil and A.S. Mora (2019)	Deep Learning Face Detection and Recognition [2]	Bagaimana meningkatkan akurasi CNN dengan 15 layer dengan optimizer SGD terhadap pengenalan wajah?	Metode yang diajukan oleh jurnal ini menghasilkan akurasi 99.67% terhadap <i>face96 dataset</i> .
3	KB. Pranav, J. Manikandan (2020)	Design and Evaluation of a Real-Time Face Recognition System using Convolutional Neural Networks[3]	Bagaimana cara meningkatkan akurasi pengenalan wajah pada aplikasi real time menggunakan CNN?	Hasil dari tuning hyperparameter, maka dicapai maximum accuracy sebesar 98.75%
4	Coskun Musab,Ucar Aysegul, Yildirm Ozal and Demir Yakup (2017)	Face Recognition Based on Convolutional Neural Network [4]	Apakah dengan menambahkan Batch normalization, akurasi pada pengenalan wajah dapat meningkat?	dicapai akurasi tertinggi untuk arsitektur CNN sebesar 94.8% .

Tabel 2.10 State of the Art

No	Peneliti	Judul	Objektif	Hasil
5	Zhiming Xie, Junjie Li and Hui Shi (2019)	A Face Recognition Method Based on CNN [5]	Apakah perubahan jumlah neuron pada convolutional layer dapat berpengaruh pada nilai akurasi pengenalan wajah?	Dicapai akurasi tertinggi sebesar 97.5%

2.2.2 Pembahasan penelitian terkait

Terdapat beberapa metode yang dapat digunakan untuk mendeteksi wajah. Pada referensi [1], pertama-tama peneliti mengaplikasikan algoritma *Viola-Jones* untuk *cropping* gambar wajah. Selanjutnya menggunakan metode *Modified Contrast Limited Adaptive Histogram Equalization* untuk *preprocessing* gambar. Selanjutnya, gambar dikirimkan ke *CNN* sebagai data masukan, fitur di ekstraksi melalui *convolution layer*. Pada jurnal [1], dicoba beberapa arsitektur *CNN* diantaranya adalah *VGG-16*, *ResNet* dan *Inception*.

Pada jurnal [2], peneliti mencoba *custom architecture CNN* dengan 15 *layer* dan menggunakan *optimizer Stochastic Gradient Descent*. Pada jurnal ini, pertama gambar diproses menggunakan *Histogram Equalization*, setelah itu gambar dilewati ke *layer CNN* untuk diekstraksi fiturnya dan diklasifikasi.

Jurnal [3] membahas bagaimana meningkatkan akurasi pengenalan wajah pada *CNN* dengan mengubah ukuran *pooling window* dan jumlah *convolution layer*. Jurnal [4] berfokus kepada pengaruh penggunaan *batch normalization* pada pengenalan wajah. Arsitektur yang dipakai peneliti adalah arsitektur *CNN* sederhana hanya ditambahkan *batch normalization* diakhir setiap *block* konvolusi. Sementara jurnal [5] membahas tentang pengaruh perubahan jumlah *neuron* pada *convolution layer* terhadap peningkatan akurasi.

2.3 Tinjauan Objek

Pada bagian ini akan diulas mengenai objek-objek yang terkait dengan sistem pengenalan wajah. Data yang didapatkan merupakan *dataset* dari internet yaitu *Extended Yale Face Database B* dan *Komnet Dataset*. *Extended Yale Face Dataset* mengandung 16128 gambar dari 28 subject manusia dengan 9 pose dan 64 kondisi iluminasi. Tetapi pada penelitian ini, hanya menggunakan *cropped Yale dataset*

yaitu *Extended Yale Face Database* yang diaplikasikan *Viola-Jones* sehingga hanya bagian *frontal face* saja yang di-crop. *Cropped Yale Dataset* mengandung 2600 gambar dari 40 subject manusia dengan beragam kondisi iluminasi. *Komnet Dataset* terdiri dari 1000 gambar dari 50 subject sebagai data latih dan 200 gambar dari 50 subject sebagai data uji.

2.3.1 *Cropped Extended Yale Face Database*

Cropped Extended Yale Face Database terdiri dari 2600 gambar yang terbagi dalam 40 kelas. Citra wajah memiliki variasi pencahayaan, tetapi posenya sama yaitu wajah yang menghadap ke depan (*frontal face*). Keseluruhan gambar berukuran sama yaitu 168×192 pixel. Satu kelasnya terdiri dari 65 gambar dengan variasi iluminasinya seperti yang terlihat pada gambar 2.61.



Gambar 2.61 *Cropped Yale Face Database Illumination*

2.3.2 *Komnet Dataset*

Gambar wajah diambil dari media kamera *mobile phone*, kamera digital dan media sosial. *Dataset* ini dibentuk oleh laboratorium komputer politeknik Bali. Gambar gambar disimpan dalam format *.jpeg*, *.jpg* dan *.png*. Gambar wajah yang diambil seragam yaitu gambar wajah yang menghadap ke arah kamera (*frontal face*). Pada penelitian ini, yang dipakai hanyalah *dataset* wajah yang diambil melalui kamera digital. Contoh dataset dapat dilihat pada gambar 2.62.



Gambar 2.62 Komnet Dataset example

BAB 3 ANALISIS DAN PERANCANGAN SISTEM

Bab ini memaparkan analisis masalah yang akan diatasi. Analisis masalah dilakukan dengan menjelaskan pendekatan dan alur kerja dari perangkat lunak yang dikembangkan, implementasi dari metode yang akan digunakan, dan hasil yang akan ditampilkan sistem.

3.1 Analisis Masalah

Pada jurnal [1] dijelaskan bahwa *preprocessing* dengan metode *Modified Contrast Limited Adaptive Histogram Equalization* pada gambar wajah dengan penggunaan arsitektur *convolutional neural network* berupa *GoogleNet (Inception)*, mencapai tingkat akurasi tertinggi dengan nilai 99.89%. Pada penelitian ini, digunakan dua *benchmark dataset* yaitu, *Cropped Extended Yale Face Database* dan *Komnet Dataset*.

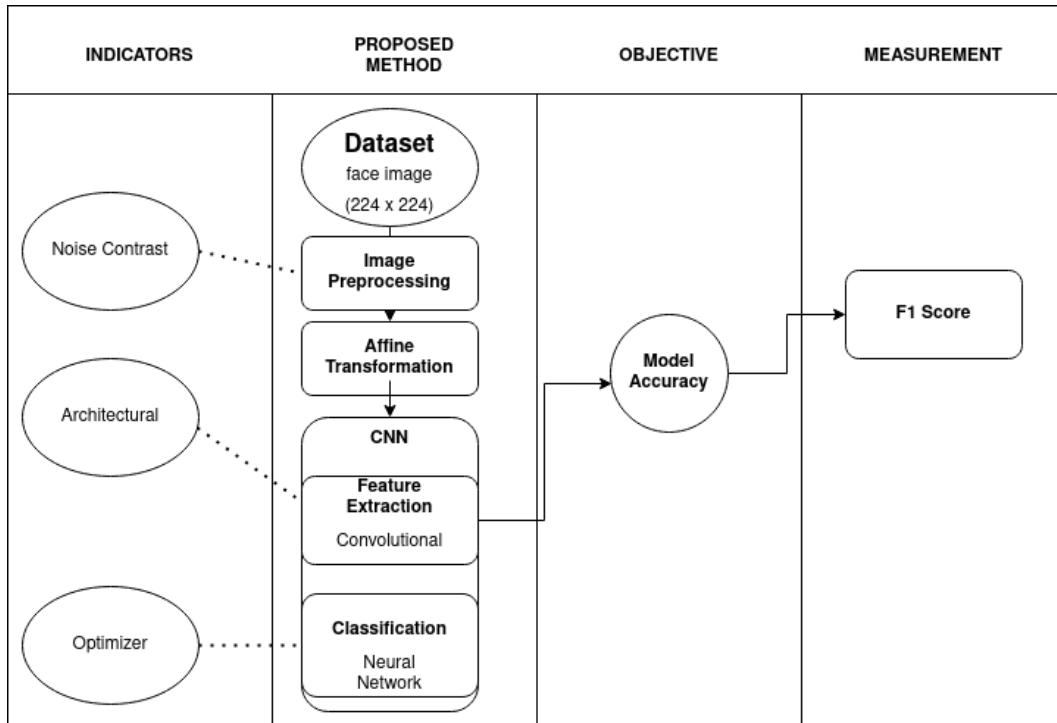
Cropped Extended Yale Face Database adalah *Yale Face Database* yang diproses menggunakan *Viola-Jones algorithm* sehingga diperoleh gambar wajah yang menghadap ke depan (*frontal face*). Sementara *Komnet Dataset* adalah *dataset* yang dibuat oleh laboratorium politeknik Bali, berisi data gambar *frontal face*.

Selain itu penulis juga menguji dua jenis arsitektur yaitu *Inception* dan *VGG16*, juga menguji lima jenis *optimizer* yaitu *Stochastic Gradient Descent*, *Nesterov Accelerated Gradient*, *Adagrad*, *Adadelta* dan *Adam*. Sebagai pembanding, disiapkan dua jenis *dataset*, original dan yang diproses oleh metode *MCLAHE*.

Data gambar selanjutnya akan dikirimkan ke dalam arsitektur *convolutional neural network*, lalu akan dihitung tingkat akurasinya.

3.2 Kerangka Pemikiran

Gambar 3.1 adalah kerangka pemikiran dari metode yang diusulkan untuk melakukan sistem pengenalan wajah (*face recognition*).



Gambar 3.1 Kerangka Pemikiran

Berdasarkan Gambar 3.1, penelitian ini dimulai dengan mempersiapkan *dataset* berupa gambar wajah. Gambar wajah nantinya akan diproses menggunakan algoritma *MCLAHE*. Sebagai pembanding, dipersiapkan juga *dataset* originalnya, nantinya dataset-dataset ini akan menjadi data latih untuk arsitektur *convolutional neural network* yang dibangun. Selain membandingkan jenis *dataset*, jenis *optimizer* yang dipakai juga turut diuji pada penelitian kali ini. Penelitian ini bertujuan untuk melihat hasil akurasi *convolutional neural network* dalam melakukan pengenalan wajah. Berikut ini akan dijelaskan setiap bagian pada gambar 3.1.

1. *Indicators* adalah variabel yang akan mempengaruhi hasil dari metode utama. Indikator dinilai pada saat pengujian *convolutional neural network* dimulai. Indikator yang diobservasi selama penelitian ini antara lain sebagai berikut.
 - (a) *MCLAHE* adalah singkatan dari *Modified Contrast Limited Adaptive Histogram Equalization*. Merupakan metode *preprocessing* gambar melalui proses *CLAHE* yang setelah itu diproses kembali dengan *gaussian blur*. *CLAHE* adalah proses *Adaptive Histogram Equalization* yaitu *Histogram Equalization* yang berlangsung pada *kernel window* yang

digeser ke seluruh *field* gambar. Lalu terdapat *attribute clip limit* yang membatasi frekuensi kemunculan suatu *pixel* pada titik tertentu. Lalu kelebihan dari keseluruhan *pixel* tersebut dibagi merata ke seluruh *pixel* pada *kernel* tersebut. Pada penelitian kali ini akan dilakukan uji coba mengenai tingkat akurasi model dengan data latih yang sudah diproses dengan *MCLAHE* dengan yang belum.

Penggunaan metode ini membagi jumlah frekuensi kemunculan *pixel* pada *channel L* pada gambar. Mempengaruhi tingkat *contrast* gambar. Oleh karenanya yang menjadi *indicator* adalah *noise contrast*.

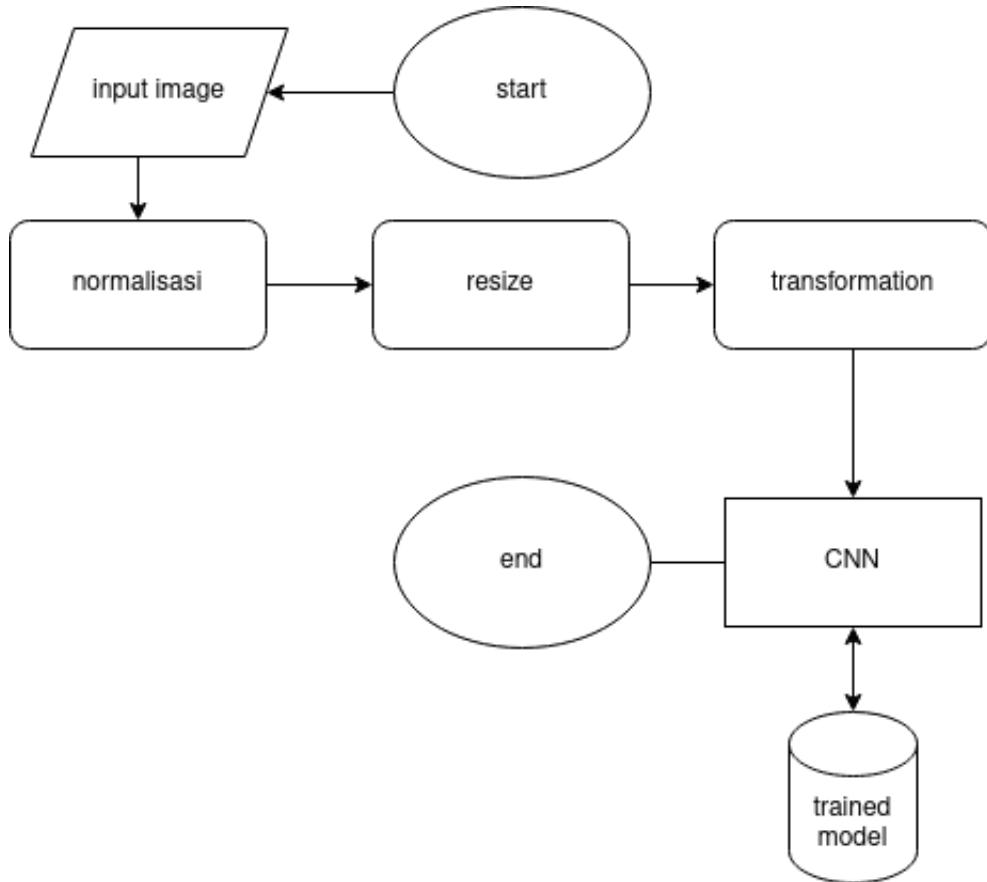
- (b) *Architectural* merupakan jenis arsitektur *convolutional neural network* yang digunakan saat proses pelatihan model. Jenis arsitektur ini akan menentukan bagaimana *neural network* mengekstraksi fitur pada gambar yang nantinya akan dipelajari pada lapisan *artificial neural network* untuk dilakukan proses klasifikasi. Perbedaan pola pada arsitektur *neural network* tentunya hasil fitur yang diperoleh juga akan berbeda. Penelitian ini bertujuan untuk menganalisa jenis arsitektur yang menghasilkan akurasi paling tinggi.
- (c) *Optimizers* merupakan *hyperparameter* yang bertujuan untuk menurunkan nilai *loss*. Berkaitan erat dengan *learning rate*, terdapat beberapa metode *optimizer* yang menggunakan sistem *decay* atau meng-update *learning rate* setiap *epoch*-nya. Pada penelitian kali ini, jenis *optimizer* yang dipakai adalah sperti yang sudah dijelaskan pada bab 2.1.8 yaitu *Stochastic Gradient Descent*, *Nesterov Accelerated Gradient*, *Adagrad*, *Adadelta* dan *Adam*. Tujuan akhir dari *optimizer* adalah menurunkan nilai *loss*. Terdapat satu lagi istilah lain yaitu *Regularizer*, ini bermaksud agar model dapat tergeneralisasi pada jenis gambar lainnya yang tidak dilatih pada proses sebelumnya. Contoh implementasi dari *regularizer* adalah *image augmentation*.
2. *Proposed Method* adalah bagian yang menjelaskan proses penelitian dari awal hingga akhir. Proses ini dimulai dengan mempersiapkan *dataset*. *Dataset* yang ada kemudian di-*preprocess*, lalu dilanjutkan dengan meng-upload *dataset* original dengan *dataset* yang sudah di-*preprocess* ke dalam *google drive*. Lalu *dataset* dikirimkan kepada model *convolutional neural network*. Model *convolutional neural network* kemudian akan dilatih dan diuji untuk memperoleh hasil akurasi pengenalan wajah.
3. *Objectives* adalah bagian yang menjelaskan target yang akan menjadi acuan pengukuran. Dalam penelitian ini, target tersebut adalah akurasi dari

pengenalan wajah oleh *convolutional neural network*.

4. *Measurement* adalah satuan ukur yang digunakan untuk mengukur hal-hal yang ada pada bagian *Objectives*. Dalam penelitian ini, karena hendak melakukan *multiclass classification* maka *cost function* yang dipakai adalah *Categorical cross entropy*. Pengukuran akurasi menggunakan *f1-score*, yang merupakan *harmonic mean* antara *precision* dan *recall*. *Precision* adalah proporsi dari data yang diidentifikasi sebagai sesuatu yang positif yang benar-benar positif. *Recall* adalah proporsi dari segala sesuatu yang positif yang berhasil kita tebak. *F1score* disebut juga *sorensen dice coefficient* atau *dice similarity coefficient*. Nilai tertingginya adalah 1.0 yang mengindikasikan nilai sempurna untuk *precision* dan *recall*. Sementara nilai terendahnya adalah 0 yang mengindikasikan *Recall* bernilai 0 ataupun *precision* bernilai 0 [24]. Untuk persamaan *precision* dan juga *recall* dapat dilihat pada subbab 2.1.10

3.3 Analisis Urutan Proses Global

Penelitian ini akan menggunakan *convolutional neural network* untuk melakukan pengenalan wajah. Setelah dilakukan pelatihan, diharapkan arsitektur yang dibangun dapat menghasilkan akurasi yang baik. Gambar 3.2 menunjukkan urutan proses global pada penelitian ini dalam bentuk *flowchart*.

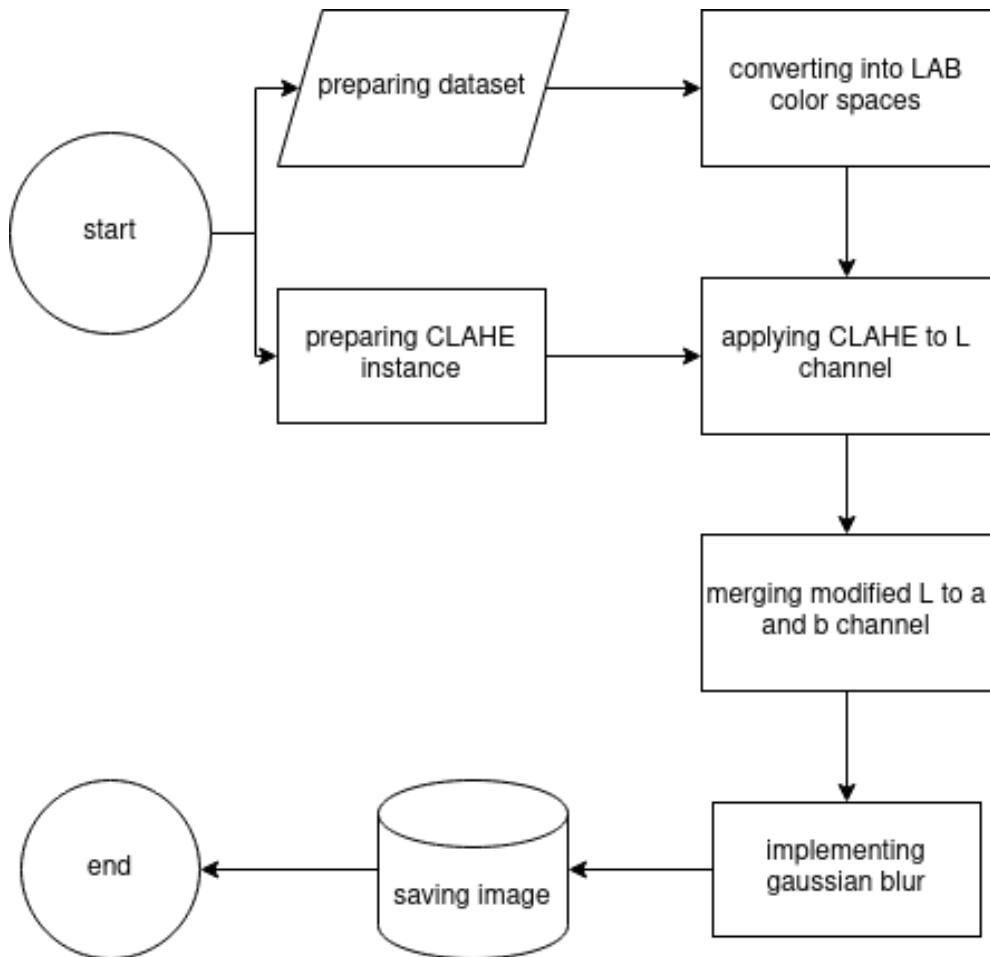


Gambar 3.2 Flowchart urutan proses global

Tahapan identifikasi wajah terbagi atas tiga proses, yaitu *preprocessing*, proses *training* dan proses *testing*. *Preprocessing* dilakukan untuk mengaplikasikan algoritma *Modified Contrast Limited Adaptive Histogram Equalization* sesuai dengan yang tertulis pada jurnal [1]. Proses *training* dilakukan untuk mendapatkan model *convolutional neural network* yang optimal untuk menghasilkan akurasi yang tinggi. Proses *testing* dilakukan untuk menguji generalisasi model.

3.3.1 Pemrosesan awal

Pada penelitian ini, proses *preprocessing* digambarkan pada Gambar 3.3.



Gambar 3.3 *Preprocessing*

Berikut adalah uraian dari proses pada *flowchart 3.3* yang dilakukan pada penelitian ini.

1. Pada penelitian ini, digunakan dua jenis *dataset* yaitu *Cropped Extended Yale Face Database* dan *Komnet Dataset*. Seperti yang dijelaskan pada subbab 2.3.1, *Cropped Extended Yale Face Database* adalah *Extended Yale Face Database* yang diproses dengan algoritma *Viola-Jones* sehingga hasilnya adalah gambar wajah yang menghadap kedepan (*frontal face*). *Komnet Dataset*, seperti yang sudah dijelaskan pada subbab 2.3.2, adalah data gambar wajah yang menghadap kedepan (*frontal face*) yang dibuat oleh laboratorium politeknik Bali. *Dataset* diambil dari banyak sumber, diantaranya sosial media, kamera digital dan kamera *smartphone*. Namun, untuk penelitian kali ini hanya menggunakan data gambar yang diambil dari *smartphone*.
2. Menyusun *code* untuk mengkonversi gambar dari *BGR channel* ke *LAB channel*. Seperti yang sudah dijelaskan pada subbab 2.1.9,⁴ *LAB channel* terdiri dari tiga *channel* yaitu, *L* yang merupakan *brightness* dari gambar, *a channel*

BGR ke *channel LAB* dilakukan oleh pustaka *OpenCV* [28]. Algoritma untuk konversi menuju *LAB channel* dapat dilihat pada algoritma 2.3.

3. Mempersiapkan code untuk membangkitkan *CLAHE instance*. Algoritma *CLAHE* akan diaplikasikan pada gambar dengan menggunakan metode dari *instance* yang dibangun.

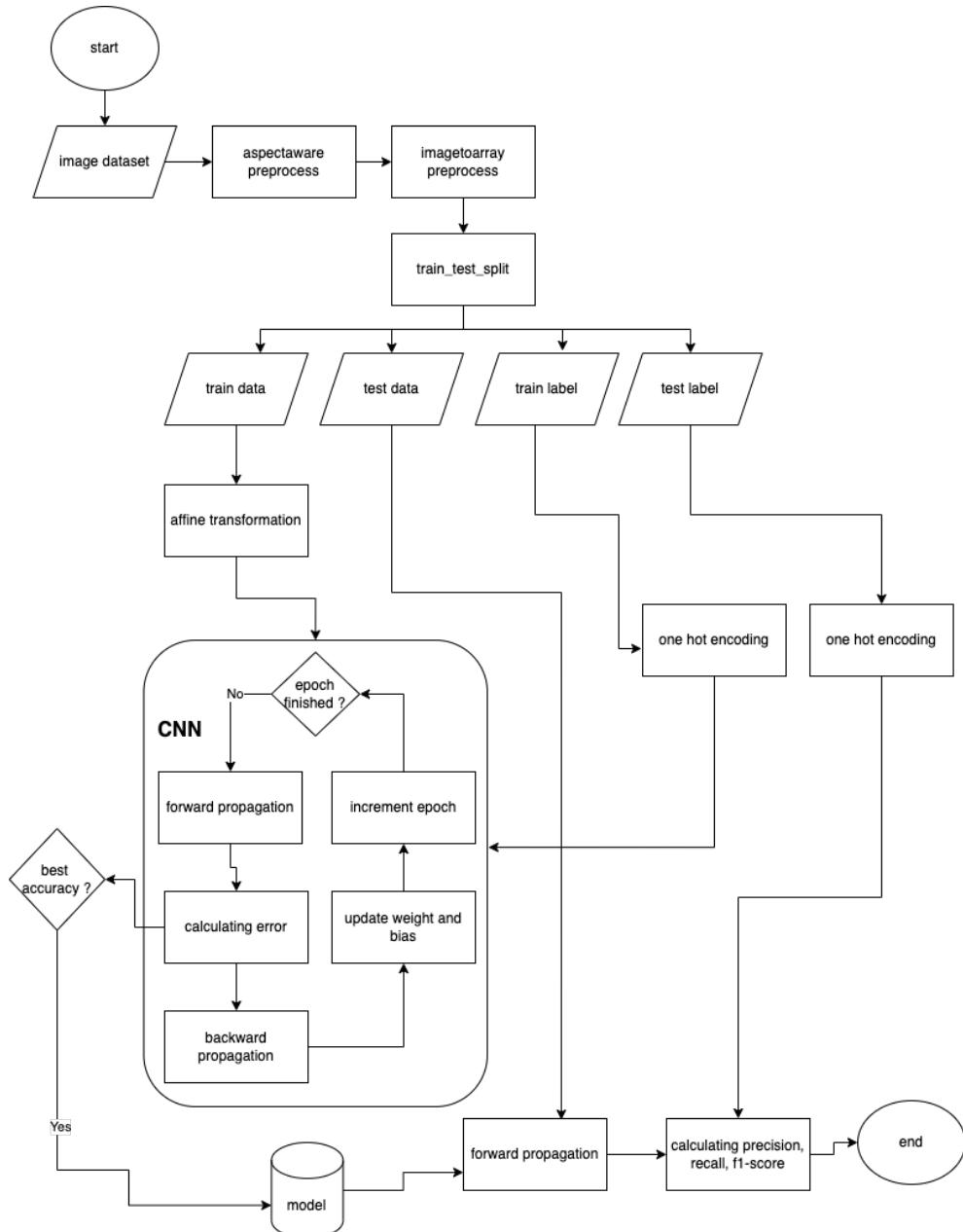
ALGORITME 3.1 *CLAHE algorithm*

- 1: Masukan dari fungsi adalah nilai channel L,a,b, *image path* dan *clip limit*.
- 2: Langkah selanjutnya, membangun *clahe instance* dengan menginisiasi *cliplimit* dengan ukuran kernel yang dipakai. (mirip proses konvolusi).
- 3: Selanjutnya adalah mengaplikasikan operasi *CLAHE* pada channel L. *CLAHE* adalah sebagaimana dijelaskan pada bab 2.1.9,8 adalah operasi *CLAHE* yang dilanjutkan dengan *gaussian blur*. Untuk algoritma detailnya bisa dilihat pada 3.6.
- 4: Selanjutnya adalah proses *merging* channel L yang sudah diproses dengan channel a dan b.
- 5: Mengkonversi channel LAB kembali ke bentuk *BGR*.
- 6: Mengaplikasikan *gaussian blur*.
- 7: Saving gambar baru ke disk.

Setelah *CLAHE* diaplikasikan pada gambar, sesuai dengan jurnal [1], gambar selanjutnya diproses dengan menggunakan *Gaussian blur*. Selanjutnya hasil gambar yang telah di-*preprocess*, di-*save* pada *disk*. Selanjutnya akan di-*upload* ke dalam *google drive* untuk dijadikan data latih model *neural network*.

3.3.2 Proses *Training* (Pelatihan)

Pada penelitian ini, proses pelatihan model *convolutional neural network* digambarkan pada Gambar 3.4.



Gambar 3.4 Flowchart pelatihan sistem pengenalan wajah

Berikut adalah uraian proses pelatihan dari *flowchart* pada Gambar 3.4 yang dilakukan dalam penelitian ini.

1. Sebagai masukan untuk *convolutional neural network* yang dibangun, digunakan data gambar wajah (*frontal face*) yang kemudian diubah ukurannya menjadi 224×224 pixel lewat proses *AspectAwareRatio*.
2. Data gambar yang sudah di-*resize* kemudian dikonversi ke dalam bentuk *array* melalui proses *imageToArray*.
3. Selanjutnya data dibagi menjadi *test data* dan *train data* melalui metode pada *scikit-learn* yaitu *train_test_split*.

4. Label untuk *train data* dan *test data* masing-masing dikonversi ke dalam bentuk matrix (*one hot encoding* dengan menggunakan metode *scikit-learn* yaitu *LabelBinarizer*).
5. *train data* di-*augmentasi* menggunakan *ImageGenerator* dari *keras*. Proses augmentasi yang dilakukan adalah *random rotation* dengan rentang 0 hingga 30 derajat. Selanjutnya adalah *random shifting* baik pada *width* dan *height* dengan rentang 0 hingga 10%. Selanjutnya *random shear* dengan rentang 0 hingga 20%. Selanjutnya adalah *random zoom* dengan rentang nilai 0 hingga 20%. Augmentasi *horizontal flip* yaitu *flip* gambar berdasarkan sumbu x juga diaplikasikan pada penelitian kali ini. Proses *augmentasi* ini dijalankan dengan *attribute fill_mode* bernilai *nearest*, ini artinya *pixel* yang kosong karena transformasi gambar akan diisi dengan nilai *pixel* yang terdekat. Untuk penjelasan mengenai proses *augmentasi* ini lebih jelasnya dapat dibaca pada subbab [2.1.9.3](#).
6. Selanjutnya adalah tahap pelatihan model. Proses pelatihan model *convolutional neural network* terdiri atas dua proses besar, yaitu *forward propagation* dan *backward propagation*. Pada subbab [3.3.2.1](#) dan [3.3.2.2](#) akan dijelaskan tahap pelatihan dengan *convolutional neural network* secara *forward propagation* dan *backward propagation*.
7. Hasil keluaran dari proses pelatihan model adalah model *deep learning* yang disimpan di dalam *disk*.
8. Model yang dihasilkan diuji berdasarkan penilaian *measurement* seperti yang dijelaskan pada subbab [3.2](#) yaitu *f1 score*, maka sebelumnya perlu mendapatkan hasil prediksi model melalui metode *predict*.
9. Setelah hasil dari *model.predict* diperoleh, *f1 score* dapat dihitung melalui metode *classification_report*.

3.3.2.1 Forward Propagation Convolutional Neural Network

Berikut ini adalah algoritma dalam proses *forward propagation* untuk menghasilkan fitur dan pengenalan wajah *face recognition*. Proses ini dilakukan secara berulang berdasarkan jumlah *epoch*, dan dalam satu *epoch*, akan ada sejumlah *batch* untuk dilatih. *Batch* merupakan *hyperparameter* yang mengontrol jumlah sampel pelatihan yang harus diproses algoritma dalam satu *epoch*. *Epoch* menentukan berapa kali algoritma pembelajaran akan melakukan iterasi pada data pelatihan untuk belajar. algoritma [3.2](#) akan menjelaskan secara umum bagaimana proses *forward propagation* dilakukan.

ALGORITME 3.2 Forward Propagation Convolutional Neural Network

- 1: Masukan merupakan data wajah yang sudah di-*preprocessing* dengan ukuran 224×224 pixel.
 - 2: Lakukan *resize* gambar ke ukuran 224×224 pixel.
 - 3: Inisialisasi nilai *epoch* dan *learning rate*, *batch_size*, dan *kernel*. Nilai inisiasi mengikuti arsitektur *neural network* yang dipakai yaitu *VGG16* dan *Inception (GoogleNet)*.
 - 4: Lakukan padding dengan metode *same padding*
 - 5: Lakukan proses konvolusi pada setiap *convolutional layer*.
 - 6: Operasikan fungsi aktivasi ReLU untuk setiap keluaran dari *convolutional layer*.
 - 7: Lakukan operasi berikut ini sesuai urutan dari rancangan arsitektur yang dibangun:
 - Lakukan proses *batch normalization* untuk normalisasi hasil konvolusi.
 - Lakukan proses *concatenate* untuk menggabungkan hasil keluaran dari beberapa blok konvolusi.
 - Lakukan *max pooling* dari *layer* yang telah dikonvolusi untuk mengurangi dimensi keluaran *feature map*.
 - Lakukan *average pooling* dari *layer* yang telah dikonvolusi untuk mengurangi dimensi keluaran *feature map*.
 - 8: Lakukan proses *dropout* dan inisialisasi nilai *dropout*. *Dropout* dilakukan agar tidak terjadi *overfitting*.
 - 9: Lakukan proses *flatten* untuk merubah tensor kedalam bentuk *array* 1 dimensi agar bisa diproses pada *fully connected layer*.
 - 10: Masukkan seluruh *feature map* setelah proses *flatten* ke dalam *dense layer*.
 - 11: Hitung keluaran dari *dense layer* untuk mencari probabilitas dari setiap objek (*face image*) serta menghitung nilai *loss* atau *error*.
 - 12: Lakukan proses *backward* untuk menghitung nilai gradien lokal dan melakukan *update bobot* pada setiap *layer*.
 - 13: Simpan model untuk digunakan pada saat pengujian. Karena pada penelitian ini digunakan *callback* berupa *ModelCheckpoint*, maka hanya model yang terdapat perkembangan dari model sebelumnya yang disave.
-

3.3.2.2 Backward Propagation Convolutional Neural Network

backward propagation bertujuan untuk memperbarui nilai bobot (*weight*) pada *hidden layer neural network* untuk meminimalkan nilai *loss* agar dapat melakukan pengenalan wajah (*face recognition*) dengan lebih baik. Pembaruan nilai bobot

dilakukan dengan menggunakan beragam *optimizer* seperti yang dijelaskan pada bab 2.1.8. Untuk sistem matematis bagaimana *backward propagation* berlangsung dapat dilihat pada bab 2.1.3.

ALGORITME 3.3 Backward Propagation Convolutional Neural Network without optimizer

- 1: Menghitung nilai error pada *Hidden layer* dengan cara melakukan operasi *dot product* antara *transpose weight* dengan nilai *cost function*.
 - 2: Mencari turunan pertama dari fungsi aktivasi yang dipakai.
 - 3: Mencari nilai *transpose* dari input masukan
 - 4: mengacu kepada persamaan 2.9, nilai error yang didapat, turunan pertama dari fungsi aktivasi dan *transpose* dari nilai masukan pada persamaan sebelumnya akan digunakan untuk mengubah nilai bobot.
-

ALGORITME 3.4 Backward Propagation Convolutional Neural Network with optimizer

- 1: Menghitung nilai error pada *Hidden layer* dengan cara melakukan operasi *dot product* antara *transpose weight* dengan nilai *cost function*.
 - 2: Menghitung nilai *gradient* pada persamaan 2.19.
 - 3: Menggunakan nilai *gradient* untuk mengkalkulasi bobot dan bias yang baru menggunakan algoritma *optimizer* yang dipakai. Teori mengenai *optimizer* dapat dilihat pada subbab 2.1.8, sedangkan implementasinya dapat dilihat untuk masing-masing *optimizer* yang dipakai pada subbab 3.4.3.
-

Algoritma *backward propagation* secara garis besar tergambar pada algoritma 3.3. Untuk setiap *optimizer* terdapat rumusan khusus masing masing seperti yang tergambar pada subbab 2.1.8.

3.3.3 Proses Testing (Pengujian)

Pada penelitian ini, proses pengujian model digambarkan pada Gambar 3.4.

Berikut ini adalah uraian proses pengujian dari *flowchart* pada Gambar 3.4 yang dilakukan dalam penelitian ini.

1. Masukan merupakan data gambar wajah hasil dari proses *preprocessing* dengan ukuran 224×224 pixel. Data yang digunakan adalah data yang sudah dipisahkan dikategorikan sebagai *test data*.

2. Lakukan proses *one hot encoding* pada *test label* dengan menggunakan metode dari *scikit-learn* yaitu *Label Binarizer*.
3. Ambil model dari *convolutional neural network* yang sudah dihasilkan pada proses pelatihan.
4. Gunakan model tersebut untuk melakukan pengenalan wajah terhadap data uji. Model ini digunakan untuk melakukan *forward propagation* pada data pengujian. Proses *forward propagation* pada tahap pengujian sama dengan proses *forward propagation* pada tahap pelatihan pada algoritma 3.2. Hasil dari proses prediksi yang dilakukan model ini adalah *array one hot encoding* yang menunjukkan klasifikasi dari *datapoint* tersebut.
5. Setelah didapatkan hasil estimasi yang adalah *array one hot encoding*, sekvens tersebut dibandingkan dengan *output label* dari *dataset* untuk masing-masing *datapoint*. Perbandingan tersebut berlangsung pada sebuah metode yang disebut *classification_report* dari pustaka *scikit-learn*. Metode ini pada akhirnya akan mengeluarkan *report* berupa informasi *accuracy*, *recall* dan *f1-score* dari masing-masing *datapoint*.

3.4 Analisis Manual

Pada bagian ini, dilakukan analisis tahapan proses yang akan dilakukan dalam sistem. Analisis dilakukan dengan contoh dan ilustrasi perhitungan manual.

3.4.1 Dataset

Dataset yang digunakan dalam penelitian ini ada dua jenis, yaitu *Cropped Extended Yale Face Database* dan *Komnet Dataset*. Seperti yang sudah dijelaskan pada subbab 2.3.1, *Cropped Extended Yale Face database* adalah *Extended Yale Face Database* yang diaplikasikan algoritma *Viola-Jones*, sehingga yang dihasilkan adalah gambar wajah yang menghadap ke depan (*frontal face*). *Cropped Extended Yale Face Database* mengandung 38 kelas dengan masing masing kelas terdiri dari 64 gambar dengan beragam jenis iluminasi pencahayaan. Gambar 2.61 adalah contoh dari dataset ini. Dimensi awal dari *cropped Yale* dataset adalah 168×192 pixel. Dengan format data .PNM (*Portable Any Map Image*), yang menyimpan satu gambar tanpa kompresi.

Dataset berikutnya yang dipakai pada penelitian ini adalah *Komnet dataset*. *Dataset* ini dibuat oleh laboratorium politeknik Bali, mengandung 50 kelas dengan masing masing kelas terdiri dari 24 gambar berukuran 224×224 pixel. Gambar berupa wajah yang menghadap ke depan (*frontal face*). Gambar diambil dengan menggunakan media kamera digital, sosial media dan juga kamera *smartphone*.

Gambar 2.62 adalah contoh dari *dataset* ini. Dimensi awal dari gambar ini adalah 224×224 pixel dengan format jpeg.

3.4.1.1 Aspect Aware Preprocessor and Image to Array preprocessor

Dalam rangka *neural network* dapat bekerja dengan optimal, kita perlu menyeragamkan ukuran dimensi gambar masukan. Teknik *image resizing* yang dipakai pada penelitian kali ini merupakan *aspect aware preprocessor* seperti yang dijelaskan pada algoritma 2.1.

Setelah gambar diubah dimensinya, langkah selanjutnya adalah mengkonversi gambar kedalam bentuk *numpy array*. Metode ini diselesaikan dengan menggunakan pustaka *keras* dari modul *image preprocessing* yaitu *img_to_array*.

3.4.1.2 Modified Contrast Limited Adaptive Histogram Equalization

MCLAHE sejatinya adalah *CLAHE* yang setelahnya diaplikasikan *gaussian blur*. Untuk algoritma dari *MCLAHE* lebih jelasnya akan dibahas pada 3.6.

Berikut adalah algoritma untuk mengaplikasikan *MCLAHE*:

ALGORITME 3.5 algoritma Histogram Equalization

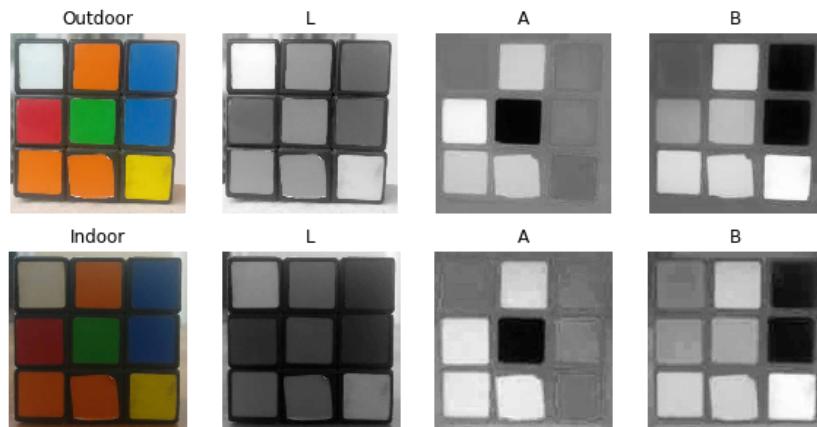
- 1: Menghitung frekuensi kemunculan dari masing-masing *grayscale pixel* (N_k)
 - 2: Menghitung *Probability Distribution Function (PDF)* dari masing masing *pixel* dengan membagi frekuensi kemunculan *pixel* dengan keseluruhan jumlah *pixel* yang ada.
 - 3: Menghitung *Cumulative Distribution Function (CDF)* dari masing masing *pixel*. Dihitung dengan cara, pertama *pixel* diurutkan secara *ascending*. Lalu karena akan dihitung secara kumulatif, maka perhitungannya adalah penjumlahan *PDF* pada *pixel* yang sekarang dengan yang total sebelumnya.
 - 4: Kemudian Menghitung perkalian *CDF* dengan total kelas *pixel* yang ada.
 - 5: Membulatkan hasil dari tahap sebelumnya dengan peraturan pembulatan matematika dasar. Misalnya bila angka desimal diatas .5 maka dibulatkan keatas. Maka itulah perubahan nilai *pixel* dari *pixel* tersebut.
-

ALGORITME 3.6 Algoritma MCLAHE

- 1: mengkonversi gambar dari *BGR channel* ke *LAB channel*. Penjelasan mengenai *LAB channel* terdapat pada subbab 2.1.9.4 dan algoritma yang

digunakan dapat dilihat pada 2.3. Contoh pemetaan kepada *LAB channel* dapat dilihat pada gambar 3.5.

- 2: membangun *CLAHE instance* lalu mengaplikasikan algoritma *CLAHE* pada channel l dari gambar. Code dapat dilihat pada 3.3.1.
 - 3: Algoritma *CLAHE* pada dasarnya mirip dengan 3.5, hanya saja perhitungan *Histogram Equalization* dilakukan pada *kernel window* yang bergerak ke seluruh field gambar (konsep *Adaptive Histogram Equalization*). *CLAHE* menetapkan *clip limit*, batas maksimum dari suatu frekuensi yang diizinkan pada proses tersebut. Jika sudah melebihi limit yang ditentukan maka nilai frekuensi akan di potong, nilai lebihnya akan ditampung. Pada akhir proses, nilai lebihnya akan dibagi merata kepada seluruh pixel yang ada pada wilayah *kernel* tersebut. Pada jurnal [1] dikatakan bahwa modifikasi dari *CLAHE* yang mereka pakai adalah melanjutkan prosesnya dengan *gaussian blur*.
 - 4: Selanjutnya channel L yang sudah diproses dengan metode *CLAHE*, di-merge dengan channel a dan b.
 - 5: Lalu gambar diproses dengan menggunakan metode *gaussian blur*.
 - 6: hasil akhir gambar di save pada disk.
-



Gambar 3.5 *LAB channel example*

3.4.2 *Forward Propagation*

Berikut ini akan dijelaskan perhitungan untuk proses-proses yang dilalui dalam *forward propagation* pada arsitektur *VGG16* dan *Inception*.

3.4.2.1 *VGG16*

Data gambar pada dasarnya adalah *array* dari pixel. Pixel direpresentasikan sebagai data numerik. Maka representasi gambar oleh komputer adalah berupa data data numerik yang disimpan dalam bentuk matriks. Contoh nilai matriks citra gambar dapat dilihat pada 3.1. Nilai ini akan digunakan untuk melakukan operasi konvolusi

BAB 3 ANALISIS DAN PERANCANGAN SISTEM

antara citra dan *kernel*. Pada penelitian ini, operasi konvolusi berjalan secara satu dimensi ke arah samping, hingga akhirnya keseluruhan pixel pada gambar berhasil diproses.



Gambar 3.6 VGG16 architecture

Tabel 3.1 Contoh matriks citra

120	120	100	100	100
120	120	200	100	100
150	150	150	100	100
100	100	100	100	100
50	80	90	100	200

Berdasarkan algoritma 3.2, citra masukan akan diberikan *padding*. *Padding* digunakan agar citra masukan sepenuhnya tertutup oleh *kernel* sehingga menghasilkan keluaran dengan ukuran yang sama dengan masukan. Tabel 3.2 menunjukkan contoh citra matriks yang sudah diberikan *padding*.

Tabel 3.2 Contoh matriks citra dengan *padding*

0	0	0	0	0	0	0
0	120	120	100	100	100	0
0	120	120	200	100	100	0
0	150	150	150	100	100	0
0	100	100	100	100	100	0
0	50	80	90	100	200	0
0	0	0	0	0	0	0

Kernel yang digunakan dalam penelitian beraneka ragam dimensinya, tergantung pada jenis arsitektur yang dipakai. Pada kasus VGG16 semua *kernel* berukuran 3×3 . Pada kasus *Inception*, *kernel* yang dipakai ada yang berukuran $1 \times 1, 3 \times 3$ maupun 5×5 . Dengan mengoperasikan nilai *kernel* kepada citra masukan, akan didapatkan sebuah *feature map* satu dimensi. Proses konvolusi bergerak secara satu dimensi, di mana pergerakan *kernel* adalah ke arah samping kanan. Nilai *kernel* pada contoh perhitungan ini berukuran 3×3 . Nilai awal *kernel* dalam contoh perhitungan manual dan yang digunakan dalam penelitian didapatkan dengan metode *He Initialization* sesuai dengan persamaan 2.25, karena metode ini mendukung penggunaan ReLU.

Tabel 3.3 Contoh nilai *kernel* 3×3

0.14590815	0.02599214	0.04402942
0.00121707	0.15733334	0.05069766
0.13805986	0.00803913	0.11923093

Perhitungan untuk sel [0, 0], sel [0,1], dan sel-sel selanjutnya memiliki pola yang sama yang dapat dilihat pada Persamaan 2.4. Perhitungan dilakukan hingga mencapai sel [x, y] yang berada pada ujung kanan bawah matriks masukan. Hasil

perkalian (konvolusi) $w \times x$ disebut sebagai *feature map*. Proses perkalian dilakukan menggunakan *stride* bernilai 1.

Hasil perhitungan operasi konvolusi sesuai dengan Persamaan 2.4. Nilai *bias* yang digunakan adalah 0, di mana digunakan parameter `bias_initializer = 0`. Hasil konvolusi antara citra dan *kernel* pada contoh kasus sel [0, 0] adalah sebagai berikut.

$$\begin{aligned}
 v_k &= (w_{k_{0,0}}x_{0,0} + b_{0,0}) + (w_{k_{0,1}}x_{0,1} + b_{0,1}) + (w_{k_{0,2}}x_{0,2} + b_{0,2}) + (w_{k_{1,0}}x_{1,0} + b_{1,0}) \\
 &\quad + (w_{k_{1,1}}x_{1,1} + b_{1,1}) + (w_{k_{1,2}}x_{1,2} + b_{1,2}) + (w_{k_{3,0}}x_{3,0} + b_{3,0}) + (w_{k_{3,1}}x_{3,1} + b_{3,1}) \\
 &\quad + (w_{k_{3,2}}x_{3,2} + b_{3,2}) \\
 &= (0 * 0.14590815 + 0) + (0 * 0.02599214 + 0) \\
 &\quad + (0 * 0.04402942 + 0) + (0 * 0.00121707 + 0) \\
 &\quad + (120 * 0.15733334 + 0) + (120 * 0.05069766 + 0) \\
 &\quad + (0 * 0.13805986 + 0) + (120 * 0.00803913 + 0) \\
 &\quad + (120 * 0.11923093 + 0) \\
 &= 39
 \end{aligned}$$

Setelah konvolusi selesai dilakukan, didapatkan *feature map* berukuran 3×3 seperti pada Tabel 3.4. Ukuran *feature map* dapat berubah bergantung pada jenis *padding* yang diaplikasikan, dimensi *kernel* yang digunakan dan juga penggunaan *stride* pada lapisan konvolusi. Jumlah *filter* akan berpengaruh kepada jumlah *feature map* yang dihasilkan. Jika *filter* yang digunakan berjumlah 1048 pada satu lapisan konvolusi, maka akan dihasilkan 1048 *feature map* pula. Jumlah *filter* yang digunakan bergantung kepada arsitektur yang dipakai.

Tabel 3.4 Contoh nilai *feature map* hasil konvolusi

39	62	47	46	27
62	86	91	76	40
58	95	82	83	40
50	78	74	89	43
22	41	44	47	49

Berdasarkan jenis arsitektur yang dipakai, setelah proses konvolusi *feature map*

kemudian dinormalisasi nilainya menggunakan *batch normalization* pada kasus *Inception*, tetapi pada kasus *VGG16* proses berikutnya adalah ReLU.

Nilai yang ada dalam *feature map* tersebut dimasukkan ke dalam fungsi aktivasi ReLU. Nilai positif akan tetap dan nilai negatif akan diubah menjadi nol. Berikut adalah contoh pengubahan nilai pada sel [0,1].

$$\begin{aligned}f(x) &= \max(0, x) \\&= \max(0, 39) \\&= 39\end{aligned}$$

Dengan menggunakan hasil perhitungan berikut, *feature map* yang dihasilkan menjadi seperti pada Tabel 3.5.

Tabel 3.5 Contoh nilai *feature map* setelah diaktivasi

39	62	47	46	27
62	86	91	76	40
58	95	82	83	40
50	78	74	89	43
22	41	44	47	49

Proses selanjutnya adalah *batch normalization*. Persamaan matematika dari *batch normalization* dapat dilihat pada 2.34. Berikut ini adalah contoh perhitungan untuk sel [0,0]. Adapun nilai gamma γ diinisialisasi 1, nilai beta β diinisialisasi 0, dan nilai epsilon ϵ diinisialisasi dengan 0.001.

$$\begin{aligned}
 \mu_B &= \frac{39 + 62 + 47 + \dots + 49}{25} \\
 &= 58.84 \\
 \sigma_B &= \sqrt{\frac{(39 - 58.84)^2 + \dots + (58.84)^2}{9}} \\
 &= 20.885 \\
 \check{x}_i &= \frac{x_i - \mu_B}{\sigma_B + \epsilon} \\
 &= \frac{39 - 58.84}{20.885 + 0.001} \\
 &= \frac{19.84}{20.886} \\
 &= -0.95 \\
 y_i &= (\check{x}_i * \gamma) + \beta \\
 &= -0.95 * 1 + 0 \\
 &= -0.95
 \end{aligned}$$

Feature map hasil perhitungan *batch normalization* dapat dilihat pada Tabel 3.6.

Tabel 3.6 Contoh *feature map* hasil perhitungan *batch normalization*

-0.95	0.151	- 0.567	- 0.615	- 1.525
0.151	1.299	1.539	0.8211	- 0.902
-0.041	1.731	1.108	1.156	- 0.902
-0.424	0.917	0.725	1.443	- 0.758
-1.764	- 0.854	- 0.711	- 0.567	- 0.471

Pada arsitektur *VGG16* terdapat 5 block. Untuk block yang pertama setelah gambar di konvolusi pada *convolution layer*, lalu diaktivasi oleh ReLU dan setelahnya di normalisasi oleh *batch normalization*. Selanjutnya *feature map* dilalukan kembali kepada *convolution layer*, ReLU dan *batch normalization*. Selanjutnya *feature map* dilalukan kepada *Max Pooling* dan *dropout layer*. *Max Pooling* yang digunakan pada *VGG16* menggunakan *pool size* = (2,2) dengan *strides* bernilai 2 dan *same padding*.

Berikut adalah contoh perhitungan untuk *max pooling* dimensi 2×2 dengan *same padding*

padding untuk keluaran pada sel [0, 0]:

$$\begin{aligned} max_{0,0} &= \max(x_{0,0}, x_{0,1}, x_{1,0}, x_{1,1}) \\ &= \max(0, 0, -0.95, 0) \\ &= 0 \end{aligned}$$

Hasil perhitungan *max pooling* dari *feature map* pada Tabel 3.5 dapat dilihat pada Tabel 3.7.

Tabel 3.7 Contoh hasil perhitungan *max pooling*

0	0	0
0	1.443	0
0	0	0

Dari *feature map* tersebut, selanjutnya dilakukan proses *dropout* dengan menggunakan Persamaan 2.36. Hal ini dilakukan untuk menonaktifkan beberapa nilai fitur agar *neural network* yang dibangun tidak terlalu banyak belajar. Parameter probabilitas yang digunakan adalah 0.25 pada arsitektur *VGG16* dan 0.4 pada arsitektur *Inception*.

Tabel 3.8 Contoh inisialisasi *random number* proses *dropout*

0.125	0.287	0.6281	0.734
-------	-------	--------	-------

Random number yang diinisialisasi tersebut akan dimasukkan pada Persamaan 2.36. Berikut adalah contoh perhitungannya.

$$\begin{aligned} O_i &= O_i \times \frac{1}{p} \\ &= 0.125 \times \frac{1}{0.25} \\ &= 0.5 \end{aligned}$$

Proses yang sama dilakukan untuk menghitung nilai keluaran lainnya. Tabel 3.9 menunjukkan hasil akhir perhitungan *dropout*.

Tabel 3.9 Contoh *random number* proses *dropout*

0.50	0	0	0
------	---	---	---

Sebelum dilalui kepada *fully connected layer*, *feature map* masih harus melalui block kedua hingga ke lima. Pada block kedua, prosesnya sama dengan proses pada block pertama. Jumlah filter pada konvolusi block pertama adalah 64, sementara pada block kedua adalah 128. Jika diskemakan adalah sebagai berikut *CONV* → *RELU* → *BATCHNORM* → *CONV* → *RELU* → *BATCHNORM* → *MAXPOOLING* → *DROPOUT*.

Proses pada block ketiga, empat dan lima juga memiliki kemiripan. Perbedaannya hanyalah pada jumlah filter konvolusinya. Pada block ketiga total filternya adalah 256 sementara pada block ke empat dan ke lima adalah 512. Skemanya adalah sebagai berikut *CONV* → *RELU* → *BATCHNORM* → *CONV* → *RELU* → *BATCHNORM* → *CONV* → *RELU* → *BATCHNORM* → *MAXPOOLING* → *DROPOUT*.

Pada *fully connected layer*, pertama tama *feature map* diproses dengan *flatten* sehingga menjadi *array* 1 dimensi. Selanjutnya dilakukan kepada *dense*.

Tabel 3.10 Contoh *kernel* pada lapisan *dense*

0.0778949	0.0586666	0.05155256	-0.01344849	0.055333	0.0111222	0.009333	0.1232323	0.01222212
-----------	-----------	------------	-------------	----------	-----------	----------	-----------	------------

Perhitungan pada lapisan ini dapat dilihat sebagai berikut. Pertama-tama, dilakukan perhitungan *dot product* antara masukan *feature map* dengan *kernel*.

$$\begin{aligned}
 y &= f(W^T x + b) \\
 &= f(0 * 0.0778949 + \dots + 0 * 0.0122 + 0) \\
 &= f(0.079845519)
 \end{aligned}$$

Hasil *output* akan dilewati kepada fungsi aktivasi (ReLU) selanjutnya akan dilakukan kembali kepada *batch normalization* lalu *Dropout* dengan nilai 0.5. Proses ini berlangsung sebanyak 2 kali. Lalu di akhir layer terdapat *output layer* dengan fungsi aktivasi *softmax*.

$$\begin{aligned}\phi_i &= \frac{e^{z_i}}{\sum_{j=1}^n e^{z_j}} \\ &= \frac{e^{0.0798}}{e^{0.0798} + e^0 + \dots + e^0} \\ &= 0.119240563\end{aligned}$$

Hasil akhir perhitungan adalah probabilitas setiap kelas *face* yang terdeteksi pada dataset terkait. Nilai probabilitas yang tertinggi akan diterima sebagai hasil estimasi

Proses yang sudah dijelaskan di atas belum sempurna sebagai proses pelatihan. Hasil perhitungan masih akan mengalami *error*. Untuk itu, tahap selanjutnya dari proses pelatihan *convolutional neural network* adalah melakukan *backward propagation*. Sebelum masuk ke proses *backward propagation*, akan dihitung terlebih dahulu nilai *loss* / *error* sesuai dengan persamaan 2.24. Contoh perhitungannya adalah sebagai berikut.

$$\begin{aligned}Loss &= -\log(\phi_i) \\ &= -\log(0.1192405633) \\ &= 0.923575982\end{aligned}$$

Kemudian, hasil regularisasi L2 ditambahkan ke nilai *loss* dengan Persamaan ?? untuk *weight* dan *bias*. Hasil penjumlahan dengan L2 menjadi nilai *loss* akhir yang akan dimasukkan ke dalam proses *backward propagation*.

1. Perhitungan L2 untuk *weight kernel* pada *dense layer* adalah sebagai berikut.

$$\begin{aligned}L_{2w} &= \lambda \sum_m w_m^2 \\ &= 0.01 * (0.0778949^2 + \dots + 0.01222^2) \\ &= 0.01 * 0.38590819 \\ &= 0.003859082\end{aligned}$$

2. Perhitungan L2 untuk *weight kernel* pada *convoutional layer* adalah sebagai berikut.

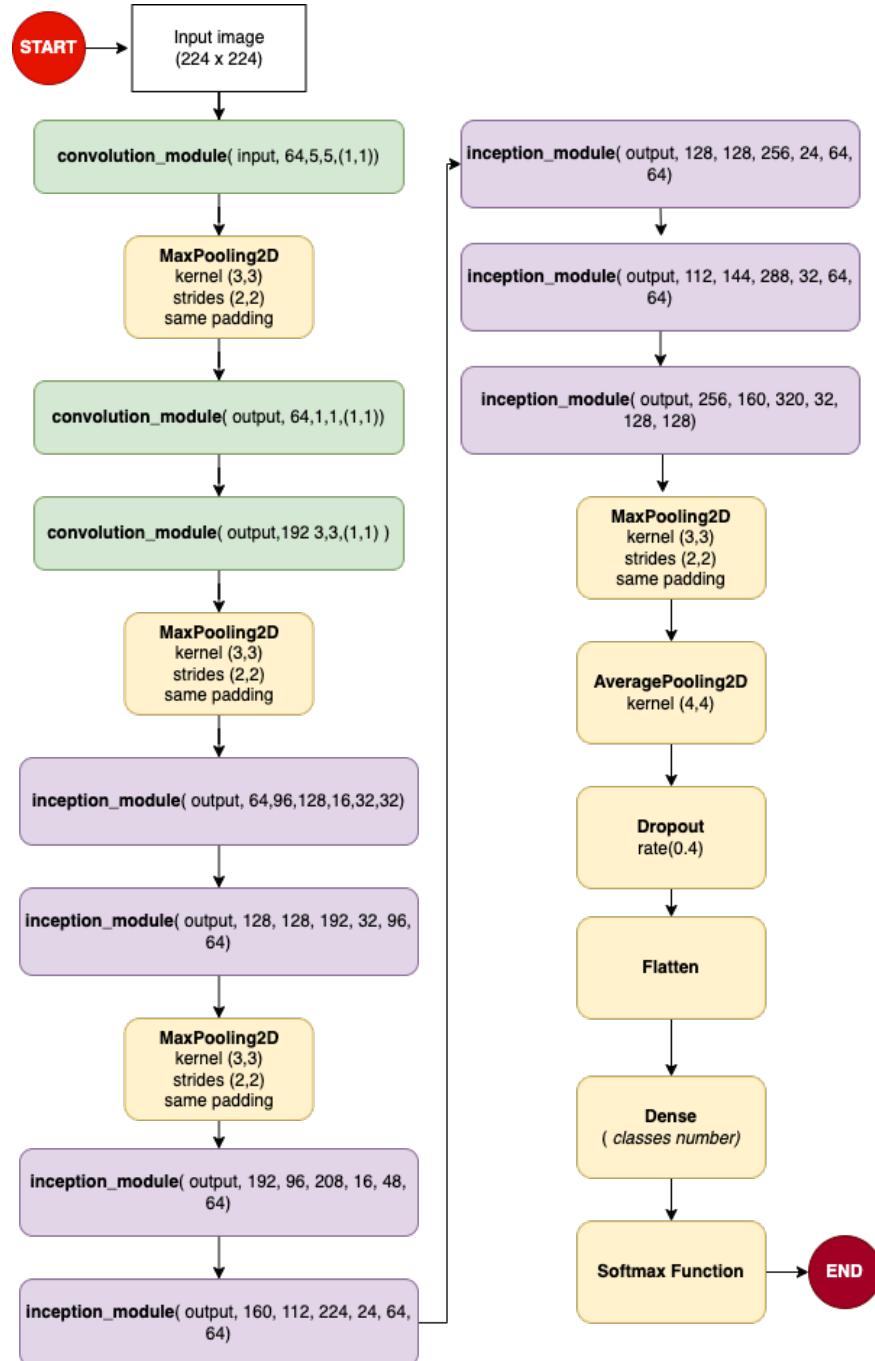
$$\begin{aligned}L_{2w} &= \lambda \sum_m w_m^2 \\&= 0.01 * (0.14590815^2 + \dots + 0.11923093^2) \\&= 0.01 * 0.6905077 \\&= 0.006905077\end{aligned}$$

3. Maka, nilai *loss* akhir adalah sebagai berikut.

$$\begin{aligned}Loss &= 0.923575982 + 0.003859082 + 0.006905077 \\&= 0.934340141\end{aligned}$$

3.4.2.2 Inception

Perbedaan mendasar antara *VGG16* dengan *Inception* adalah pada jenis modelnya. Pada *VGG16* model berjalan secara *sequential* sementara pada *Inception* berjalan secara *parallel*. Karenanya penggunaan metode *library*-nya berbeda. Pada *Inception* metode yang digunakan adalah *Input*. *Input* memungkinkan *keras* membangun dan menjalankan pelatihan model *neural network* secara *parallel*.



Gambar 3.7 Inception architecture

Secara grafis, proses pada arsitektur *Inception* dapat digambarkan pada gambar 3.7. Mengenai penjelasan *inception module* dan *convolution module* dapat dibaca pada subbab 2.1.4. Untuk penjelasan lebih mendetail mengenai proses grafik dapat dibaca pada penjelasan berikut.

Pada arsitektur *Inception*, gambar dengan dimensi 224×224 pada mulanya dilalui kepada *convolution layer* dengan filter sebanyak 64, dimensi *kernel* 5×5 dan *strides* bernilai 1. Block konvolusi ini diberi nama *block1*. Keluaran dari *block1* adalah

feature map dengan dimensi $224 \times 224 \times 64$. Setelahnya *feature map* dilalukan *Activation Layer*, *Batch Normalization Layer*, lalu dilanjutkan ke *Max pooling layer* dengan dimensi 3×3 , nilai *strides* = (2,2) dengan *same padding*. Hasil keluarannya adalah *feature map* dengan dimensi $112 \times 112 \times 64$.

Selanjutnya *feature map* dilalukan kembali ke *convolution layer* dengan spesifikasi jumlah filter sebanyak 64, besar dimensi 1×1 untuk mengurangi jumlah parameter seperti yang dibahas pada subbab 2.1.4, dan juga nilai *strides* = (1,1). Block ini diberi nama *block2*. Keluaran dari *block2* adalah *feature map* dengan dimensi $112 \times 112 \times 64$. Selanjutnya *feature map* dilalukan kembali kepada *convolution layer* dengan jumlah filter sebanyak 192, besar dimensi 3×3 dan nilai *strides* = 1. Block ini diberi nama *block3*. Keluaran dari *block3* adalah *feature map* dengan dimensi $56 \times 56 \times 192$. Jika diskemakan maka akan menjadi seperti berikut *CONV* → *BATCH* → *RELU* → *MAXPOOLING* → *CONV* → *BATCH* → *RELU* → *CONV* → *BATCH* → *RELU* → *MAXPOOLING*. satu rute dari *CONV* → *BATCH* → *RELU* adalah 1 module yang dinamakan *convolution module*. Untuk pembahasan berikutnya setiap kali penyebutan *convolution module* mengacu kepada rute ini.

Selanjutnya *feature map* akan diproses secara *parallel*. Dilalui ke proses yang dinamakan *Inception module*, pada modul ini terdapat 4 pemrosesan, *feature map* akan diproses secara *parallel* dan nanti hasilnya akan digabungkan menjadi satu *feature map*. Modul *inception* yang pertama adalah sebagai berikut :

1. *Feature map* dengan dimensi $56 \times 56 \times 192$ dilewatkan kepada *convolution module* dengan nilai filter 64 dan dimensi *kernel* 1×1 , nilai *strides* = 1 dan diberi nama *3a_first*. Keluaran dari block ini adalah *feature map* dengan dimensi $56 \times 56 \times 64$.
2. *Feature map* dengan dimensi $56 \times 56 \times 192$ dilewatkan kepada *convolution module* dengan nilai filter 96, dimensi *kernel* 1×1 , nilai *strides* = 1 dan diberi nama *3a_second1*. Hasil keluaran pada block ini adalah *feature map* dengan dimensi $56 \times 56 \times 96$. Nilai keluaran dari fungsi ini dilalukan kembali kepada *convolution module* dengan filter 128, dimensi *kernel* 3×3 , nilai *strides* = 1 dan diberi nama *3a_second2*. Hasil keluaran dari block ini adalah *feature map* dengan dimensi $56 \times 56 \times 128$.
3. *Feature map* dengan dimensi $56 \times 56 \times 192$ dilewatkan kepada *convolution module* dengan nilai filter 16, dimensi *kernel* 1×1 , nilai *strides* = 1 dan diberi nama *3a_third1*. Keluaran dari block ini adalah *feature map* dengan dimensi $56 \times 56 \times 16$. *Output* dari fungsi ini kemudian dilalukan kembali kepada

convolution module dengan nilai filter 32, dimensi *kernel* 5×5 , nilai *strides* = 1 dan diberi nama *3a_third2*. Hasil keluaran dari block ini adalah *feature map* dengan dimensi $56 \times 56 \times 32$.

4. *Feature map* dengan dimensi $56 \times 56 \times 192$ dilalukan kepada *Max Pooling* dengan dimensi *kernel* 3×3 , nilai *strides* = 1, *same padding* dan diberi nama *3a_pool*. Keluaran dari block ini adalah *feature map* dengan dimensi $56 \times 56 \times 32$. Lalu output dari fungsi ini dilalukan kepada *convolution module* dengan nilai filter 32, dimensi *kernel* 1×1 , nilai *strides* = 1 dan diberi nama *3a_fourth*. Keluaran dari block ini adalah *feature map* dengan dimensi $56 \times 56 \times 32$.
5. Pada tahap akhir, hasil keluaran dari ke empat proses di atas digabung menjadi 1 *feature map* dan dilalukan ke proses berikutnya.

Hasil keluaran pada *inception module* yang pertama adalah *feature map* dengan dimensi $56 \times 56 \times 256$. Pada proses berikutnya, *feature map* kembali dilalukan kepada *inception module* yang baru, tahapannya adalah sebagai berikut:

1. *Feature map* dilewatkan kepada *convolution module* dengan nilai filter 128 dan dimensi *kernel* 1×1 , nilai *strides* = 1 dan diberi nama *3b_first*. Keluaran pada block ini adalah *feature map* dengan dimensi $56 \times 56 \times 128$.
2. *Feature map* dilewatkan kepada *convolution module* dengan nilai filter 128, dimensi *kernel* 1×1 , nilai *strides* = 1 dan diberi nama *3b_second1*. Keluaran dari block ini adalah *feature map* dengan dimensi $56 \times 56 \times 128$. Nilai keluaran dari fungsi ini dilalukan kembali kepada *convolution module* dengan filter 192, dimensi *kernel* 3×3 , nilai *strides* = 1 dan diberi nama *3b_second2*. Hasil keluaran dari block ini adalah *feature map* dengan dimensi $56 \times 56 \times 192$.
3. *Feature map* dilewatkan kepada *convolution module* dengan nilai filter 32, dimensi *kernel* 1×1 , nilai *strides* = 1 dan diberi nama *3b_third1*. Hasil keluaran dari block ini adalah *feature map* dengan dimensi $56 \times 56 \times 32$. *Output* dari fungsi ini kemudian dilalukan kembali kepada *convolution module* dengan nilai filter 96, dimensi *kernel* 5×5 , nilai *strides* = 1 dan diberi nama *3b_third2*. Hasil keluaran dari modul ini adalah *feature map* dengan dimensi $56 \times 56 \times 96$.
4. *Feature map* dilalukan kepada *Max Pooling* dengan dimensi *kernel* 3×3 , nilai *strides* = 1, *same padding* dan diberi nama *3b_pool*. Lalu output dari fungsi ini dilalukan kepada *convolution module* dengan nilai filter 64, dimensi *kernel* 1×1 , nilai *strides* = 1 dan diberi nama *3b_fourth*. Hasil keluaran dari block ini adalah *feature map* dengan dimensi $56 \times 56 \times 64$.
5. Pada tahap akhir, hasil keluaran dari ke empat proses di atas digabung menjadi 1

feature map dan dilalukan ke proses berikutnya.

Hasil keluaran dari *inception module* ini adalah *feature map* dengan dimensi $56 \times 56 \times 480$. Selanjutnya *feature map* dilalukan kepada *Max Pooling* dengan dimensi *kernel* 3×3 , nilai *strides* = (2,2), *same padding* dan diberi nama *pool3*. Keluaran dari *pool3* adalah *feature map* dengan dimensi $28 \times 28 \times 480$. Selanjutnya *feature map* dilalukan kembali kepada 5 *inception module*. Module yang pertama adalah sebagai berikut:

1. *Feature map* dilewatkan kepada *convolution module* dengan nilai filter 192 dan dimensi *kernel* 1×1 , nilai *strides* = 1 dan diberi nama *4a_first*. Hasil keluaran dari block ini adalah *feature map* dengan dimensi $28 \times 28 \times 192$.
2. *Feature map* dilewatkan kepada *convolution module* dengan nilai filter 96, dimensi *kernel* 1×1 , nilai *strides* = 1 dan diberi nama *4a_second1*. Hasil keluaran dari block ini adalah *feature map* dengan dimensi $28 \times 28 \times 96$. Nilai keluaran dari fungsi ini dilalukan kembali kepada *convolution module* dengan filter 208, dimensi *kernel* 3×3 , nilai *strides* = 1 dan diberi nama *4a_second2*. Hasil keluaran dari block ini adalah *feature map* dengan dimensi $28 \times 28 \times 208$.
3. *Feature map* dilewatkan kepada *convolution module* dengan nilai filter 16, dimensi *kernel* 1×1 , nilai *strides* = 1 dan diberi nama *4a_third1*. Hasil keluaran dari block ini adalah *feature map* dengan dimensi $28 \times 28 \times 16$. *Output* dari fungsi ini kemudian dilalukan kembali kepada *convolution module* dengan nilai filter 48, dimensi *kernel* 5×5 , nilai *strides* = 1 dan diberi nama *4a_third2*. Hasil keluaran dari block ini adalah *feature map* dengan dimensi $28 \times 28 \times 48$.
4. *Feature map* dilalukan kepada *Max Pooling* dengan dimensi *kernel* 3×3 , nilai *strides* = 1, *same padding* dan diberi nama *4a_pool*. Lalu output dari fungsi ini dilalukan kepada *convolution module* dengan nilai filter 64, dimensi *kernel* 1×1 , nilai *strides* = 1 dan diberi nama *4a_fourth*. Hasil keluaran dari block ini adalah *feature map* dengan dimensi $28 \times 28 \times 64$.
5. Pada tahap akhir, hasil keluaran dari ke empat proses diatas digabung menjadi 1 *feature map* dan dilalukan ke proses berikutnya.

Hasil keluaran dari *inception modul* ini adalah *feature map* dengan dimensi $28 \times 28 \times 512$. Module *inception* yang kedua adalah sebagai berikut:

1. *Feature map* dilewatkan kepada *convolution module* dengan nilai filter 160 dan dimensi *kernel* 1×1 , nilai *strides* = 1 dan diberi nama *4b_first*. Hasil keluaran dari block ini adalah *feature map* dengan dimensi $28 \times 28 \times 160$.
2. *Feature map* dilewatkan kepada *convolution module* dengan nilai filter 112,

dimensi *kernel* 1×1 , nilai *strides* = 1 dan diberi nama *4b_second1*. Hasil keluaran dari block ini adalah *feature map* dengan dimensi $28 \times 28 \times 112$. Nilai keluaran dari fungsi ini dilalukan kembali kepada *convolution module* dengan filter 224, dimensi *kernel* 3×3 , nilai *strides* = 1 dan diberi nama *4b_second2*. Hasil keluaran dari block ini adalah *feature map* dengan dimensi $28 \times 28 \times 224$.

3. *Feature map* dilewatkan kepada *convolution module* dengan nilai filter 24, dimensi *kernel* 1×1 , nilai *strides* = 1 dan diberi nama *4b_third1*. Hasil keluaran pada block ini adalah *feature map* dengan dimensi $28 \times 28 \times 24$. *Output* dari fungsi ini kemudian dilalukan kembali kepada *convolution module* dengan nilai filter 64, dimensi *kernel* 5×5 , nilai *strides* = 1 dan diberi nama *4b_third2*. Hasil keluaran pada block ini adalah *feature map* dengan dimensi $28 \times 28 \times 64$.
4. *Feature map* dilalukan kepada *Max Pooling* dengan dimensi *kernel* 3×3 , nilai *strides* = 1, *same padding* dan diberi nama *4b_pool*. Lalu output dari fungsi ini dilalukan kepada *convolution module* dengan nilai filter 64, dimensi *kernel* 1×1 , nilai *strides* = 1 dan diberi nama *4b_fourth*. Hasil keluaran pada block ini adalah *feature map* dengan dimensi $28 \times 28 \times 64$.
5. Pada tahap akhir, hasil keluaran dari ke empat proses diatas digabung menjadi 1 *feature map* dan dilalukan ke proses berikutnya.

Hasil keluaran pada *inception module* ini adalah *feature map* dengan dimensi $28 \times 28 \times 512$. Module *inception* yang ketiga adalah sebagai berikut:

1. *Feature map* dilewatkan kepada *convolution module* dengan nilai filter 128 dan dimensi *kernel* 1×1 , nilai *strides* = 1 dan diberi nama *4c_first*. Hasil keluaran pada block ini adalah *feature map* dengan dimensi $28 \times 28 \times 128$.
2. *Feature map* dilewatkan kepada *convolution module* dengan nilai filter 128, dimensi *kernel* 1×1 , nilai *strides* = 1 dan diberi nama *4c_second1*. Hasil keluaran pada block ini adalah *feature map* dengan dimensi $28 \times 28 \times 128$. Nilai keluaran dari fungsi ini dilalukan kembali kepada *convolution module* dengan filter 256, dimensi *kernel* 3×3 , nilai *strides* = 1 dan diberi nama *4c_second2*. Hasil keluaran pada block ini adalah *feature map* dengan dimensi $28 \times 28 \times 256$.
3. *Feature map* dilewatkan kepada *convolution module* dengan nilai filter 24, dimensi *kernel* 1×1 , nilai *strides* = 1 dan diberi nama *4c_third1*. Hasil keluaran pada block ini adalah *feature map* dengan dimensi $28 \times 28 \times 24$. *Output* dari fungsi ini kemudian dilalukan kembali kepada *convolution module* dengan nilai filter 64, dimensi *kernel* 5×5 , nilai *strides* = 1 dan diberi nama *4c_third2*. Hasil keluaran pada block ini adalah *feature map* dengan dimensi $28 \times 28 \times 64$.

4. *Feature map* dilakukan kepada *Max Pooling* dengan dimensi *kernel* 3×3 , nilai *strides* = 1, *same padding* dan diberi nama *4c_pool*. Lalu output dari fungsi ini dilakukan kepada *convolution module* dengan nilai filter 64, dimensi *kernel* 1×1 , nilai *strides* = 1 dan diberi nama *4c_fourth*. Hasil keluaran pada block ini adalah *feature map* dengan dimensi $28 \times 28 \times 64$.
5. Pada tahap akhir, hasil keluaran dari ke empat proses diatas digabung menjadi 1 *feature map* dan dilakukan ke proses berikutnya.

Hasil keluaran pada *inception module* ini adalah *feature map* dengan dimensi $28 \times 28 \times 512$. Module *inception* yang ke empat adalah sebagai berikut:

1. *Feature map* dilewatkan kepada *convolution module* dengan nilai filter 112 dan dimensi *kernel* 1×1 , nilai *strides* = 1 dan diberi nama *4d_first*. Hasil keluaran pada block ini adalah *feature map* dengan dimensi $28 \times 28 \times 112$.
2. *Feature map* dilewatkan kepada *convolution module* dengan nilai filter 144, dimensi *kernel* 1×1 , nilai *strides* = 1 dan diberi nama *4d_second1*. Hasil keluaran pada block ini adalah *feature map* dengan dimensi $28 \times 28 \times 144$. Nilai keluaran dari fungsi ini dilakukan kembali kepada *convolution module* dengan filter 288, dimensi *kernel* 3×3 , nilai *strides* = 1 dan diberi nama *4d_second2*. Hasil keluaran pada block ini adalah *feature map* dengan dimensi $28 \times 28 \times 288$.
3. *Feature map* dilewatkan kepada *convolution module* dengan nilai filter 32, dimensi *kernel* 1×1 , nilai *strides* = 1 dan diberi nama *4d_third1*. Hasil keluaran pada block ini adalah *feature map* dengan dimensi $28 \times 28 \times 32$. Output dari fungsi ini kemudian dilakukan kembali kepada *convolution module* dengan nilai filter 64, dimensi *kernel* 5×5 , nilai *strides* = 1 dan diberi nama *4d_third2*. Hasil keluaran pada block ini adalah *feature map* dengan dimensi $28 \times 28 \times 64$.
4. *Feature map* dilakukan kepada *Max Pooling* dengan dimensi *kernel* 3×3 , nilai *strides* = 1, *same padding* dan diberi nama *4d_pool*. Lalu output dari fungsi ini dilakukan kepada *convolution module* dengan nilai filter 64, dimensi *kernel* 1×1 , nilai *strides* = 1 dan diberi nama *4d_fourth*. Hasil keluaran pada block ini adalah *feature map* dengan dimensi $28 \times 28 \times 64$.
5. Pada tahap akhir, hasil keluaran dari ke empat proses diatas digabung menjadi 1 *feature map* dan dilakukan ke proses berikutnya.

Hasil keluaran pada *inception module* ini adalah *feature map* dengan dimensi $28 \times 28 \times 528$. Module *inception* yang ke lima adalah sebagai berikut:

1. *Feature map* dilewatkan kepada *convolution module* dengan nilai filter 256 dan

- dimensi *kernel* 1×1 , nilai *strides* = 1 dan diberi nama *4e_first*. Hasil keluaran pada block ini adalah *feature map* dengan dimensi $28 \times 28 \times 256$.
2. *Feature map* dilewatkan kepada *convolution module* dengan nilai filter 160, dimensi *kernel* 1×1 , nilai *strides* = 1 dan diberi nama *4e_second1*. Hasil keluaran pada block ini adalah *feature map* dengan dimensi $28 \times 28 \times 160$. Nilai keluaran dari fungsi ini dilakukan kembali kepada *convolution module* dengan filter 320, dimensi *kernel* 3×3 , nilai *strides* = 1 dan diberi nama *4e_second2*. Hasil keluaran pada block ini adalah *feature map* dengan dimensi $28 \times 28 \times 320$.
 3. *Feature map* dilewatkan kepada *convolution module* dengan nilai filter 32, dimensi *kernel* 1×1 , nilai *strides* = 1 dan diberi nama *4e_third1*. Hasil keluaran pada block ini adalah *feature map* dengan dimensi $28 \times 28 \times 32$. Output dari fungsi ini kemudian dilakukan kembali kepada *convolution module* dengan nilai filter 128, dimensi *kernel* 5×5 , nilai *strides* = 1 dan diberi nama *4e_third2*. Hasil keluaran pada block ini adalah *feature map* dengan dimensi $28 \times 28 \times 128$.
 4. *Feature map* dilakukan kepada *Max Pooling* dengan dimensi *kernel* 3×3 , nilai *strides* = 1, *same padding* dan diberi nama *4e_pool*. Lalu output dari fungsi ini dilakukan kepada *convolution module* dengan nilai filter 128, dimensi *kernel* 1×1 , nilai *strides* = 1 dan diberi nama *4e_fourth*. Hasil keluaran pada block ini adalah *feature map* dengan dimensi $28 \times 28 \times 128$.
 5. Pada tahap akhir, hasil keluaran dari ke empat proses diatas digabung menjadi 1 *feature map* dan dilakukan ke proses berikutnya.

Hasil keluaran pada *inception module* adalah *feature map* dengan dimensi $28 \times 28 \times 832$. Selanjutnya *feature map* dilakukan kepada *Max Pooling* dengan dimensi *kernel* 3×3 , nilai *strides* = (2,2), *same padding* dan diberi nama *pool4*. Hasil keluarannya adalah *feature map* dengan dimensi $14 \times 14 \times 832$.

Selanjutnya *feature map* dilakukan kepada *average pooling* dengan dimensi *kernel* 4×4 dan diberi nama *pool5*. Hasil keluaran dari proses ini adalah *feature map* dengan dimensi $3 \times 3 \times 832$. Lalu *feature map* di *drop out* dengan nilai 0.4. Selanjutnya sebelum memasuki *fully connected layer*, *feature map* di *flatten* sehingga menjadi *array* 1 dimensi dengan jumlah feature 7488. Selanjutnya *feature map* dilakukan ke *dense layer* dengan fungsi aktivasi *softmax*. Hasil keluaran dari fungsi ini adalah probabilitas terhadap masing-masing kelas klasifikasi.

3.4.3 Backward Propagation

Backward propagation pada convolutional neural network digunakan dalam pelatihan untuk memperbarui bobot dari *kernel*.

3.4.3.1 Stochastic Gradient Descent

Berikut adalah contoh perhitungan menggunakan algoritma SGD pada convolutional neural network:

1. Inisialisasi $\alpha = 0.01$. Misalkan bobot masukan adalah 0.05981104, dengan asumsi perhitungan dari *loss function* mendapatkan nilai *gradient* $g_t = 0.934340141$ (lihat perhitungan *loss* pada *forward propagation*).
2. Perhitungan bobot baru, secara matematika dapat dilihat pada persamaan 2.19. Pada *keras library* perhitungannya mengikuti persamaan 3.1

$$w_t = w_0 - \alpha * g \quad (3.1)$$

Keterangan :

w_t	: bobot baru
w_0	: bobot lama
α	: <i>learning rate</i>
g	: <i>gradient</i>

$$\begin{aligned} m_t &= m_0 - \alpha * g \\ m_t &= 0.05981104 - 0.05 * 0.934340141 \\ m_t &= 0.05981104 - 0.046717007 \\ m_t &= 0.01311104 \end{aligned}$$

3. Nilai bobot sebelumnya adalah 0.05981104 dan diperbarui dengan nilai bobot baru 0.01311104 dalam sekali iterasi pada *neuron* pertama di lapisan keluaran.
4. Ulangi langkah pertama hingga kedua sampai jumlah *epoch* atau iterasi yang ditentukan.

5. Simpan semua bobot, *bias*, lapisan dan *neuron* untuk digunakan pada tahap pengujian.

3.4.3.2 Nesterov Accelerated Gradient

Berikut adalah contoh perhitungan menggunakan algoritma *NAG* pada *convolutional neural network*:

1. Inisialisasi $\alpha = 0.01$, $decay = 0.01 / 100$, $momentum = 0.9$ dan $Nesterov = True$. Misalkan bobot masukan adalah 0.05981104 dan *velocity* awal adalah 0, dengan asumsi perhitungan dari *loss function* mendapatkan nilai *gradient* g_t 0.934340141 (lihat perhitungan *loss* pada *forward propagation*).
2. Perhitungan bobot baru, secara matematika dapat dilihat pada persamaan 2.90 dan 2.91. Pada *keras library* perhitungannya mengikuti persamaan 2.41 dan 2.42.

$$\begin{aligned} w_t &= w_0 + m * v_0 - \alpha * g \\ w_t &= 0.05981104 + 0.9 * 0 - 0.01 * 0.934340141 \\ w_t &= 0.05981104 - 0.009343401 \\ w_t &= 0.05051104 \end{aligned}$$

3. Nilai bobot sebelumnya adalah 0.05981104 dan diperbarui dengan nilai bobot baru 0.01311104 dalam sekali iterasi pada *neuron* pertama di lapisan keluaran.
4. *learning rate* di setiap iterasinya pun mengalami perubahan sesuai dengan nilai *decay* yang kita tentukan diawal. Penurunan nilai *learning rate* sesuai dengan persamaan 2.43.

$$\begin{aligned} \alpha_t &= \alpha_0 * \frac{1}{1 + d * i} \\ \alpha_t &= 0.01 * \frac{1}{1 + 0.0001 * 1} \\ \alpha_t &= 0.01 * 0.99990001 \\ \alpha_t &= 0.009999 \end{aligned}$$

5. Nilai *learning rate* yang akan dipakai pada iterasi selanjutnya adalah 0.009999.

6. Ulangi langkah ketiga hingga keempat sampai jumlah *epoch* atau iterasi yang ditentukan.

7. Simpan semua bobot, *bias*, lapisan dan *neuron* untuk digunakan pada tahap pengujian.

3.4.3.3 *Adagrad*

Berikut adalah contoh perhitungan menggunakan algoritma *Adagrad* pada *convolution neural network*:

1. Inisialisasi $\alpha = 0.01$, *initial accumulator* = 0.1 dan *epsilon* = 0.0000001. Misalkan bobot masukan adalah 0.05981104 dan asumsi perhitungan dari *loss function* mendapatkan nilai *gradient* g_t 0.934340141 (lihat perhitungan *loss* pada *forward propagation*).

2. Perhitungan bobot baru, secara matematika dapat dilihat pada persamaan 2.46.

$$w_t = w_0 - \alpha * g_t \quad (3.2)$$

$$\alpha_t = \frac{\alpha_0}{\sqrt{\sigma_t + \epsilon}} \quad (3.3)$$

$$\sigma_t = \sum_{i=1}^{\infty} \left(\frac{\partial L}{\partial w_i} \right)^2 \quad (3.4)$$

Keterangan :

w_t	: bobot baru	
w_0	: bobot lama	
α_t	: <i>learning rate</i> baru α_0	: <i>learning rate</i> lama
g_t	: <i>gradient</i>	
<i>sigma</i>	: total nilai <i>gradient</i> sebelumnya	

$$\begin{aligned}
 w_t &= w_0 - \alpha * g \\
 w_t &= 0.05981104 - 0.01 * 0.934340141 \\
 m_t &= 0.05981104 - 0.009343401 \\
 m_t &= 0.050467639
 \end{aligned}$$

3. Nilai bobot sebelumnya adalah 0.05981104 dan diperbarui dengan nilai bobot baru 0.050467639 dalam sekali iterasi pada *neuron* pertama di lapisan keluaran.
4. *learning rate* di setiap iterasinya pun mengalami perubahan sesuai dengan persamaan 3.3. Nilai *sigma* adalah nilai *initial accumulator* dari fungsi yaitu 0.1.

$$\begin{aligned}
 \alpha_t &= \frac{\alpha_0}{\sqrt{\sigma_t + \epsilon}} \\
 \alpha_t &= \frac{0.01}{\sqrt{0.1 + 0.0000001}} \\
 \alpha_t &= 0,031622761
 \end{aligned}$$

5. Nilai *learning rate* yang akan dipakai pada iterasi selanjutnya adalah 0.031622761 dan nilai dari *sigma* akan berubah dengan nilai *sigma* yang sekarang ditambah dengan nilai *gradient* iterasi ini.
6. Ulangi langkah kedua hingga kelima sampai jumlah *epoch* atau iterasi yang ditentukan.
7. Simpan semua bobot, *bias*, lapisan dan *neuron* untuk digunakan pada tahap pengujian.

3.4.3.4 Adadelta

Berikut adalah contoh perhitungan menggunakan algoritma *Adadelta* pada *convolution neural network*:

1. Inisialisasi $\alpha = 0.1$, $\rho = 0.9$ dan $\epsilon = 0.0000001$. Misalkan bobot masukan adalah 0.05981104 dan asumsi perhitungan dari *loss function* mendapatkan nilai

gradient g_t 0.934340141 (lihat perhitungan *loss* pada *forward propagation*).

2. Perhitungan bobot baru, secara matematika dapat dilihat pada persamaan 2.54. Berdasarkan persamaan 2.48, maka nilai dari *exponential weighted average* adalah

$$\begin{aligned} E[g^2]_t &= \gamma E[g^2]_{t-1} + (1 - \gamma) g_t^2 \\ E[g^2]_t &= 0.9 * 0 + (1 - 0.9) * 0.934340141^2 \\ E[g^2]_t &= 0.1 * 0.872991497 \\ E[g^2]_t &= 0.0872991497 \end{aligned}$$

Selanjutnya berdasarkan persamaan 2.50 maka perubahan nilai bobotnya adalah sebagai berikut

$$\begin{aligned} w_t &= w_0 - \frac{\alpha}{RMS[g]_t} g_t \\ w_t &= 0.05981104 - \frac{0.1}{\sqrt{0.0872991497 + 0.0000001}} * 0.93434014 \\ w_t &= 0.05981104 - 0.327150256 \\ w_t &= -0.267339216 \end{aligned}$$

3. Nilai bobot sebelumnya adalah 0.05981104 dan diperbarui dengan nilai bobot baru - 0.267339216 dalam sekali iterasi pada *neuron* pertama di lapisan keluaran. Namun nilai perubahan dari bobot ini yaitu - 0.327150256 nilainya disimpan dan akan diakumulasikan berdasarkan persamaan 2.52 yang nantinya dipakai untuk perubahan bobot selanjutnya pada persamaan 2.54.
4. *learning rate* di setiap iterasinya pun mengalami perubahan sesuai dengan persamaan 2.50.

$$\alpha_t = \frac{\alpha_0}{\sqrt{E[g^2]} + \epsilon}$$

$$\alpha_t = \frac{0.1}{\sqrt{0.0872991497} + 0.0000001}$$

$$\alpha_t = 0.338450176$$

5. Nilai *learning rate* yang akan dipakai pada iterasi selanjutnya adalah 0.338450176.
6. Ulangi langkah kedua hingga kelima sampai jumlah *epoch* atau iterasi yang ditentukan.
7. Simpan semua bobot, *bias*, lapisan dan *neuron* untuk digunakan pada tahap pengujian.

3.4.3.5 Adam

Berikut adalah contoh perhitungan menggunakan algoritma *Adam* pada *convolutional neural network*:

1. Inisialisasi $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ dan inisialisasi m_0 , v_0 , dan t dengan nol.
2. Nilai acak bobot pada iterasi *forward propagation* adalah 0.05981104.
3. Tambahkan *timestep* (t) dengan 1 ($t = 0 + 1 = 1$). Asumsi perhitungan dari *loss function* mendapatkan nilai *gradient* g_t 0.934340141 (lihat perhitungan *loss* pada *forward propagation*).
4. Perhitungan estimasi momen pertama terhadap bobot yang menerima masukan nilai *gradient* pada lapisan keluaran di *neuron* pertama dengan Persamaan 2.57.

$$m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$$

$$= 0.9 * 0 + (1 - 0.9) * 0.934340141$$

$$= 0 + 0.1 * 0.934340141$$

$$= 0.0934340141$$

5. Perhitungan Adam pada estimasi momen kedua terhadap bobot yang menerima masukan nilai *gradient* yang sama seperti langkah ketiga yaitu 0.934340141 dengan Persamaan 2.58.

$$\begin{aligned}
 v_t &= \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2 \\
 &= 0.999 * 0 + (1 - 0.999) * (0.934340141)^2 \\
 &= 0 + 0.001 * (0.934340141)^2 \\
 &= 0.000872991
 \end{aligned}$$

6. Menghitung nilai koreksi bobot pada estimasi momen pertama dengan Persamaan 2.62.

$$\begin{aligned}
 \hat{m}_t &= \frac{m_t}{1 - \beta_1^t} \\
 &= \frac{0.0934340141}{1 - 0.9} \\
 &= 0.934340141
 \end{aligned}$$

7. Menghitung nilai koreksi bobot pada estimasi momen kedua dengan Persamaan 2.64.

$$\begin{aligned}
 \hat{v}_t &= \frac{v_t}{1 - \beta_2^t} \\
 &= \frac{0.000872991}{1 - 0.999} \\
 &= 0.872991
 \end{aligned}$$

8. Pada tahap ini, nilai bobot yang lama akan diperbarui dengan nilai bobot yang baru dengan Persamaan 2.60. Nilai *w* yang merupakan parameter untuk bobot telah diinisialisasi dengan nilai 0.05981104. *Learning rate* (α) yang digunakan pada contoh perhitungan ini adalah 0.001 (10^{-3}), sedangkan nilai *epsilon* yang digunakan pada contoh perhitungan ini (ϵ) adalah 10^{-7} .

$$\begin{aligned}
 w_t &= w_{t-1} - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \\
 &= 0.05981104 - 10^{-3} * \frac{0.934340141}{\sqrt{0.872991} + 10^{-7}} \\
 &= 0.05981104 - 10^{-3} * \frac{0.934340141}{0.934339874 + 10^{-7}} \\
 &= 0.05981104 - 10^{-3} * \frac{0.934340141}{0.934339974} \\
 &= 0.05981104 - 10^{-3} * 1.000000179 \\
 &= 0.05981104 - 0.001 \\
 &= 0.05881104
 \end{aligned}$$

9. Nilai bobot sebelumnya adalah 0.05981104 dan diperbarui dengan nilai bobot baru 0.05881104 dalam sekali iterasi pada *neuron* pertama di lapisan keluaran.
10. Ulangi langkah ketiga hingga kedelapan sampai jumlah *epoch* atau iterasi yang ditentukan.
11. Simpan semua bobot, *bias*, lapisan dan *neuron* untuk digunakan pada tahap pengujian.

BAB 4 IMPLEMENTASI DAN PENGUJIAN

Bab ini menjelaskan proses implementasi dan pengujian terhadap sistem yang telah dibangun berdasarkan penjelasan pada bab sebelumnya. Hal yang akan dibahas adalah lingkungan implementasi sistem, implementasi perangkat lunak, proses dan hasil pengujian sistem, serta analisis kesalahan.

4.1 Lingkungan Implementasi

Bagian ini membahas perangkat yang digunakan dalam proses pembangunan sistem. Perangkat yang digunakan terdiri dari perangkat keras dan perangkat lunak.

4.1.1 Spesifikasi Perangkat Keras

Spesifikasi perangkat keras yang digunakan dalam pembangunan aplikasi adalah sebagai berikut.

1. *Laptop DELL Inspiron 15 3000 series.*
2. *Processor Intel Core i5-4210U CPU @ 1.7GHz.*
3. *Solid State Drive kapasitas 512GB.*
4. *RAM dengan kapasitas 8GB.*
5. *VGA Intel Corporation Haswell-ULT Integrated Graphics Controller.*

4.1.2 Lingkungan Perangkat Lunak

Lingkungan perangkat lunak yang digunakan dalam pembangunan aplikasi adalah sebagai berikut.

1. Sistem Operasi Linux Ubuntu 20.04.3 LTS
2. IDE: Google Colaboratory , Jupyter Lab
3. *Development Tools:* Anaconda3 2020.07 (Python 3.8.10 64-bit).

4.2 Implementasi Perangkat Lunak

Bagian ini membahas implementasi perangkat lunak dari sistem. Hal yang akan dibahas antara lain daftar *class* dan *method*, penggunaan Google Colaboratory, penggunaan Jupyter Lab dan penggunaan *dataset*.

4.2.1 Penggunaan Jupyter Lab

Penelitian ini menggunakan Jupyter Lab sebagai *tools* untuk melakukan *preprocessing* gambar. Jupyter Lab diakses melalui *development tools* yaitu

Anaconda 3 dan pemrosesan dilakukan pada *local machine*. Metode-metode yang digunakan pada Jupyter Lab terdaftar pada tabel 4.1.

Tabel 4.1 Daftar *method* pada *Jupyter lab*

No.	Method	Masukan	Keluaran	Keterangan
1	convert_to_lab	img: numpy array	l : integer a : integer b : integer	Mengkonversi <i>channel</i> pada gambar masukan menjadi <i>lab</i> , lalu mengembalikan nilai untuk masing-masing <i>attribute</i> pada <i>lab channel</i> .
2	m_clahe_equalize	l: integer a: integer b: integer img_name : string clipLimit: float	-	Mengaplikasikan algoritma <i>Modified Contrast Limited Adaptive Histogram Equalization</i> pada <i>channel L</i> pada masukan. Metode <i>CLAHE</i> diaplikasikan dengan <i>clip limit</i> sesuai dengan nilai masukan. Setelahnya <i>channel L</i> akan di- <i>merge</i> dengan <i>channel a</i> dan <i>b</i> . Selanjutnya gambar diaplikasikan <i>gaussian blur</i> . Tahap akhir pada metode ini adalah gambar yang sudah diproses di <i>save</i> ke <i>disk</i> .

4.2.2 Penggunaan Google Colaboratory

Google Colaboratory dinilai cocok untuk mengeksekusi algoritme pembelajaran mesin serta analisis data karena menyediakan *resource GPU* dengan performa tinggi. Google Colaboratory dipilih sebagai alternatif Jupyter Notebook pada *local machine* dengan pertimbangan proses pelatihan CNN akan memakan *resource* yang besar, dan jika dilakukan pada *local machine*, akan memperlambat proses

implementasi aplikasi.

4.2.2.1 Class VGG16

Kelas ini merupakan *blueprint* untuk instansiasi arsitektur *VGG16*. Adapun metode-metode pada kelas ini terdaftar pada tabel 4.2.

Tabel 4.2 Daftar *method* pada *class VGG16*

No.	Method	Masukan	Keluaran	Keterangan
1	build	width : integer height : integer depth: integer classes: integer initializers : keras.initializers	model : keras.models	Membangun model vgg16. Seperti pada gambar 2.21 dan 2.22, vgg16 memiliki 5 blok konvolusi sebelum memasuki <i>fully connected layer</i> . Pada metode ini model mengaplikasikan 5 blok konvolusi dengan <i>drop out layer</i> dan <i>batch normalization layer</i> pada akhir setiap blok konvolusi.

4.2.2.2 Class Inception

4.2.3 Implementasi Penggunaan Dataset

Seperti yang sudah dijelaskan pada bagian 3.4.1, *dataset* untuk penelitian ini diambil dari MIR-QBSH [?]. Dari seluruh data yang digunakan, data tersebut kemudian dibagi dengan komposisi 80% untuk proses pelatihan, dan 20% untuk proses pengujian. Tabel 4.3 menjelaskan perincian pembagian *dataset* untuk proses pelatihan dan pengujian. Pada proses pelatihan, dari 80% data yang sudah dibagi, diambil lagi 20% data untuk dijadikan validasi proses pelatihan. Validasi ini diperlukan untuk melihat perkembangan model saat menerima data baru, apakah sudah memenuhi *generalization* dengan baik atau cenderung *overfit* atau *underfit*.

Tabel 4.3 Perincian Penggunaan *Dataset* untuk Implementasi

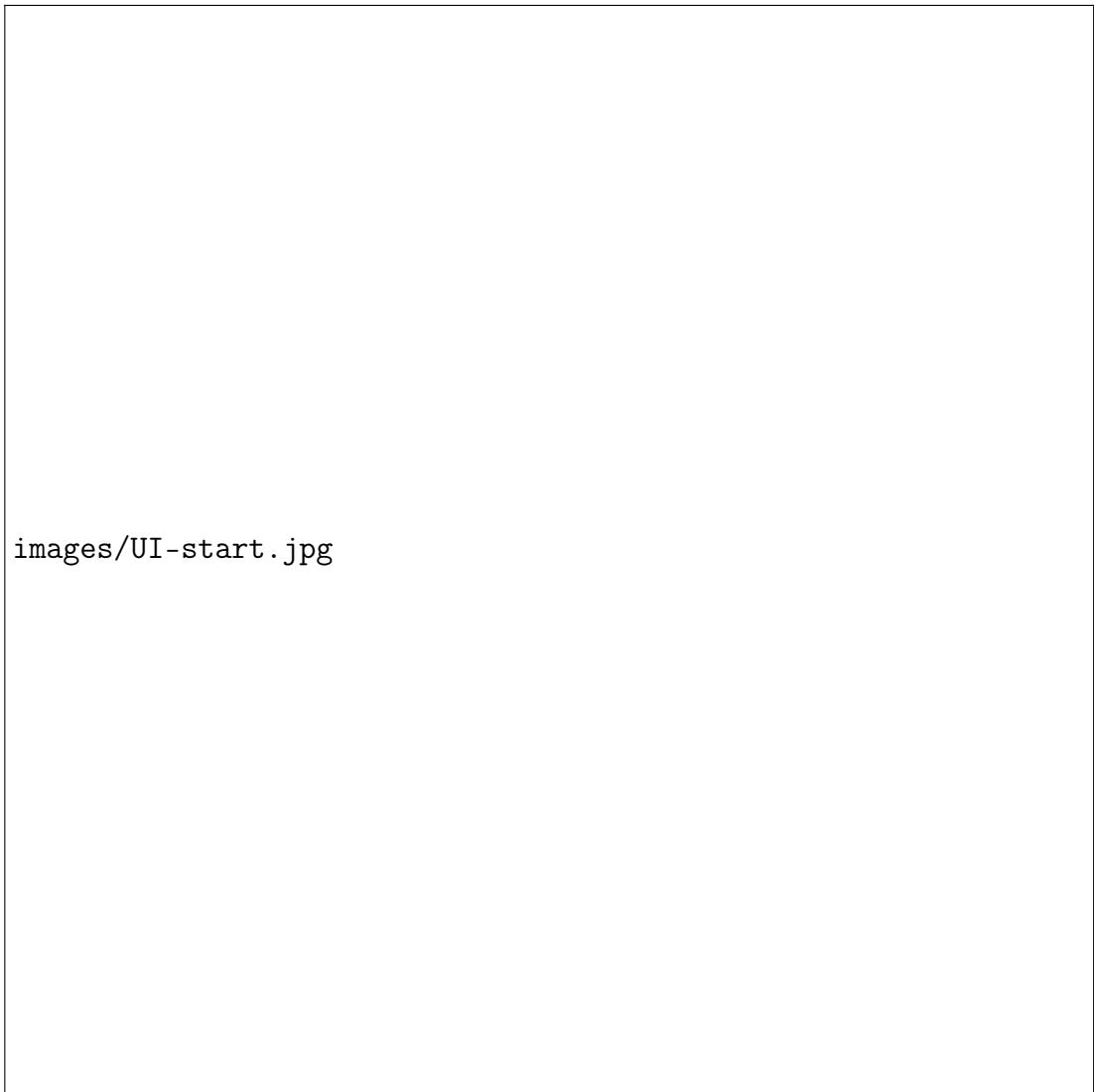
No	Kode-Judul Lagu	Total Data	Pembagian Data	
			Pelatihan	Pengujian
1	014 - Twinkle, Twinkle, Little Star	159	127	32
2	017 - Old MacDonald Had a Farm	152	121	31
3	020 - Happy Birthday	170	136	34
4	022 - Brother John (Are you sleeping?)	173	138	35
5	030 - London Bridge Is Falling Down	145	116	29

4.3 Implementasi Aplikasi

Bagian ini menjelaskan GUI (*graphical user interface*) yang digunakan untuk mempermudah akses pengguna terhadap aplikasi estimasi *pitch* dan pengujian kemiripan. Tampilan dengan GUI dibuat untuk proses pengujian, di mana pengguna dapat melakukan *upload* berkas rekaman suara nyanyiannya untuk pengujian estimasi *pitch* dan pengujian kemiripan. GUI tersebut digambarkan pada Gambar 4.1 hingga Gambar 4.7 berikut.

1. Gambar 4.1 merupakan tampilan awal ketika aplikasi diakses oleh pengguna. Pada halaman ini, pengguna diminta untuk memasukkan data judul lagu dan rekaman yang akan diuji. Selain itu, pengguna juga diminta memilih kombinasi dari arsitektur, *epoch*, dan *learning rate* untuk melakukan estimasi *pitch*. Hasil estimasi *pitch* dan *distance score* akan ditampilkan berdasarkan model dari kombinasi arsitektur, *epoch*, dan *learning rate* yang dipilih. Judul lagu diperlukan untuk menentukan sekuens s1 yang digunakan dalam pengujian kemiripan.

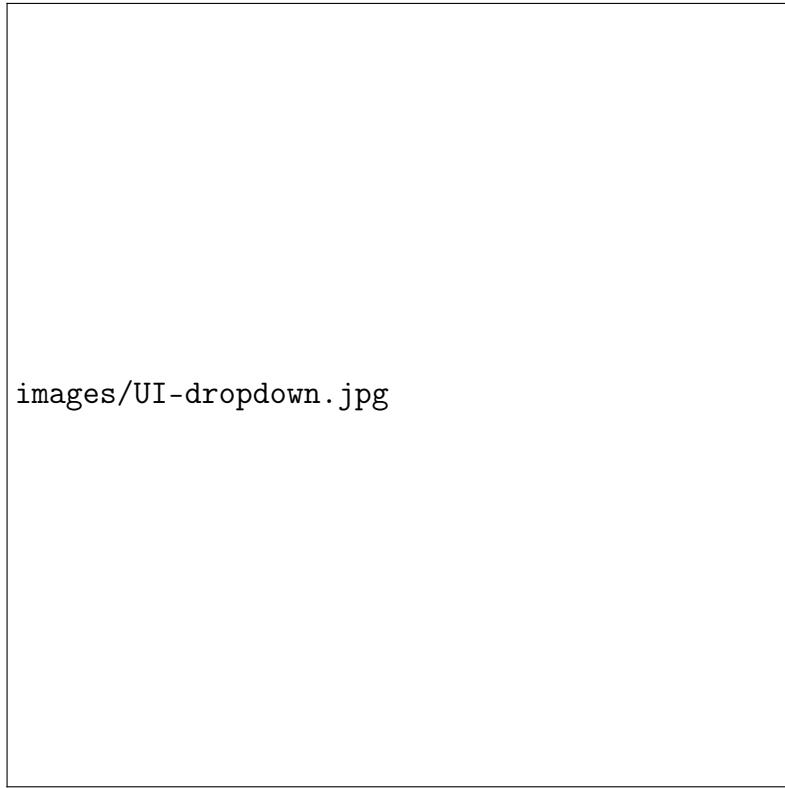
BAB 4 IMPLEMENTASI DAN PENGUJIAN



images/UI-start.jpg

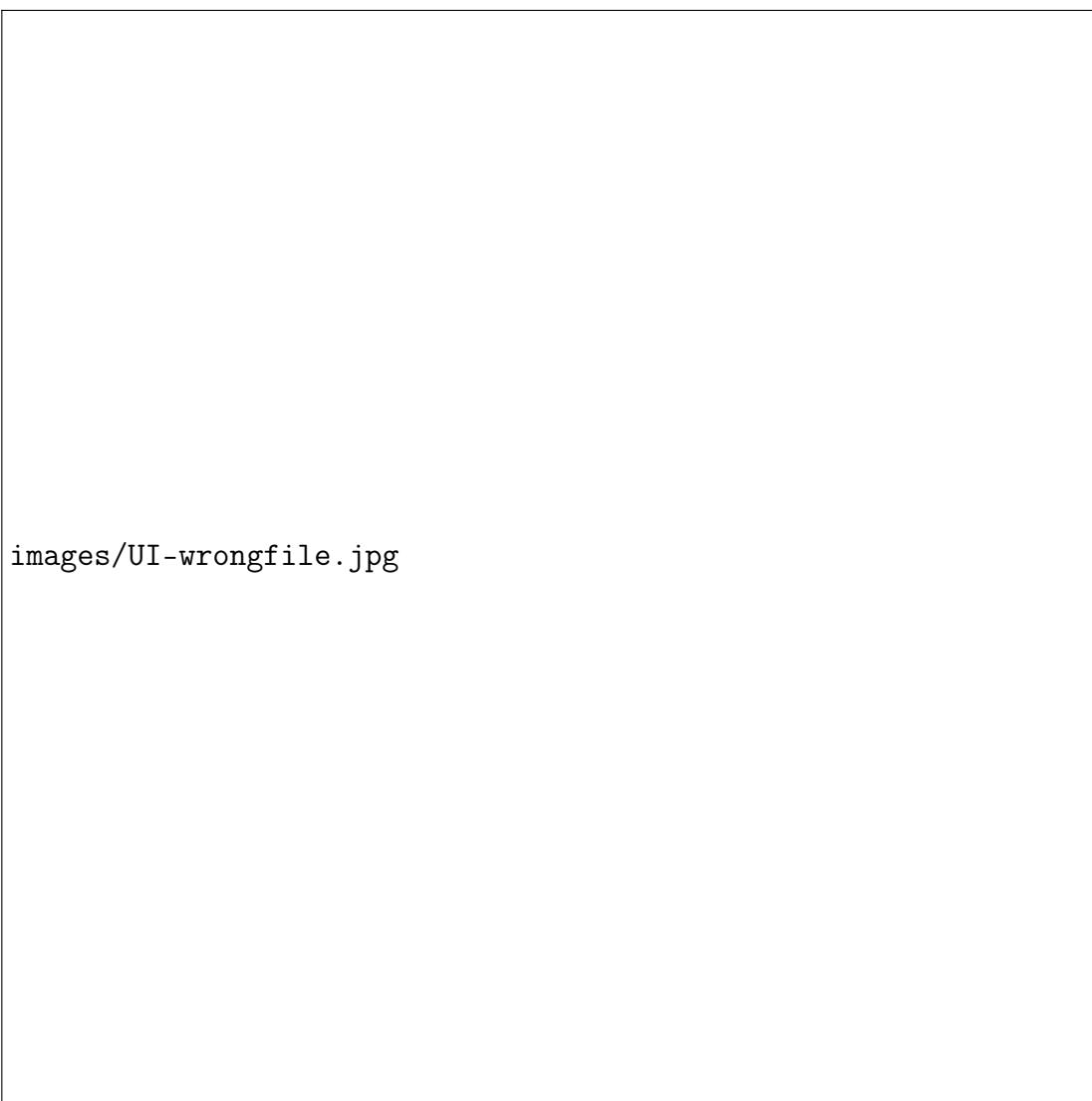
Gambar 4.1 Tampilan GUI awal aplikasi

2. Gambar 4.2 merupakan tampilan menu *dropdown* yang dapat digunakan pengguna untuk memilih judul lagu yang akan dinyanyikannya.



Gambar 4.2 Tampilan GUI untuk pemilihan judul lagu

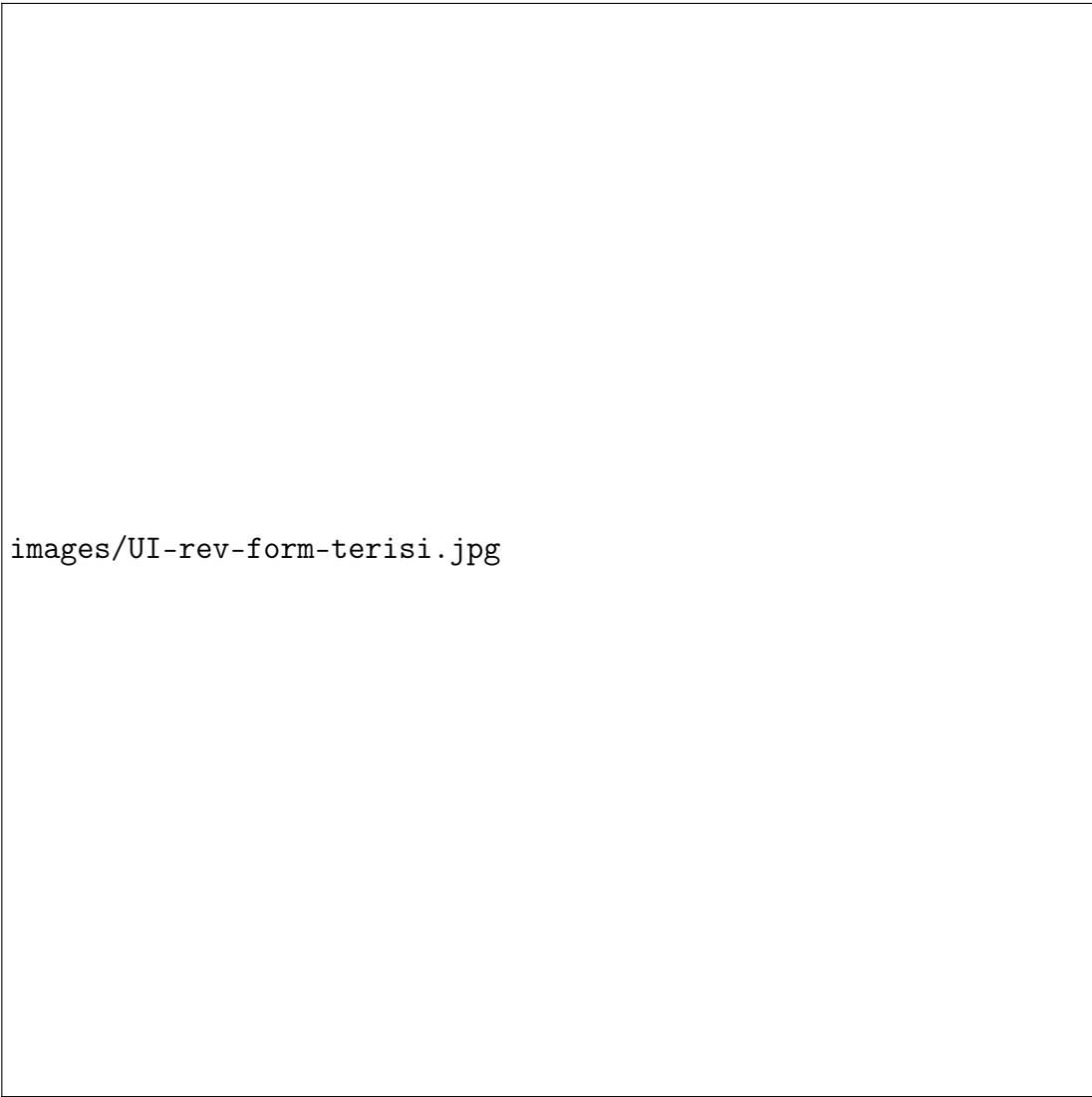
3. Gambar 4.3 menunjukkan *error handling* jika pengguna salah melakukan *upload* berkas rekaman. Aplikasi yang dibangun hanya dapat menerima berkas audio dengan ekstensi .wav, .mp3, atau .m4a saja. Setelah dimunculkan *alert*, pengguna diminta kembali untuk melakukan *upload* berkas rekaman.



images/UI-wrongfile.jpg

Gambar 4.3 Tampilan GUI ketika berkas yang diupload memiliki ekstensi yang salah

4. Gambar 4.4 menunjukkan judul lagu, berkas audio, dan pemilihan kombinasi arsitektur yang sudah berhasil dimasukkan oleh pengguna. Setelah itu, pengguna dapat memilih tombol *show test result* untuk melakukan pengujian terhadap rekamannya.

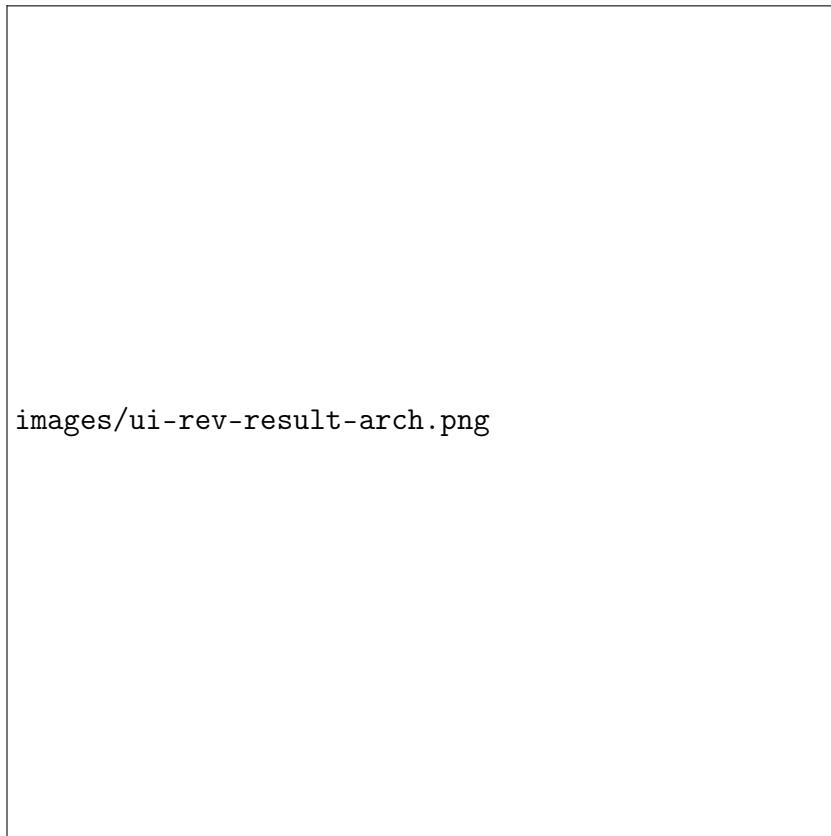


images/UI-rev-form-terisi.jpg

Gambar 4.4 Tampilan GUI yang berisi data masukan dari pengguna

5. Gambar 4.5 hingga Gambar 4.7 merupakan contoh hasil pengujian rekaman yang ditampilkan kepada pengguna. Ketika pengujian selesai dilakukan, hasil pengujian dari *back-end engine* akan diteruskan ke GUI agar dapat dilihat pengguna. Pada halaman ini, pengguna dapat melihat skor kemiripan yang ditunjukkan dalam *distance score*, sekuens *pitch* yang unik pada *template* di *dataset*, serta sekuens *pitch* yang terdeteksi pada rekaman pengguna (Gambar 4.6). Selain itu, dapat dilihat juga informasi nilai akurasi rata-rata dan nilai F1 Score rata-rata dari kombinasi arsitektur, *epoch*, dan *learning rate* yang telah dipilih sebelumnya (Gambar 4.5). Di bawah hasil pengujian, ada juga penjelasan singkat tentang pengujian dan kemiripan rekaman pengguna (Gambar 4.7). Teks yang dicetak tebal dan digarisbawahi akan menunjukkan *similar* jika rekaman dianggap mirip dan menunjukkan *not similar* jika rekaman dianggap tidak mirip.

BAB 4 IMPLEMENTASI DAN PENGUJIAN



Gambar 4.5 Tampilan GUI hasil pengujian: informasi nilai akurasi dan F1 *Score* rata-rata model

BAB 4 IMPLEMENTASI DAN PENGUJIAN



images/ui-rev-result-pitch-result.png

Gambar 4.6 Tampilan GUI hasil pengujian: hasil pengujian *pitch* dan *distance score*



images/ui-rev-result-explanation.png

Gambar 4.7 Tampilan GUI hasil pengujian: penjelasan singkat dari hasil pengujian yang dilakukan

4.4 Pengujian

Bagian ini menjelaskan pengujian terhadap sistem estimasi *pitch* dan pengujian kemiripan yang telah dibuat. Pengujian yang akan dilakukan adalah sebagai berikut.

4.4.1 Pengujian Nilai *Epoch*

Epoch diuji untuk mengetahui berapa banyak *epoch* yang optimal bagi arsitektur yang dibangun agar arsitektur tidak terlalu banyak belajar. Nilai *epoch* yang akan diuji adalah 50, 100, 150, 200, 250, dan 300.

4.4.2 Pengujian Nilai *Learning Rate*

Learning rate merupakan *hyperparameter* dari arsitektur CNN yang dapat memengaruhi akurasi sistem dalam melakukan estimasi *pitch*. Seperti yang sudah dijelaskan pada bagian ??, *learning rate* akan menentukan seberapa besar langkah yang akan diambil setiap iterasi pelatihan hingga mencapai lokal minimum.

Pengujian dilakukan untuk mengetahui nilai *learning rate* yang optimal untuk menghasilkan akurasi yang tinggi. Nilai *learning rate* yang akan diuji adalah 0.001 (10^{-3}), 0.0001 (10^{-4}), 0.0002 (2×10^{-4}), dan 0.00001 (10^{-5}).

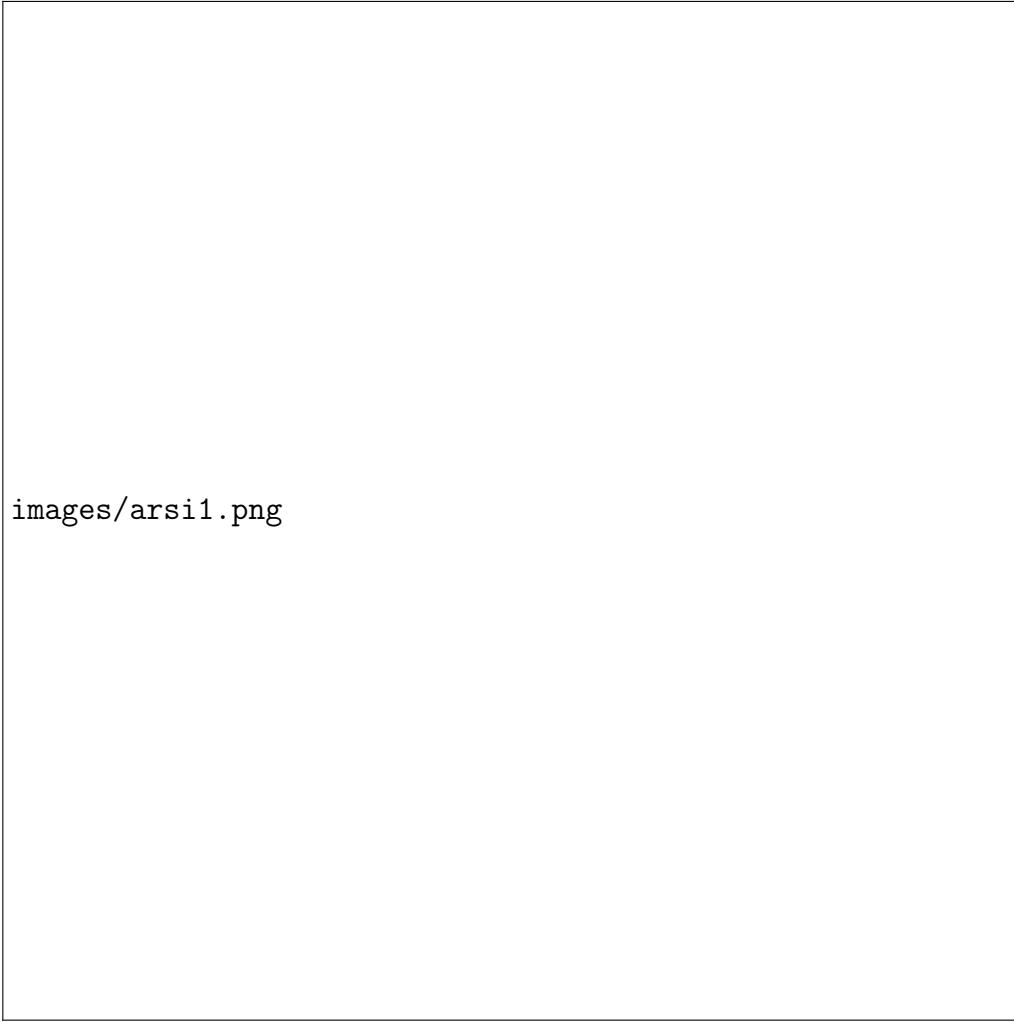
4.4.3 Pengujian Jumlah Konvolusi dan *Filter*

Jumlah *filter* yang digunakan dalam satu *convolutional layer* akan memengaruhi akurasi. Dengan jumlah *filter* yang tepat, fitur akan semakin banyak diekstrak dari data. *Filter* berperan sebagai penyimpan fitur yang penting bersamaan dengan jumlah konvolusi yang memengaruhi akurasi sistem mempelajari karakteristik data. Pengujian dilakukan untuk mengetahui pengaruh jumlah *convolutional layer* dan *filter* terhadap faktor-faktor tersebut.

Berikut ini adalah rancangan arsitektur konvolusi dan jumlah *filter* yang akan diuji. Ada empat buah rancangan arsitektur sebagai berikut.

1. Rancangan Arsitektur 1

Gambar 4.8 adalah ilustrasi dari arsitektur 1.



Gambar 4.8 Rancangan arsitektur CNN yang pertama

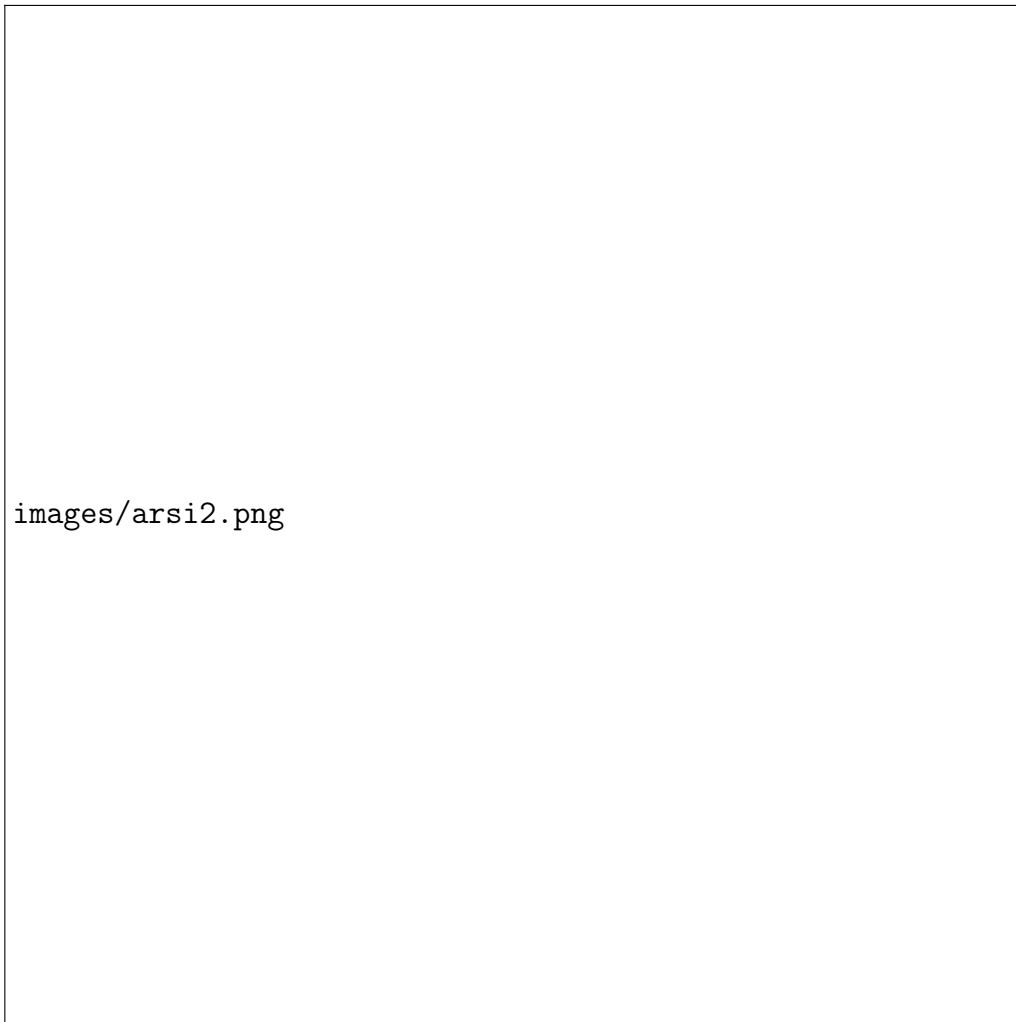
Berikut ini adalah urutan proses yang akan dilakukan pada arsitektur pertama berdasarkan Gambar 4.8.

- (a) 1 *layer* konvolusi dengan 1024 *filter*, fungsi aktivasi ReLU, menggunakan *padding*, ukuran *kernel* 2, *regularizer* L2 dengan nilai 0.001, *bias initializer* 0 dan *kernel initializer random normal*
- (b) *Batch Normalization*
- (c) 1 *layer* konvolusi dengan 512 *filter*, fungsi aktivasi ReLU, menggunakan *padding*, *regularizer* L2 dengan nilai 0.001
- (d) *Max Pooling* dengan *pool size* 2
- (e) *Batch Normalization*
- (f) 1 *layer* konvolusi dengan 1024 *filter*, fungsi aktivasi ReLU, menggunakan *padding*, *regularizer* L2 dengan nilai 0.001
- (g) *Max Pooling* dengan *pool size* 2
- (h) *Batch Normalization*

- (i) 1 *layer* konvolusi dengan 2048 *filter*, fungsi aktivasi ReLU, menggunakan *padding*, *regularizer* L2 dengan nilai 0.001
- (j) *Batch Normalization*
- (k) *Dropout* dengan nilai 0.5
- (l) 1 *dense layer*, fungsi aktivasi Softmax, memprediksi 51 kelas, dengan *kernel regularizer* L2 nilai 0.001

2. Rancangan Arsitektur 2

Gambar 4.9 adalah ilustrasi dari arsitektur 2.



Gambar 4.9 Rancangan arsitektur CNN yang kedua

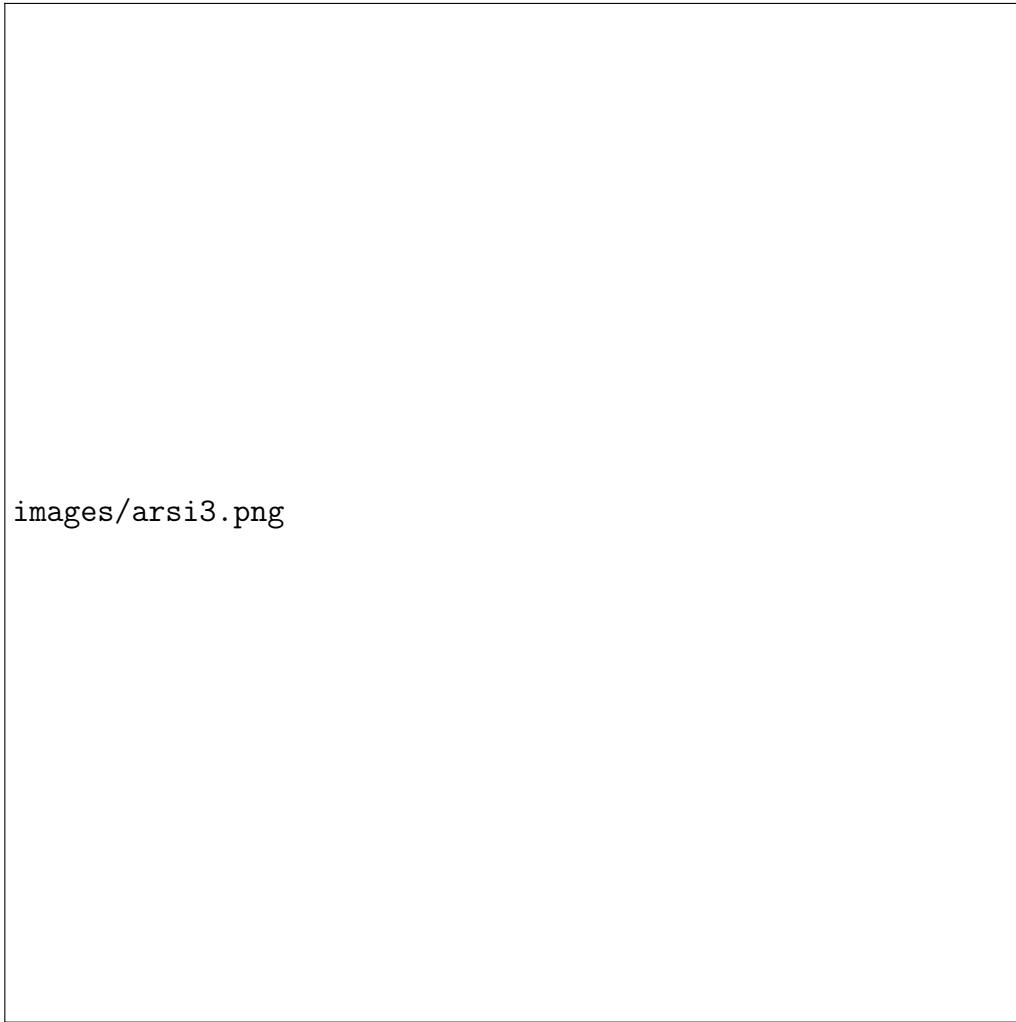
Berikut ini adalah urutan proses yang akan dilakukan pada arsitektur pertama berdasarkan Gambar 4.9.

- (a) 1 *layer* konvolusi dengan 1024 *filter*, fungsi aktivasi ReLU, menggunakan *padding*, ukuran *kernel* 2, *regularizer* L2 dengan nilai 0.001, *bias initializer* 0 dan *kernel initializer random normal*

- (b) *Batch Normalization*
- (c) 1 *layer* konvolusi dengan 128 *filter*, fungsi aktivasi ReLU, menggunakan *padding, regularizer L2* dengan nilai 0.001
- (d) *Max Pooling* dengan *pool size 2*
- (e) *Batch Normalization*
- (f) 1 *layer* konvolusi dengan 256 *filter*, fungsi aktivasi ReLU, menggunakan *padding, regularizer L2* dengan nilai 0.001
- (g) *Max Pooling* dengan *pool size 2*
- (h) *Batch Normalization*
- (i) 1 *layer* konvolusi dengan 512 *filter*, fungsi aktivasi ReLU, menggunakan *padding, regularizer L2* dengan nilai 0.001
- (j) *Batch Normalization*
- (k) *Dropout* dengan nilai 0.5
- (l) 1 *dense layer*, fungsi aktivasi Softmax, memprediksi 51 kelas, dengan *kernel regularizer L2* nilai 0.001

3. Rancangan Arsitektur 3

Gambar [4.10](#) adalah ilustrasi dari arsitektur 3.



Gambar 4.10 Rancangan arsitektur CNN yang ketiga

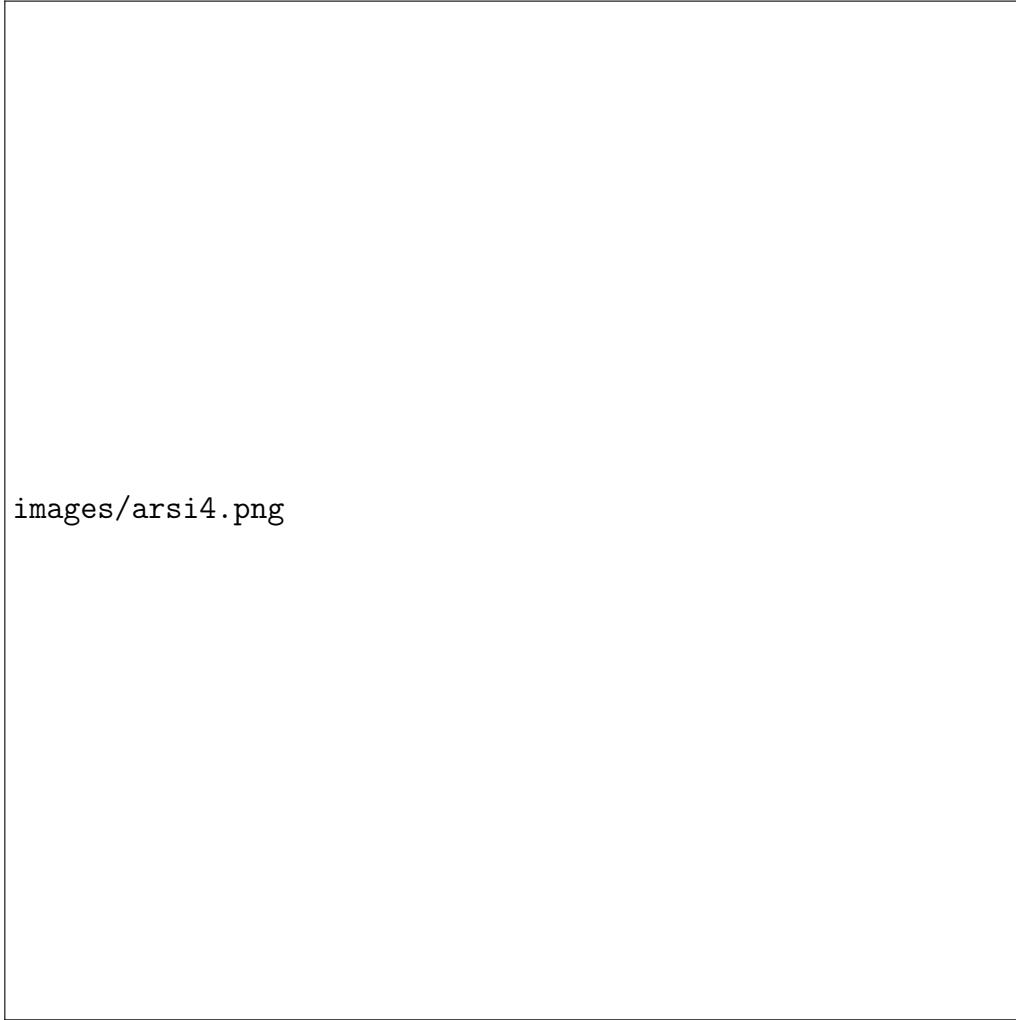
Berikut ini adalah urutan proses yang akan dilakukan pada arsitektur pertama berdasarkan Gambar 4.10.

- (a) 1 *layer* konvolusi dengan 1024 *filter*, fungsi aktivasi ReLU, menggunakan *padding*, ukuran *kernel* 2, *regularizer* L2 dengan nilai 0.001, *bias initializer* 0 dan *kernel initializer random normal*
- (b) *Batch Normalization*
- (c) 1 *layer* konvolusi dengan 1024 *filter*, fungsi aktivasi ReLU, menggunakan *padding*, *regularizer* L2 dengan nilai 0.001
- (d) *Max Pooling* dengan *pool size* 2
- (e) *Batch Normalization*
- (f) 1 *layer* konvolusi dengan 1024 *filter*, fungsi aktivasi ReLU, menggunakan *padding*, *regularizer* L2 dengan nilai 0.001
- (g) *Max Pooling* dengan *pool size* 2
- (h) *Dropout* dengan nilai 0.5

- (i) 1 *dense layer*, fungsi aktivasi Softmax, memprediksi 51 kelas, dengan *kernel regularizer L2* nilai 0.001

4. Rancangan Arsitektur 4

Gambar 4.11 adalah ilustrasi dari arsitektur 4.



Gambar 4.11 Rancangan arsitektur CNN yang keempat

Berikut ini adalah urutan proses yang akan dilakukan pada arsitektur pertama berdasarkan Gambar 4.11.

- (a) 1 *layer* konvolusi dengan 512 *filter*, fungsi aktivasi ReLU, menggunakan *padding*, ukuran *kernel* 2, *regularizer L2* dengan nilai 0.001, *bias initializer* 0 dan *kernel initializer random normal*
- (b) *Batch Normalization*
- (c) 1 *layer* konvolusi dengan 512 *filter*, fungsi aktivasi ReLU, menggunakan *padding*, *regularizer L2* dengan nilai 0.001
- (d) *Max Pooling* dengan *pool size* 2

- (e) *Batch Normalization*
- (f) 1 *layer* konvolusi dengan 512 *filter*, fungsi aktivasi ReLU, menggunakan *padding*, *regularizer* L2 dengan nilai 0.001
- (g) *Max Pooling* dengan *pool size* 2
- (h) *Dropout* dengan nilai 0.5
- (i) 1 *dense layer*, fungsi aktivasi Softmax, memprediksi 51 kelas, dengan *kernel regularizer* L2 nilai 0.001

4.4.4 Pengujian Nilai *Threshold*

Setelah didapatkan jumlah *epoch*, *learning rate*, dan arsitektur (konvolusi dan *filter*) yang tepat dengan akurasi terbaik dari seluruh hasil pengujian, barulah dilakukan pengujian untuk mengetahui nilai *threshold* yang optimal untuk menganalisis kemiripan rekaman lagu masukan dengan *template* dari dataset. Nilai *threshold* yang akan dianalisis adalah 500, 1000, 1500, 2000, dan 2500.

4.5 Hasil Pengujian

Berdasarkan pemaparan yang sudah dijelaskan pada bagian 4.4 , berikut ini akan dituliskan seluruh hasil pengujian yang sudah dilakukan. Pengujian *epoch*, *learning rate*, jumlah konvolusi dan *filter* dilakukan dengan mengombinasikan semua nilai. Hasil pengujian ini dikelompokkan berdasarkan arsitektur, di mana pada setiap arsitektur, diujikan setiap kombinasi nilai *epoch* dan *learning rate* untuk mencari nilai akurasi yang terbaik.

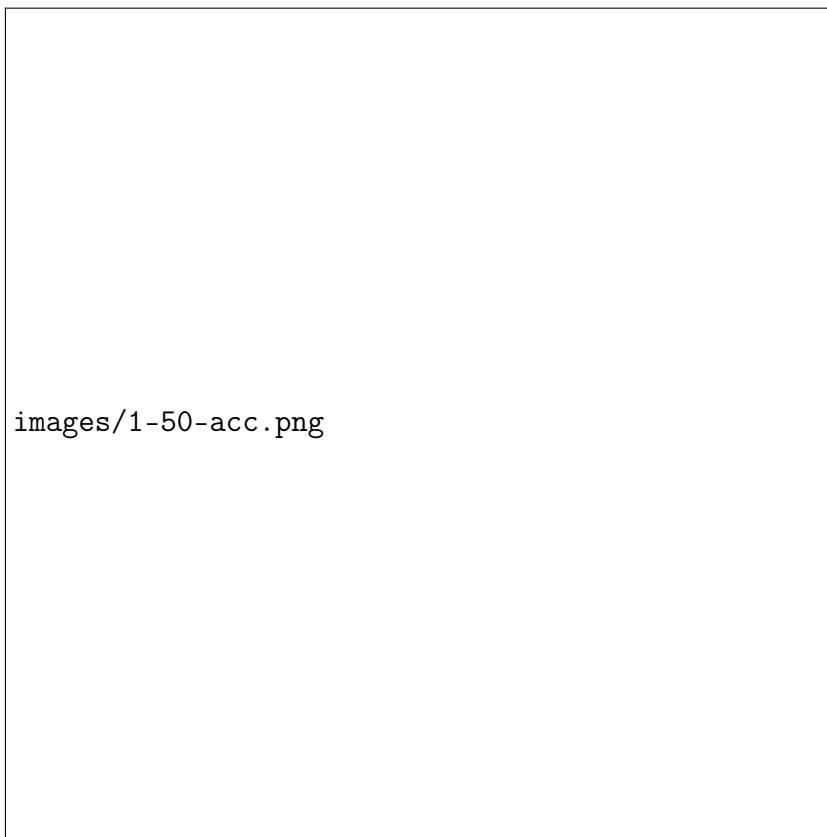
Gambar grafik (Gambar 4.12 hingga Gambar 4.71) yang akan dimunculkan pada bagian 4.5.1.1 hingga bagian 4.5.4.6 merupakan grafik hasil pengujian. Seluruh grafik tersebut menunjukkan laju pergerakan akurasi selama proses pengujian per *epoch* untuk data latih dan data validasi. Data validasi diambil dari 20% data pelatihan. Data validasi digunakan untuk melihat pengaruh data baru yang dimasukkan terhadap proses pelatihan yang sedang dijalankan, seperti yang sudah dijelaskan pada bagian 4.2.3. Hal ini dilakukan untuk mengambil keputusan apakah model yang dibangun mengalami *overfitting* atau *underfitting*. Model dinyatakan *overfit* jika selisih akurasi dan akurasi validasi terlalu jauh, sedangkan model dinyatakan *underfit* jika akurasi terlalu rendah. Pada grafik, sumbu x menunjukkan nilai *epoch*, sedangkan sumbu y menunjukkan nilai *learning rate* (pada grafik, penulisannya disingkat menjadi LR).

4.5.1 Pengujian pada Arsitektur 1

Bagian ini akan merangkum hasil pengujian pada arsitektur pertama. Penjelasan dibagi per *epoch* dan *learning rate*. Tabel yang merangkum hasil pengujian dapat dilihat pada Tabel 4.4 dan 4.5 di bagian 4.5.1.7.

4.5.1.1 Nilai *Epoch* = 50

Grafik pengujian dengan nilai *epoch* 50 untuk masing-masing nilai *learning rate* dapat dilihat pada Gambar 4.12 untuk akurasi pengujian dan Gambar 4.13 untuk akurasi validasi pengujian. Dapat dilihat pada kedua gambar berikut, nilai *learning rate* memiliki pengaruh kenaikan akurasi sekaligus memengaruhi kestabilan model. Nilai *learning rate* yang tepat akan menghasilkan grafik yang stabil.



Gambar 4.12 Grafik perubahan akurasi pada pengujian arsitektur 1 untuk *epoch* = 50

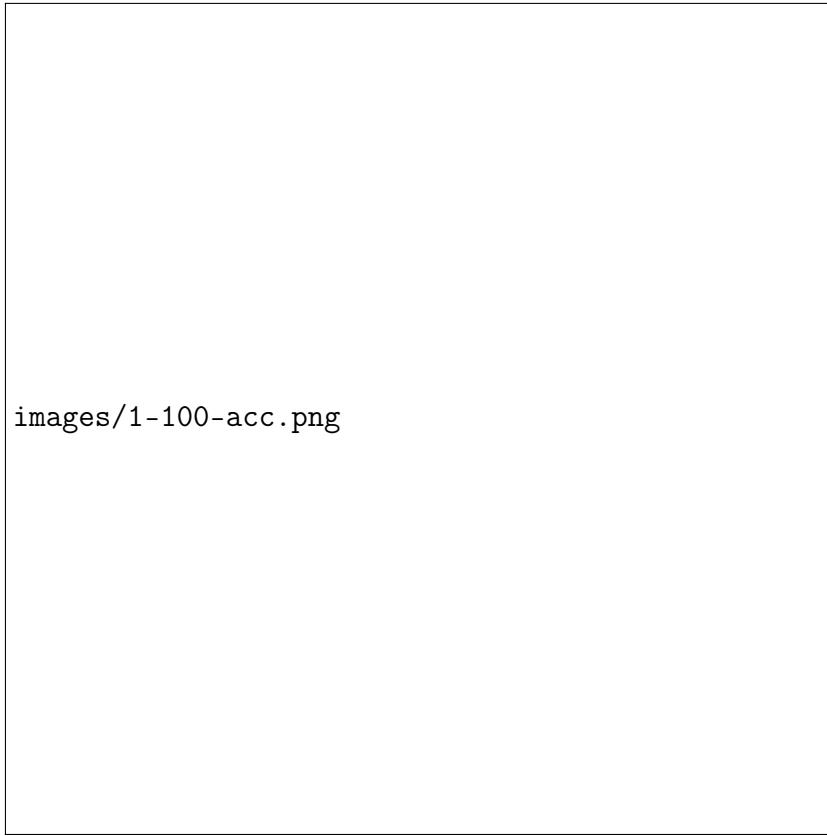


Gambar 4.13 Grafik perubahan akurasi validasi pengujian pada arsitektur 1 untuk $epoch = 50$

4.5.1.2 Nilai *Epoch* = 100

Gambar 4.14 dan Gambar 4.15 merupakan hasil pengujian pada arsitektur pertama dengan nilai *epoch* 100. Hasil pengujian menunjukkan bahwa nilai *learning rate* sangat berpengaruh. Hal ini ditunjukkan dari pergerakan grafik pada masing-masing nilai parameter yang diuji. Nilai *learning rate* yang baik untuk model tidak akan menghasilkan bentuk grafik yang tidak stabil. Hal ini dapat dilihat dengan jelas dari pergerakan akurasi validasi. *Learning rate* mengontrol seberapa cepat model beradaptasi dengan masalah, sehingga harus ditemukan nilai yang tepat dan optimal untuk melakukan estimasi.

BAB 4 IMPLEMENTASI DAN PENGUJIAN



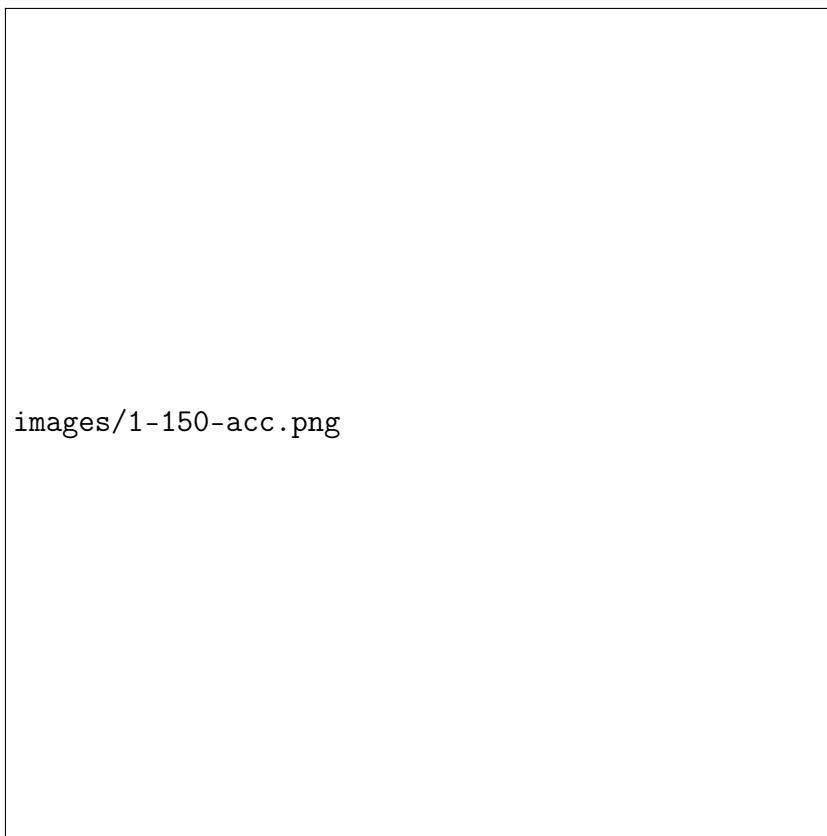
Gambar 4.14 Grafik perubahan akurasi pada pengujian arsitektur 1 untuk $epoch = 100$



Gambar 4.15 Grafik perubahan akurasi validasi pada pengujian arsitektur 1 untuk $epoch = 100$

4.5.1.3 Nilai *Epoch* = 150

Hasil pengujian dengan nilai *epoch* 150 pada arsitektur pertama dapat dilihat pada Gambar 4.16 dan Gambar 4.17. Bukti bahwa nilai *learning rate* sangat berpengaruh pada laju pergerakan akurasi semakin terlihat pada pengujian ini. Contohnya, dengan *learning rate* 0.001, meskipun grafik akurasi terlihat stabil, grafik validasi terlihat sangat tidak stabil. Grafik yang tidak stabil ini menunjukkan bahwa model melakukan estimasi secara acak (*random*) pada data yang baru dimasukkan kepadanya, sehingga tidak dapat memberikan hasil estimasi yang tepat. Selain itu, nilai *learning rate* yang terlalu besar juga berpengaruh kepada akurasi karena akan membuat model terlalu cepat mencapai konvergensi ke solusi yang sebenarnya kurang optimal. Jika konvergensi dicapai terlalu cepat, model akan kehilangan kemampuan mengenali data baru.



Gambar 4.16 Grafik perubahan akurasi pada pengujian arsitektur 1 untuk *epoch* = 150



images/1-150-valacc.png

Gambar 4.17 Grafik perubahan akurasi validasi pada pengujian arsitektur 1 untuk $epoch = 150$

4.5.1.4 Nilai $Epoch = 200$

Gambar 4.18 dan Gambar 4.19 menunjukkan hasil pengujian pada arsitektur pertama dengan nilai $epoch$ 200. Selain adanya pengaruh dari *learning rate*, pengaruh nilai $epoch$ mulai terlihat juga pada pengujian ini. Jika nilai $epoch$ terlalu banyak, laju pergerakan grafik akurasi dengan data latih pun akan mengalami ketidakstabilan. Hal ini dapat terjadi karena model terlalu banyak belajar menangani data, sehingga model kurang tepat melakukan estimasi karena banyaknya kemungkinan yang dipelajari oleh model. Nilai $epoch$ ini dapat dinyatakan tidak cocok untuk *learning rate* 0.0002 dan 0.001.

BAB 4 IMPLEMENTASI DAN PENGUJIAN



Gambar 4.18 Grafik perubahan akurasi pada pengujian arsitektur 1 untuk $epoch = 200$



Gambar 4.19 Grafik perubahan akurasi validasi pada pengujian arsitektur 1 untuk $epoch = 200$

4.5.1.5 Nilai *Epoch* = 250

Di bawah ini, ditunjukkan hasil pengujian dengan nilai *epoch* 250 pada arsitektur pertama dengan Gambar 4.20 dan Gambar 4.21. Sejauh ini, dapat diambil kesimpulan bahwa nilai *learning rate* 0.001 dan 0.0002 tidak cocok digunakan untuk model arsitektur pertama karena meskipun menghasilkan akurasi yang baik, model mengalami *overfit* dan tidak stabil ketika dimasukkan data yang baru. Nilai *epoch* ini juga tidak dapat dikatakan optimal jika dikombinasikan dengan kedua nilai *learning rate* tersebut. Pada grafik akurasi, mulai terlihat ketidakstabilan model yang menandakan sudah terlalu banyak data yang diberikan kepada model. Grafik akurasi validasi juga menegaskan performa model yang tidak baik untuk nilai *epoch* dan *learning rate* tersebut.



images/1-250-acc.png

Gambar 4.20 Grafik perubahan akurasi pada pengujian arsitektur 1 untuk *epoch* = 250



images/1-250-valacc.png

Gambar 4.21 Grafik perubahan akurasi validasi pada pengujian arsitektur 1 untuk $epoch = 250$

4.5.1.6 Nilai $Epoch = 300$

Hasil pengujian dengan nilai $epoch$ 300 pada arsitektur pertama dapat dilihat pada Gambar 4.22 dan Gambar 4.23. Sebelumnya, telah dinyatakan bahwa nilai *learning rate* dan nilai $epoch$ memiliki pengaruh terhadap laju pergerakan grafik akurasi dan akurasi validasi. Ketika nilai $epoch$ optimal, grafik akan mulai berbentuk landai, tidak lagi menukik ke atas atau bawah secara ekstrem. Jika nilai $epoch$ sudah terlalu banyak, hal ini menyebabkan ketidakstabilan model karena model akan terlalu banyak belajar mengenali data. Jika nilai *learning rate* optimal, model akan membentuk grafik akurasi yang stabil, khususnya pada akurasi validasi, tidak cenderung naik dan turun secara fluktuatif dan tidak beraturan. Dengan nilai $epoch$ 300, dapat dinyatakan bahwa model optimal dengan *learning rate* $0.00001 (10^{-5})$. Grafik akurasi masih menunjukkan kenaikan, sedangkan grafik validasi sudah menunjukkan kestabilan.

BAB 4 IMPLEMENTASI DAN PENGUJIAN



Gambar 4.22 Grafik perubahan akurasi pada pengujian arsitektur 1 untuk $epoch = 300$



Gambar 4.23 Grafik perubahan akurasi validasi pada pengujian arsitektur 1 untuk $epoch = 300$

4.5.1.7 Tabel Hasil Pengujian pada Arsitektur 1

Tabel 4.4 dan 4.5 merangkum seluruh hasil pengujian estimasi *pitch* yang sudah dilakukan pada arsitektur pertama. Kedua tabel masing-masing berisi nilai *mean* (rata-rata) dan nilai maksimum dari setiap parameter percobaan dari setiap *epoch* yang sudah dijalankan. Pengujian menghasilkan nilai dalam bentuk bilangan desimal. Tabel yang menunjukkan seluruh hasil pengujian untuk setiap *epoch* dapat dilihat pada Lampiran B. Pada Tabel 4.4 dan 4.5 serta tabel hasil pengujian di arsitektur kedua hingga keempat (Tabel 4.6 hingga 4.11), nilai desimal tersebut dikonversi ke dalam bentuk persentase. Kesimpulan yang diambil dari Tabel 4.4 dan 4.5 dapat dilihat setelah Tabel 4.5.

Tabel 4.4 Hasil Pengujian (nilai rata-rata) pada Arsitektur 1

<i>Learning Rate</i>	<i>Epoch</i>	Hasil Nilai Rata-rata			
		<i>Accuracy</i>	<i>F1 Score</i>	<i>Val Accuracy</i>	Δ <i>Accuracy</i>
0.001 (10^{-3})	50	86.175%	86.190%	39.034%	47.141%
	100	87.060%	87.063%	35.658%	51.402%
	150	88.897%	88.908%	39.219%	49.679%
	200	90.670%	90.675%	45.917%	44.752%
	250	90.687%	90.688%	47.475%	43.212%
	300	92.433%	92.434%	50.031%	42.402%
0.0001 (10^{-4})	50	88.197%	88.079%	79.983%	8.214%
	100	93.208%	93.143%	81.822%	11.386%
	150	95.350%	95.305%	82.412%	12.938%
	200	96.338%	96.307%	82.720%	13.618%
	250	96.974%	96.947%	82.461%	14.514%
	300	97.425%	97.400%	83.042%	14.383%
0.0002 (2×10^{-4})	50	90.476%	90.436%	78.873%	11.602%
	100	94.498%	94.476%	80.277%	14.221%
	150	95.978%	95.953%	79.406%	16.572%
	200	96.315%	96.301%	74.684%	21.631%

BAB 4 IMPLEMENTASI DAN PENGUJIAN

Tabel 4.4 Hasil Pengujian (nilai rata-rata) pada Arsitektur 1

<i>Learning Rate</i>	<i>Epoch</i>	Hasil Nilai Rata-rata			
		<i>Accuracy</i>	<i>F1 Score</i>	<i>Val Accuracy</i>	$\Delta Accuracy$
0.00001 (10^{-5})	250	97.101%	97.088%	76.636%	20.465%
	300	97.133%	97.122%	71.193%	25.940%
	50	74.388%	73.682%	73.027%	1.361%
	100	80.323%	79.959%	78.418%	1.905%
	150	83.275%	83.097%	80.392%	2.883%
	200	85.213%	85.068%	81.498%	3.715%
	250	86.924%	86.815%	82.195%	4.729%
	300	88.233%	88.140%	82.689%	5.543%

Tabel 4.5 Hasil Pengujian (nilai maksimal) pada Arsitektur 1

<i>Learning Rate</i>	<i>Epoch</i>	Hasil Nilai Maksimal			
		<i>Accuracy</i>	<i>F1 Score</i>	<i>Val Accuracy</i>	$\Delta Accuracy$
0.001 (10^{-3})	50	89.340%	89.280%	73.800%	15.540%
	100	89.760%	89.790%	69.990%	19.770%
	150	93.490%	93.430%	72.480%	21.010%
	200	96.690%	96.680%	74.860%	21.830%
	250	97.430%	97.390%	81.980%	15.450%
	300	98.770%	98.750%	83.570%	15.200%
0.0001 (10^{-4})	50	96.100%	96.050%	85.010%	11.090%
	100	99.680%	99.670%	85.160%	14.520%
	150	99.910%	99.900%	85.100%	14.810%
	200	99.970%	99.970%	84.890%	15.080%
	250	99.970%	99.970%	85.120%	14.850%

Tabel 4.5 Hasil Pengujian (nilai maksimal) pada Arsitektur 1

<i>Learning Rate</i>	<i>Epoch</i>	Hasil Nilai Maksimal			
		<i>Accuracy</i>	<i>FI Score</i>	<i>Val Accuracy</i>	Δ <i>Accuracy</i>
	300	100.000%	100.000%	85.510%	14.490%
0.0002 (2×10^{-4})	50	98.180%	98.160%	84.760%	13.420%
	100	99.900%	99.890%	84.840%	15.060%
	150	99.820%	99.790%	84.690%	15.130%
	200	99.970%	99.970%	84.290%	15.680%
	250	99.990%	99.990%	85.170%	14.820%
	300	99.990%	99.990%	84.870%	15.120%
0.00001 (10^{-5})	50	84.820%	84.840%	83.500%	1.320%
	100	87.920%	87.930%	84.490%	3.430%
	150	90.310%	90.260%	84.910%	5.400%
	200	92.340%	92.370%	85.120%	7.220%
	250	94.320%	94.320%	85.000%	9.320%
	300	95.970%	95.950%	85.100%	10.870%

Untuk mempermudah pembacaan Tabel 4.4 dan 4.5, khususnya pembacaan nilai akurasi, dibuat sebuah grafik yang ditunjukkan pada Gambar 4.24 hingga 4.27. Grafik tersebut menggambarkan nilai rata-rata akurasi, akurasi validasi, dan selisih antara keduanya. Berikut adalah gambar-gambar tersebut. Kesimpulan mengenai tabel dan gambar dapat dilihat pada bagian setelah Gambar 4.27.

BAB 4 IMPLEMENTASI DAN PENGUJIAN



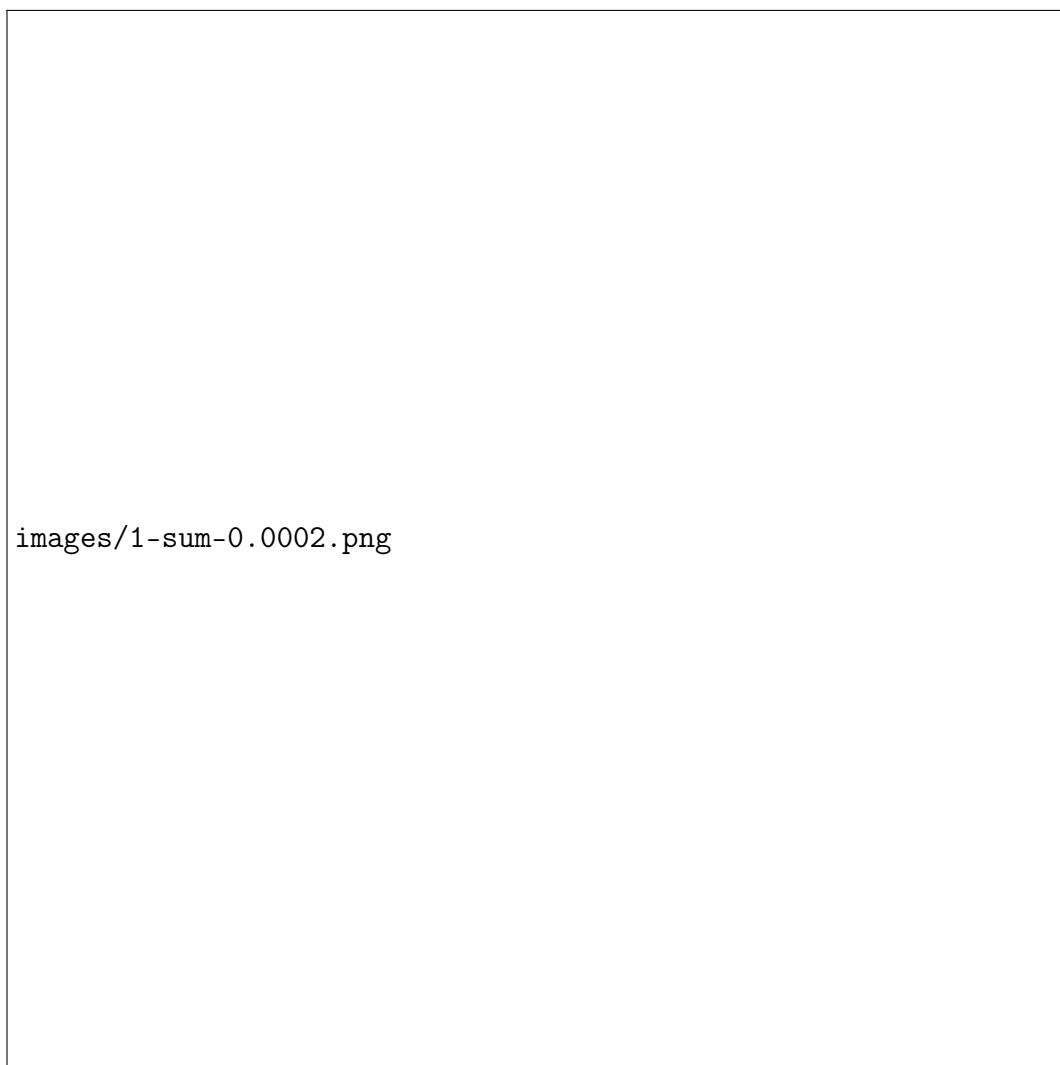
Gambar 4.24 Grafik akurasi, akurasi validasi, dan selisihnya pada pengujian di arsitektur pertama dengan $learning\ rate = 0.001 (10^{-3})$

BAB 4 IMPLEMENTASI DAN PENGUJIAN



Gambar 4.25 Grafik akurasi, akurasi validasi, dan selisihnya pada pengujian di arsitektur pertama dengan $learning\ rate = 0.0001 (10^{-4})$

BAB 4 IMPLEMENTASI DAN PENGUJIAN



Gambar 4.26 Grafik akurasi, akurasi validasi, dan selisihnya pada pengujian di arsitektur pertama dengan $learning\ rate = 0.0002 (2 \times 10^{-4})$



images/1-sum-0.00001.png

Gambar 4.27 Grafik akurasi, akurasi validasi, dan selisihnya pada pengujian di arsitektur pertama dengan $learning\ rate = 0.00001 (10^{-5})$

Berdasarkan hasil pengujian dengan arsitektur pertama yang ditunjukkan pada Tabel 4.4 dan 4.5 serta yang digambarkan pada Gambar 4.24 hingga Gambar 4.27, dapat diambil kesimpulan tentang beberapa hal berikut.

1. Nilai $learning\ rate$ 0.001 (10^{-3}) tidak cocok digunakan pada arsitektur ini karena selisih rata-rata antara akurasi dan akurasi validasi terlalu jauh, mengakibatkan *overfitting* pada model.
2. Nilai $learning\ rate$ 0.0001 (10^{-4}) dan $0.0002 (2 \times 10^{-4})$ dapat menghasilkan akurasi yang tinggi, namun sedikit mengalami *overfitting* karena selisih rata-rata antara akurasi dan akurasi validasi tidak sekecil $learning\ rate$ 0.00001 (10^{-5}). Hal ini berarti meskipun model dapat menghasilkan akurasi tinggi, model hanya akan dapat berperforma baik pada data yang sudah dilihat dan dipelajari olehnya. Performa ini akan berkurang ketika model menjumpai data yang baru.

3. Nilai *learning rate* $0.00001 (10^{-5})$ memiliki performa yang baik pada arsitektur ini. Jika dilihat dari grafik akurasi dan akurasi validasinya, tidak pernah ada laju pergerakan yang terlalu ekstrem baik ke arah atas maupun bawah. Namun, *learning rate* ini tidak dapat menghasilkan akurasi yang tinggi seperti yang dihasilkan pada nilai *learning rate* $0.0001 (10^{-4})$ dan $0.0002 (2 \times 10^{-4})$.
4. Semakin banyak nilai *epoch*, semakin mungkin model dapat menghasilkan akurasi yang tinggi. Namun, jika tidak dikombinasikan dengan nilai *learning rate* yang tepat, model akan mengalami *overfitting*.

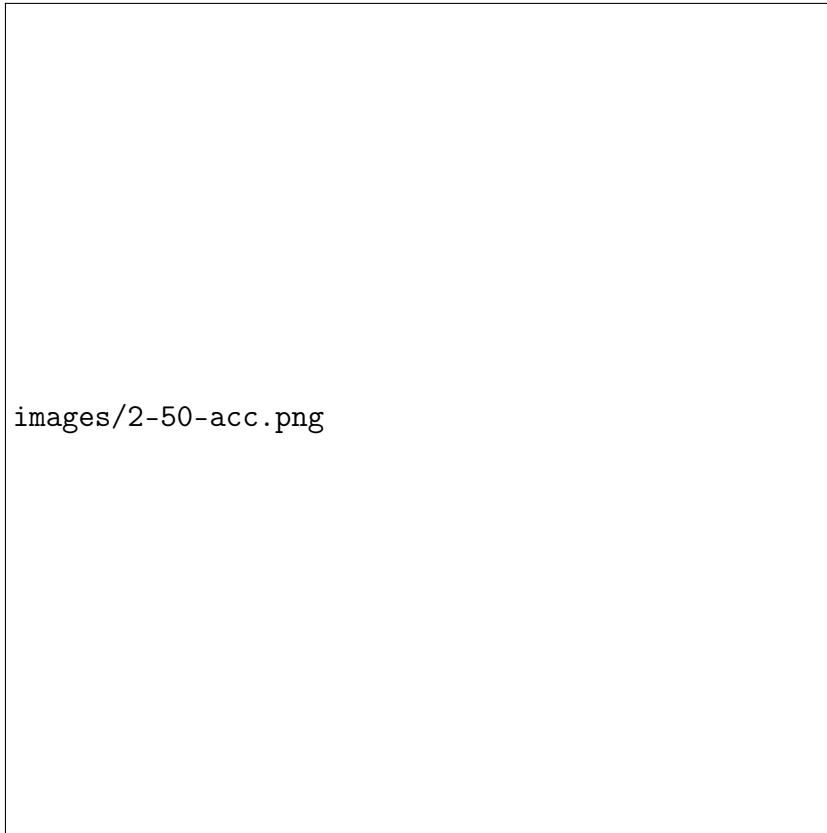
4.5.2 Pengujian pada Arsitektur 2

Bagian ini akan merangkum hasil pengujian pada arsitektur kedua. Penjelasan dibagi per *epoch* dan *learning rate*. Tabel yang merangkum hasil pengujian ada pada Tabel 4.6 dan 4.7 di bagian 4.5.2.7.

4.5.2.1 Nilai *Epoch* = 50

Grafik pengujian dengan nilai *epoch* 50 untuk masing-masing nilai *learning rate* dapat dilihat pada Gambar 4.28 untuk akurasi pengujian dan Gambar 4.29 untuk akurasi validasi pengujian. Dapat dilihat pada kedua gambar berikut, nilai *learning rate* memiliki pengaruh kenaikan akurasi sekaligus memengaruhi kestabilan model. Model dengan *learning rate* $0.001 (10^{-3})$ cenderung tidak stabil, dapat dilihat pada grafik akurasi validasinya. Nilai *learning rate* yang tepat akan menghasilkan grafik yang stabil. Namun, akurasi yang dihasilkan tidak terlalu tinggi. Dapat dilihat dengan *learning rate* $0.00001 (10^{-5})$, akurasi tidak dapat mencapai 80%. Hal ini berarti jumlah *filter* memiliki pengaruh terhadap akurasi. Semakin banyak *filter* yang digunakan, model dapat belajar lebih banyak fitur.

BAB 4 IMPLEMENTASI DAN PENGUJIAN



images/2-50-acc.png

Gambar 4.28 Grafik perubahan akurasi pada pengujian arsitektur 2 untuk $epoch = 50$



images/2-50-valacc.png

Gambar 4.29 Grafik perubahan akurasi validasi pada pengujian arsitektur 2 untuk $epoch = 50$

4.5.2.2 Nilai *Epoch* = 100

Hasil pengujian dalam arsitektur kedua dengan nilai *epoch* 100 dapat dilihat pada Gambar 4.30 dan 4.31. Hasil pengujian menunjukkan bahwa nilai *learning rate* sangat berpengaruh. Hal ini ditunjukkan dari pergerakan grafik pada masing-masing nilai parameter yang diuji. Nilai *learning rate* yang baik untuk model tidak akan menghasilkan bentuk grafik yang tidak stabil. Hal ini dapat dilihat dengan jelas dari pergerakan akurasi validasi. Pada arsitektur ini, dengan jumlah *epoch* 100, nilai *learning rate* terbaik yang dapat diambil adalah $0.0001 (10^{-4})$ karena menghasilkan akurasi tertinggi dan akurasi validasi yang cenderung stabil. Nilai *learning rate* $0.00001 (10^{-5})$ belum dapat menghasilkan akurasi yang tinggi, meskipun jumlah *epoch* sudah ditambahkan.



images/2-100-acc.png

Gambar 4.30 Grafik perubahan akurasi pada pengujian arsitektur 2 untuk *epoch* = 100



images/2-100-valacc.png

Gambar 4.31 Grafik perubahan akurasi validasi pada pengujian arsitektur 2 untuk $epoch = 100$

4.5.2.3 Nilai $Epoch = 150$

Hasil pengujian dengan nilai $epoch$ 150 pada arsitektur kedua dapat dilihat pada Gambar 4.32 dan Gambar 4.33. Nilai akurasi cenderung memiliki kesamaan antara tiga buah *learning rate* yang diujikan, namun nilai *learning rate* $0.00001 (10^{-5})$ masih tetap belum dapat menghasilkan akurasi setinggi yang lainnya. Nilai *learning rate* $0.0001 (10^{-4})$ dan $0.00001 (10^{-5})$ dapat dinyatakan memiliki performa yang cukup baik pada kombinasi $epoch$ 150 di arsitektur kedua, dengan mempertimbangkan kestabilan model dan nilai akurasi tertinggi yang dihasilkan.

BAB 4 IMPLEMENTASI DAN PENGUJIAN



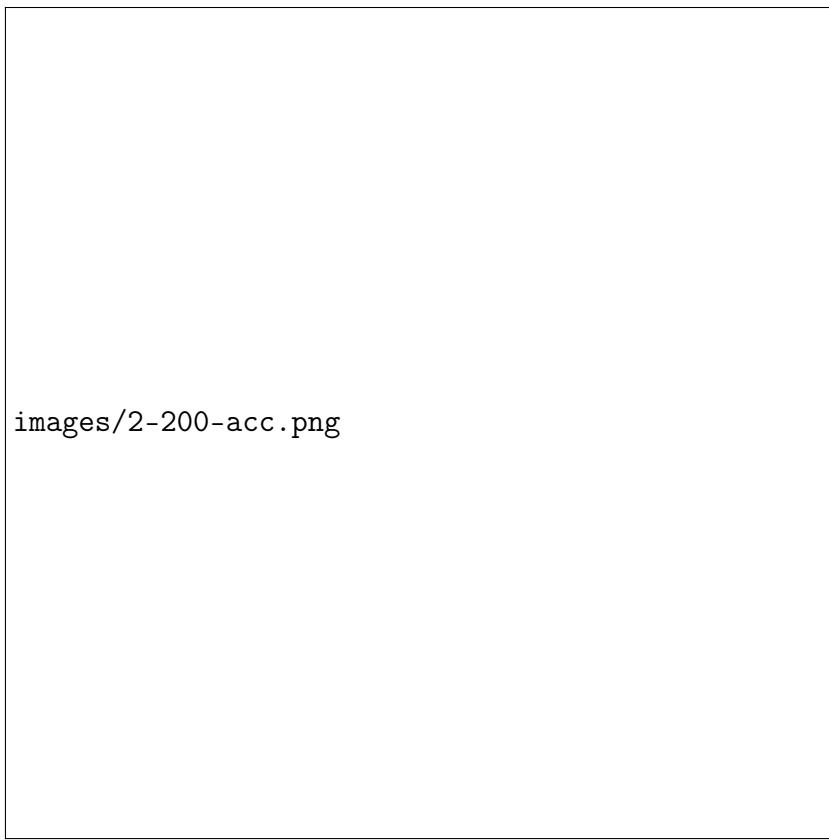
Gambar 4.32 Grafik perubahan akurasi pada pengujian arsitektur 2 untuk $epoch = 150$



Gambar 4.33 Grafik perubahan akurasi validasi pada pengujian arsitektur 2 untuk $epoch = 150$

4.5.2.4 Nilai *Epoch* = 200

Gambar 4.34 dan Gambar 4.35 menunjukkan hasil pengujian pada arsitektur kedua dengan nilai *epoch* 200. Selain adanya pengaruh dari *learning rate*, pengaruh nilai *epoch* mulai terlihat juga pada pengujian ini. Model masih menunjukkan peningkatan akurasi, di mana grafik yang dihasilkan belum landai sepenuhnya, tidak seperti pada arsitektur pertama. Pada arsitektur pertama, dengan nilai *epoch* 200, grafik akurasi sudah mulai mengalami ketidakstabilan. Arsitektur ini menggunakan lebih sedikit jumlah *filter*, dan pada kombinasi *epoch* 200, model masih belum belajar terlalu banyak sehingga masih ada peningkatan akurasi. Hal ini menyiratkan bahwa jumlah *filter* juga memengaruhi akurasi. Nilai *epoch* ini dapat dinyatakan tidak cocok untuk *learning rate* $0.0002 (2 \times 10^{-4})$ dan $0.001 (10^{-3})$ karena menghasilkan akurasi validasi yang tidak stabil.



images/2-200-acc.png

Gambar 4.34 Grafik perubahan akurasi pada pengujian arsitektur 2 untuk *epoch* = 200

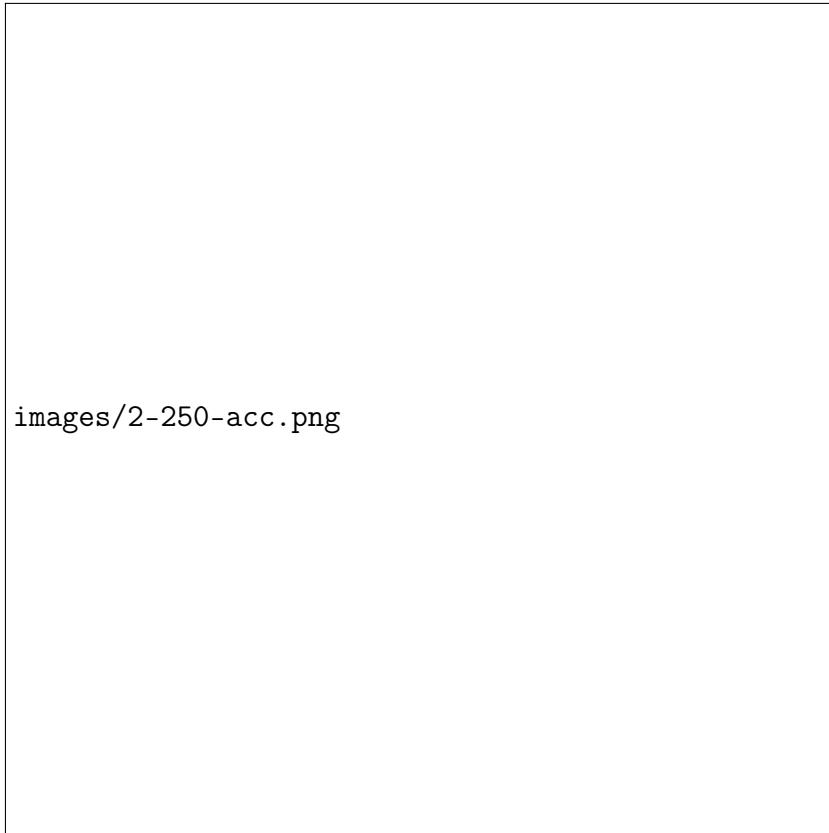


Gambar 4.35 Grafik perubahan akurasi validasi pada pengujian arsitektur 2 untuk $epoch = 200$

4.5.2.5 Nilai $Epoch = 250$

Di bawah ini, ditunjukkan hasil pengujian dengan nilai $epoch$ 250 pada arsitektur kedua dengan Gambar 4.36 dan Gambar 4.37. Sejauh ini, dapat diambil kesimpulan bahwa nilai *learning rate* 0.001 (10^{-3}) dan 0.0002 (2×10^{-4}) masih tidak cocok digunakan pada model, karena meskipun menghasilkan akurasi yang tinggi, model mengalami *overfit* dan tidak stabil ketika dimasukkan data yang baru. Hal ini ditunjukkan pada laju grafik akurasi validasi. Grafik akurasi validasi juga menegaskan performa model yang tidak baik untuk nilai $epoch$ dan *learning rate* tersebut karena semakin tidak beraturan. Dengan nilai $epoch$ 250 pada arsitektur ini, model belum menghasilkan grafik akurasi yang landai, sehingga masih dapat diuji penambahan $epoch$ untuk melihat kapan laju grafik akurasi akan mulai melandai.

BAB 4 IMPLEMENTASI DAN PENGUJIAN



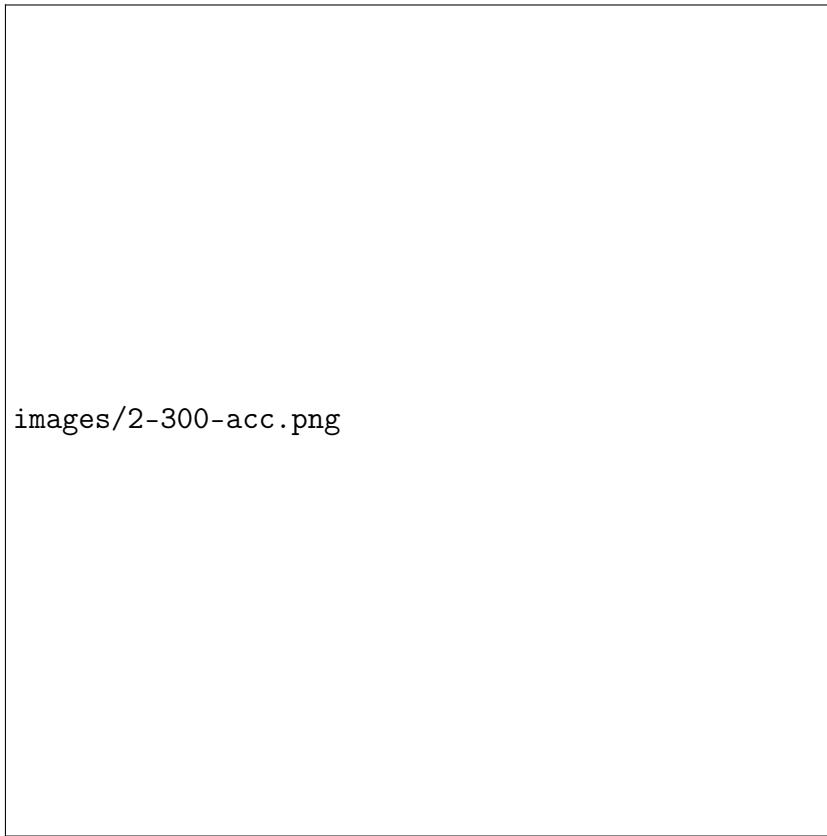
Gambar 4.36 Grafik perubahan akurasi pada pengujian arsitektur 2 untuk $epoch = 250$



Gambar 4.37 Grafik perubahan akurasi validasi pada pengujian arsitektur 2 untuk $epoch = 250$

4.5.2.6 Nilai *Epoch* = 300

Hasil pengujian dengan nilai *epoch* 300 pada arsitektur kedua dapat dilihat pada Gambar 4.38 dan Gambar 4.39. Sebelumnya, telah dinyatakan bahwa nilai *learning rate*, nilai *epoch*, dan jumlah *filter* memiliki pengaruh terhadap laju pergerakan grafik akurasi dan akurasi validasi. Ketika nilai *epoch* optimal, grafik akan mulai berbentuk landai, tidak lagi menuik ke atas atau bawah secara ekstrem. Pada pengujian dengan arsitektur ini, jumlah *epoch* dapat terus ditambahkan hingga 300 untuk memaksimalkan akurasi yang dapat dihasilkan model. Jumlah *filter* yang tidak sebanyak pada arsitektur pertama memungkinkan hal ini terjadi, karena pada arsitektur pertama, nilai *epoch* 200 sudah mulai menunjukkan ketidakstabilan grafik akurasi. Dapat disimpulkan bahwa dengan menggunakan arsitektur kedua, *learning rate* $0.00001 (10^{-5})$ menghasilkan model yang stabil, namun tidak dapat mencapai akurasi yang tinggi. Nilai *learning rate* $0.001 (10^{-3})$ dan $0.0002 (2 \times 10^{-4})$ tetap tidak cocok digunakan jika mempertimbangkan laju perubahan pada grafik akurasi validasi.



Gambar 4.38 Grafik perubahan akurasi pada pengujian arsitektur 2 untuk *epoch* = 300



images/2-300-valacc.png

Gambar 4.39 Grafik perubahan akurasi validasi pada pengujian arsitektur 2 untuk $epoch = 300$

4.5.2.7 Tabel Hasil Pengujian pada Arsitektur 2

Tabel 4.6 dan 4.7 merangkum seluruh hasil pengujian estimasi *pitch* yang sudah dilakukan pada arsitektur kedua. Kedua tabel masing-masing berisi nilai *mean* (rata-rata) dan nilai maksimum dari setiap parameter percobaan. Kesimpulan yang diambil dari Tabel 4.6 dan 4.7 dapat dilihat setelah Tabel 4.7.

Tabel 4.6 Hasil Pengujian (nilai rata-rata) pada Arsitektur 2

<i>Learning Rate</i>	<i>Epoch</i>	Hasil Nilai Rata-rata			
		<i>Accuracy</i>	<i>F1 Score</i>	<i>Val Accuracy</i>	$\Delta Accuracy$
0.001 (10^{-3})	50	85.649%	85.530%	46.317%	39.332%
	100	87.710%	87.655%	44.447%	43.263%
	150	89.187%	89.164%	50.576%	38.611%
	200	90.674%	90.661%	51.947%	38.727%
	250	91.718%	91.699%	55.974%	35.744%
	300	92.511%	92.487%	59.435%	33.076%

BAB 4 IMPLEMENTASI DAN PENGUJIAN

Tabel 4.6 Hasil Pengujian (nilai rata-rata) pada Arsitektur 2

<i>Learning Rate</i>	<i>Epoch</i>	Hasil Nilai Rata-rata			
		<i>Accuracy</i>	<i>F1 Score</i>	<i>Val Accuracy</i>	$\Delta Accuracy$
0.0001 (10^{-4})	50	81.314%	80.882%	76.666%	4.648%
	100	85.657%	85.449%	80.237%	5.419%
	150	88.277%	88.142%	81.683%	6.595%
	200	90.236%	90.139%	82.149%	8.087%
	250	91.801%	91.737%	82.184%	9.617%
	300	93.019%	92.943%	82.666%	10.353%
0.0002 (2×10^{-4})	50	84.068%	83.764%	77.591%	6.477%
	100	87.863%	87.739%	80.060%	7.803%
	150	90.338%	90.267%	79.447%	10.891%
	200	91.869%	91.801%	79.937%	11.932%
	250	93.204%	93.156%	79.019%	14.186%
	300	94.079%	94.033%	78.470%	15.609%
0.00001 (10^{-5})	50	60.389%	56.964%	60.451%	0.062%
	100	69.913%	67.715%	70.820%	0.908%
	150	74.243%	73.004%	74.423%	0.180%
	200	77.048%	76.007%	76.634%	0.414%
	250	78.806%	77.942%	78.048%	0.758%
	300	80.301%	79.664%	79.485%	0.816%

Tabel 4.7 Hasil Pengujian (nilai maksimal) pada Arsitektur 2

<i>Learning Rate</i>	<i>Epoch</i>	Hasil Nilai Maksimal			
		<i>Accuracy</i>	<i>F1 Score</i>	<i>Val Accuracy</i>	$\Delta Accuracy$
0.001 (10^{-3})	50	89.000%	89.030%	71.530%	17.470%
	100	91.260%	91.310%	74.410%	16.850%

BAB 4 IMPLEMENTASI DAN PENGUJIAN

Tabel 4.7 Hasil Pengujian (nilai maksimal) pada Arsitektur 2

<i>Learning Rate</i>	<i>Epoch</i>	Hasil Nilai Maksimal			
		<i>Accuracy</i>	<i>F1 Score</i>	<i>Val Accuracy</i>	Δ <i>Accuracy</i>
0.0001 (10^{-4})	150	93.620%	93.650%	80.580%	13.040%
	200	95.720%	95.660%	82.180%	13.540%
	250	97.240%	97.250%	81.500%	15.740%
	300	97.790%	97.770%	82.720%	15.070%
0.0002 (2×10^{-4})	50	88.530%	88.530%	84.770%	3.760%
	100	92.030%	92.100%	85.060%	6.970%
	150	95.110%	95.110%	85.220%	9.890%
	200	97.700%	97.690%	85.340%	12.360%
	250	98.790%	98.770%	85.190%	13.600%
	300	99.400%	99.380%	85.310%	14.090%
0.00001 (10^{-5})	50	89.970%	90.030%	84.730%	5.240%
	100	93.750%	93.790%	84.720%	9.030%
	150	96.800%	96.790%	84.840%	11.960%
	200	97.870%	97.880%	84.680%	13.190%
	250	98.740%	98.720%	84.440%	14.300%
	300	99.200%	99.170%	84.830%	14.370%

Untuk mempermudah pembacaan Tabel 4.6 dan 4.7, khususnya pembacaan nilai

BAB 4 IMPLEMENTASI DAN PENGUJIAN

akurasi, dibuat sebuah grafik yang ditunjukkan pada Gambar 4.40 hingga 4.43. Grafik tersebut menggambarkan nilai rata-rata akurasi, akurasi validasi, dan selisih antara keduanya. Berikut adalah gambar-gambar tersebut. Kesimpulan mengenai tabel dan gambar dapat dilihat pada bagian setelah Gambar 4.43.



images/2-sum-0.001.png

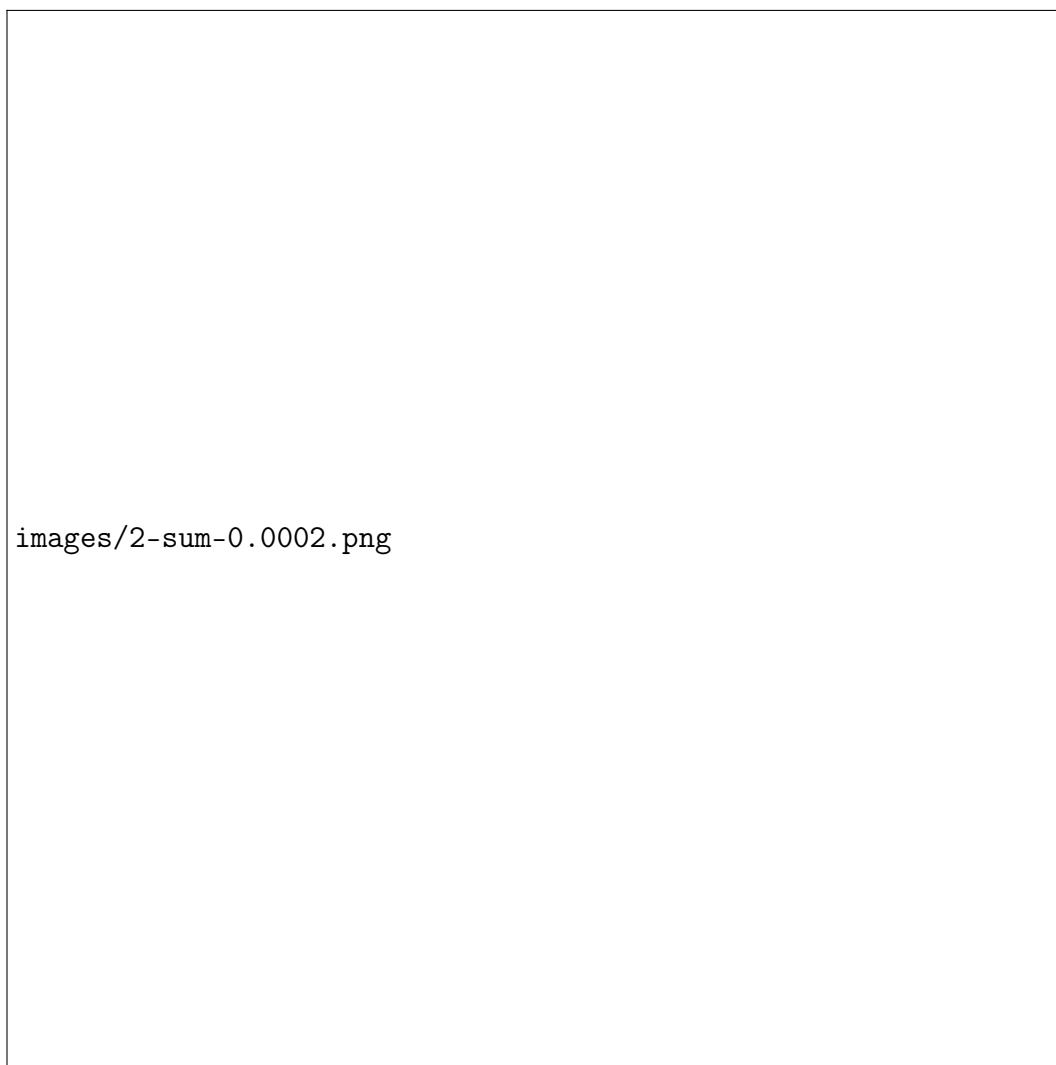
Gambar 4.40 Grafik akurasi, akurasi validasi, dan selisihnya pada pengujian di arsitektur kedua dengan $learning\ rate = 0.001 (10^{-3})$

BAB 4 IMPLEMENTASI DAN PENGUJIAN



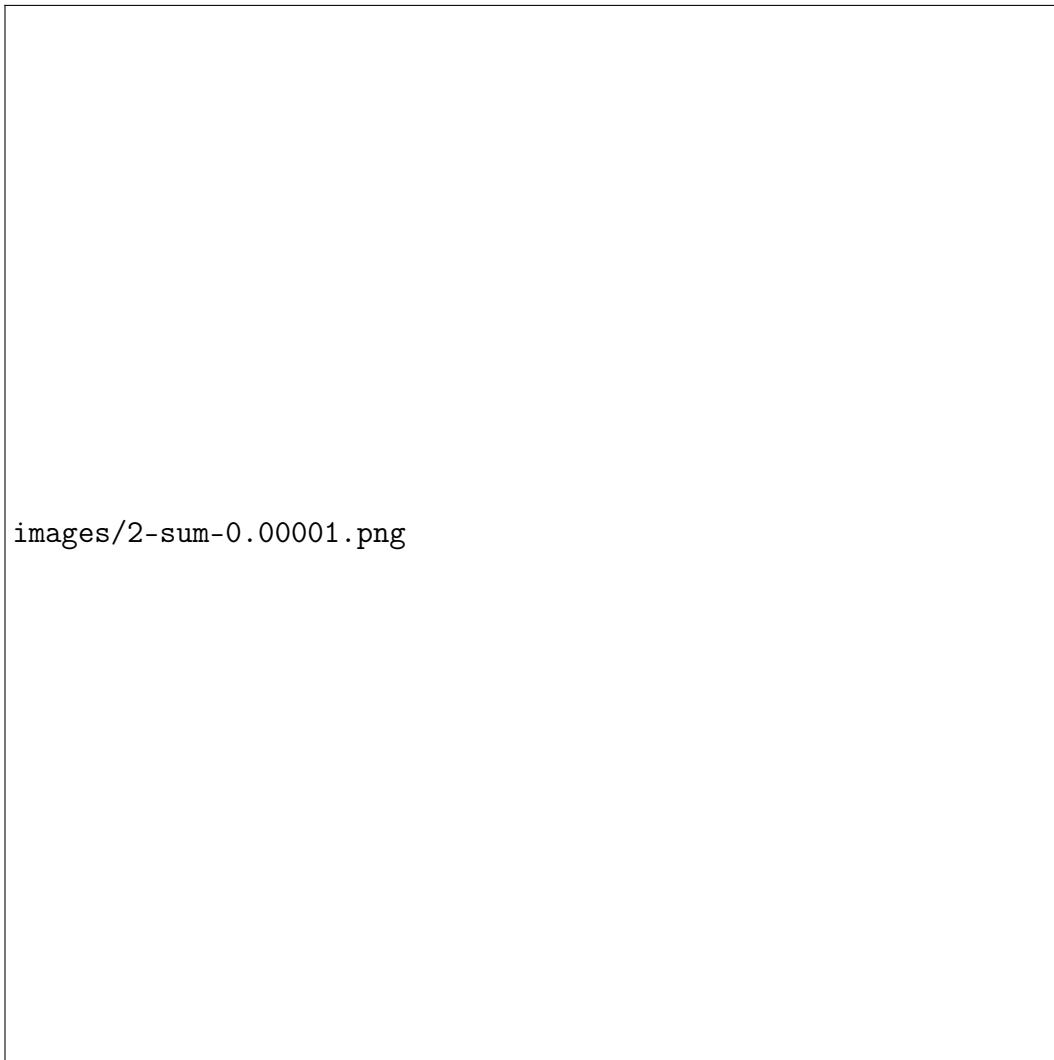
Gambar 4.41 Grafik akurasi, akurasi validasi, dan selisihnya pada pengujian di arsitektur kedua dengan $learning\ rate = 0.0001 (10^{-4})$

BAB 4 IMPLEMENTASI DAN PENGUJIAN



images/2-sum-0.0002.png

Gambar 4.42 Grafik akurasi, akurasi validasi, dan selisihnya pada pengujian di arsitektur kedua dengan $learning\ rate = 0.0002 (2 \times 10^{-4})$



images/2-sum-0.00001.png

Gambar 4.43 Grafik akurasi, akurasi validasi, dan selisihnya pada pengujian di arsitektur kedua dengan $learning\ rate = 0.00001 (10^{-5})$

Berdasarkan hasil pengujian dengan arsitektur kedua yang ditunjukkan pada Tabel 4.6 dan 4.7 serta yang digambarkan pada Gambar 4.40 hingga Gambar 4.43, dapat diambil kesimpulan tentang beberapa hal berikut.

1. Dengan jumlah konvolusi yang sama namun *filter* yang berbeda, nilai *epoch* dapat terus ditambahkan hingga laju grafik mulai landai. Jumlah *filter* akan menentukan banyaknya fitur yang dapat dipelajari oleh CNN, sehingga dengan mengurangi jumlah *filter*, dapat digunakan *epoch* yang semakin banyak. Sebaliknya, bila jumlah *filter* sudah banyak dengan jumlah konvolusi yang sama, tidak perlu menggunakan terlalu banyak *epoch* untuk menghindari model yang terlalu banyak belajar yang akan mengakibatkan *overfitting*.
2. Nilai *learning rate* $0.001 (10^{-3})$ dan $0.0002 (2 \times 10^{-4})$ kurang cocok digunakan pada arsitektur ini karena selisih rata-rata antara akurasi dan akurasi validasi

yang jauh.

3. Nilai *learning rate* 0.0001 (10^{-4}) dapat menghasilkan akurasi yang tinggi dan dapat dinyatakan memiliki performa yang baik pada arsitektur ini. Namun, dengan nilai *learning rate* tersebut, model sedikit mengalami *overfitting* karena selisih rata-rata antara akurasi dan akurasi validasi tidak sekecil *learning rate* 0.00001 (10^{-5}).
4. Nilai *learning rate* 0.00001 (10^{-5}) memiliki performa yang kurang baik pada arsitektur ini. Meskipun jika dilihat dari grafik akurasi dan akurasi validasinya tidak pernah ada laju pergerakan yang terlalu ekstrem baik ke arah atas maupun bawah, namun akurasi yang dihasilkan tidak terlalu tinggi dan selalu berada di bawah akurasi yang dihasilkan oleh kombinasi *learning rate* yang lainnya.

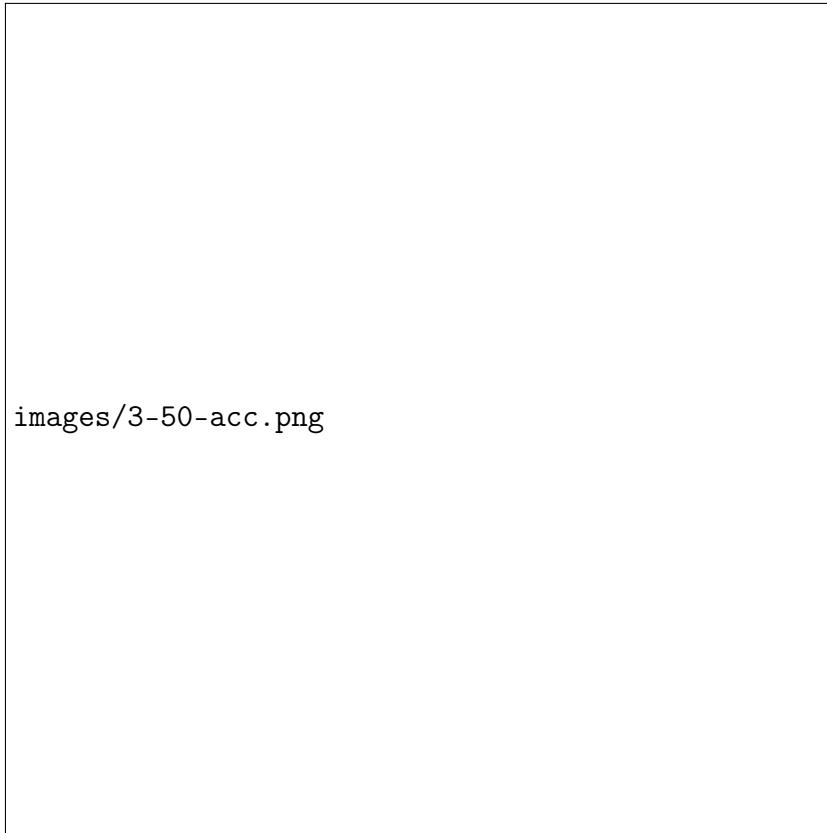
4.5.3 Pengujian pada Arsitektur 3

Bagian ini akan merangkum hasil pengujian pada arsitektur ketiga. Penjelasan dibagi per *epoch* dan *learning rate*. Tabel yang merangkum hasil pengujian ada pada Tabel 4.8 dan 4.9 di bagian 4.5.3.7.

4.5.3.1 Nilai *Epoch* = 50

Grafik pengujian dengan nilai *epoch* 50 untuk masing-masing nilai *learning rate* pada arsitektur ketiga dapat dilihat pada Gambar 4.44 untuk akurasi pengujian dan Gambar 4.45 untuk akurasi validasi pengujian. Dapat dilihat pada kedua gambar tersebut, nilai *learning rate* memiliki pengaruh kenaikan akurasi sekaligus memengaruhi kestabilan model. Nilai *learning rate* yang tepat akan menghasilkan grafik yang stabil. Nilai *learning rate* 0.00001 (10^{-5}) belum dapat menghasilkan akurasi yang tinggi seperti *learning rate* lainnya dengan *epoch* 50, sehingga dengan *learning rate* tersebut, jumlah *epoch* masih dapat ditambahkan. Nilai *learning rate* 0.001 (10^{-3}) terlalu besar, dibuktikan dengan perubahan akurasi validasi yang sudah *random* pada *epoch* yang belum terlalu banyak.

BAB 4 IMPLEMENTASI DAN PENGUJIAN



Gambar 4.44 Grafik perubahan akurasi pada pengujian arsitektur 3 untuk $epoch = 50$



Gambar 4.45 Grafik perubahan akurasi validasi pada pengujian arsitektur 3 untuk $epoch = 50$

4.5.3.2 Nilai *Epoch* = 100

Gambar 4.46 dan Gambar 4.47 merupakan hasil pengujian pada arsitektur ketiga dengan nilai *epoch* 100. Hasil pengujian menunjukkan bahwa nilai *learning rate* sangat berpengaruh. Hal ini ditunjukkan dari pergerakan grafik pada masing-masing nilai parameter yang diuji. Nilai *learning rate* yang baik untuk model tidak akan menghasilkan bentuk grafik yang tidak stabil yang dapat dilihat dari pergerakan grafik akurasi validasi. Dengan tiga kali konvolusi, didapatkan hasil pada *epoch* 100, nilai *learning rate* $0.0002 (2 \times 10^{-4})$ dan $0.0001 (10^{-4})$ tidak dapat memberikan model yang stabil.



Gambar 4.46 Grafik perubahan akurasi pada pengujian arsitektur 3 untuk *epoch* = 100



Gambar 4.47 Grafik perubahan akurasi validasi pada pengujian arsitektur 3 untuk $epoch = 100$

4.5.3.3 Nilai $Epoch = 150$

Hasil pengujian dengan nilai $epoch$ 150 pada arsitektur ketiga dapat dilihat pada Gambar 4.48 dan Gambar 4.49. Bukti bahwa nilai *learning rate* sangat berpengaruh pada laju pergerakan akurasi semakin terlihat pada pengujian ini. Pada nilai *learning rate* $0.0002 (2 \times 10^{-4})$ dan $0.001 (10^{-3})$, akurasi validasi semakin jelas terlihat ketidakaturannya, yang menandakan bahwa nilai *learning rate* ini terlalu besar untuk $epoch$ 150 pada arsitektur ketiga.

BAB 4 IMPLEMENTASI DAN PENGUJIAN



Gambar 4.48 Grafik perubahan akurasi pada pengujian arsitektur 3 untuk $epoch = 150$



Gambar 4.49 Grafik perubahan akurasi validasi pada pengujian arsitektur 3 untuk $epoch = 150$

4.5.3.4 Nilai *Epoch* = 200

Gambar 4.50 dan Gambar 4.51 menunjukkan hasil pengujian pada arsitektur ketiga dengan nilai *epoch* 200. Selain adanya pengaruh dari *learning rate*, pengaruh nilai *epoch* mulai terlihat juga pada pengujian ini. Jika nilai *epoch* terlalu banyak, laju pergerakan grafik akurasi dengan data latih pun akan mengalami ketidakstabilan. Hal ini dapat terjadi karena model terlalu banyak belajar menangani data, sehingga model kurang tepat melakukan estimasi karena banyaknya kemungkinan yang dipelajari oleh model. Nilai *epoch* ini dapat dinyatakan tidak cocok untuk *learning rate* $0.0002 (2 \times 10^{-4})$, di mana dapat dilihat menjelang akhir pengujian, model sudah mengalami penurunan akurasi karena kesalahan estimasi yang dilakukan. Selain itu, akurasi validasi juga semakin tidak beraturan, yang menandakan *learning rate* $0.0002 (2 \times 10^{-4})$ sudah terlalu besar. Pada *epoch* 200 ini juga, dengan jumlah *filter* yang digunakan, dapat dinyatakan bahwa jumlah *epoch* sudah mendekati optimal. Hal ini dapat dilihat dari grafik akurasi yang sudah mulai melandai.



Gambar 4.50 Grafik perubahan akurasi pada pengujian arsitektur 3 untuk *epoch* = 200



Gambar 4.51 Grafik perubahan akurasi validasi pada pengujian arsitektur 3 untuk $epoch = 200$

4.5.3.5 Nilai $Epoch = 250$

Gambar 4.52 dan Gambar 4.53 menunjukkan hasil pengujian pada arsitektur ketiga dengan nilai $epoch$ 250. Selain adanya pengaruh dari *learning rate*, pengaruh nilai $epoch$ semakin terlihat juga pada pengujian ini. Dengan nilai *learning rate* 0.0002 (2×10^{-4}) dan 0.001 (10^{-3}) yang dikombinasikan dengan nilai $epoch$ ini, kombinasi nilai-nilai tersebut dapat dinyatakan tidak cocok.

BAB 4 IMPLEMENTASI DAN PENGUJIAN



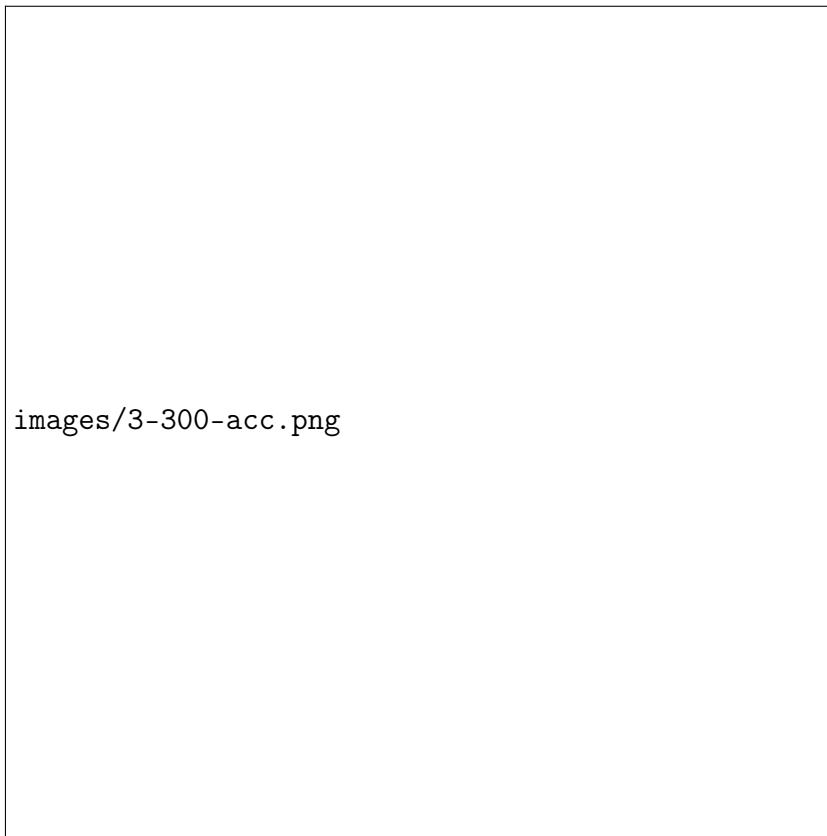
Gambar 4.52 Grafik perubahan akurasi pada pengujian arsitektur 3 untuk $epoch = 250$



Gambar 4.53 Grafik perubahan akurasi validasi pada pengujian arsitektur 3 untuk $epoch = 250$

4.5.3.6 Nilai *Epoch* = 300

Hasil pengujian dengan nilai *epoch* 300 pada arsitektur ketiga dapat dilihat pada Gambar 4.54 dan Gambar 4.55. Sebelumnya, telah dinyatakan bahwa nilai *learning rate* dan nilai *epoch* memiliki pengaruh terhadap laju pergerakan grafik akurasi dan akurasi validasi. Nilai *epoch* sebanyak 300 sudah dapat dinyatakan terlalu banyak, karena mulai timbul ketidakstabilan pada laju pergerakan akurasi. Dengan nilai *epoch* 300, dapat dinyatakan bahwa model optimal dengan *learning rate* 0.00001 (10^{-5}). Meskipun tidak mendapatkan akurasi maksimal, model menunjukkan adanya kestabilan di mana laju perubahan akurasi dan akurasi validasi tidak terlalu fluktuatif.



images/3-300-acc.png

Gambar 4.54 Grafik perubahan akurasi pada pengujian arsitektur 3 untuk *epoch* = 300



Gambar 4.55 Grafik perubahan akurasi validasi pada pengujian arsitektur 3 untuk $epoch = 300$

4.5.3.7 Tabel Hasil Pengujian pada Arsitektur 3

Tabel 4.8 dan 4.9 merangkum seluruh hasil pengujian estimasi *pitch* yang sudah dilakukan pada arsitektur ketiga. Kedua tabel masing-masing berisi nilai *mean* (rata-rata) dan nilai maksimum dari setiap parameter percobaan. Kesimpulan yang diambil dari Tabel 4.8 dan 4.9 dapat dilihat setelah Tabel 4.9.

Tabel 4.8 Hasil Pengujian (nilai rata-rata) pada Arsitektur 3

<i>Learning Rate</i>	<i>Epoch</i>	Hasil Nilai Rata-rata			
		<i>Accuracy</i>	<i>F1 Score</i>	<i>Val Accuracy</i>	$\Delta Accuracy$
0.001 (10^{-3})	50	85.553%	85.504%	41.951%	43.602%
	100	86.969%	86.933%	40.973%	45.996%
	150	87.847%	87.831%	41.014%	46.833%
	200	88.518%	88.493%	41.851%	46.667%
	250	89.405%	89.399%	44.592%	44.813%
	300	89.769%	89.758%	45.173%	44.596%

BAB 4 IMPLEMENTASI DAN PENGUJIAN

Tabel 4.8 Hasil Pengujian (nilai rata-rata) pada Arsitektur 3

<i>Learning Rate</i>	<i>Epoch</i>	Hasil Nilai Rata-rata			
		<i>Accuracy</i>	<i>F1 Score</i>	<i>Val Accuracy</i>	$\Delta Accuracy$
0.0001 (10^{-4})	50	85.483%	85.146%	81.582%	3.901%
	100	89.590%	89.428%	82.881%	6.710%
	150	92.556%	92.408%	83.511%	9.045%
	200	94.032%	93.926%	83.489%	10.544%
	250	95.266%	95.174%	82.998%	12.268%
	300	95.933%	95.853%	82.584%	13.349%
0.0002 (2×10^{-4})	50	87.565%	87.397%	79.082%	8.483%
	100	91.558%	91.446%	78.117%	13.441%
	150	93.686%	93.591%	74.133%	19.553%
	200	94.897%	94.830%	73.539%	21.357%
	250	95.777%	95.713%	71.385%	24.392%
	300	96.298%	96.244%	70.019%	26.278%
0.00001 (10^{-5})	50	73.683%	71.688%	73.396%	0.287%
	100	78.605%	77.473%	77.866%	0.739%
	150	81.882%	81.124%	80.690%	1.192%
	200	83.200%	82.609%	81.510%	1.690%
	250	84.527%	84.078%	82.310%	2.217%
	300	85.670%	85.320%	82.899%	2.771%

Tabel 4.9 Hasil Pengujian (nilai maksimal) pada Arsitektur 3

<i>Learning Rate</i>	<i>Epoch</i>	Hasil Nilai Maksimal			
		<i>Accuracy</i>	<i>F1 Score</i>	<i>Val Accuracy</i>	$\Delta Accuracy$
0.001 (10^{-3})	50	88.220%	88.240%	68.990%	19.230%
	100	89.470%	89.430%	70.580%	18.890%

BAB 4 IMPLEMENTASI DAN PENGUJIAN

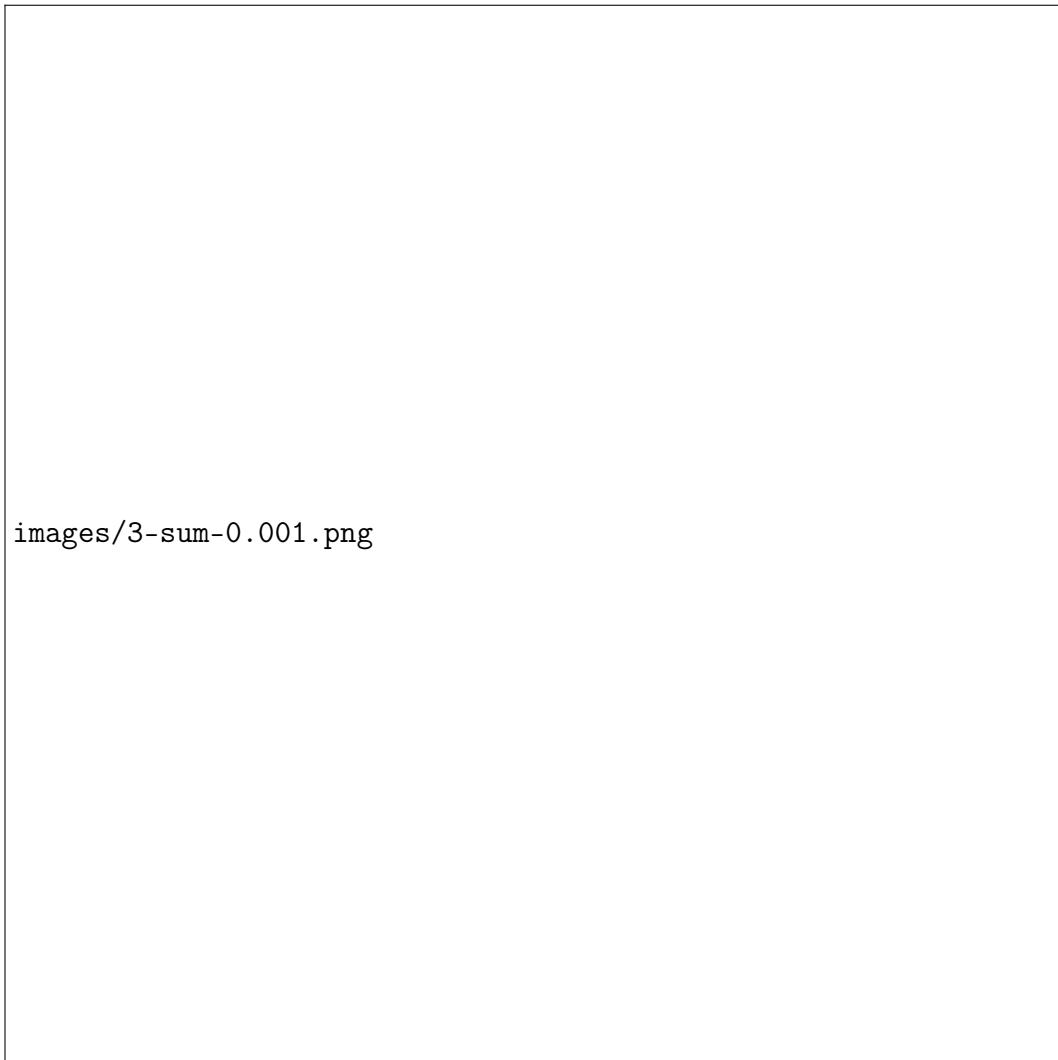
Tabel 4.9 Hasil Pengujian (nilai maksimal) pada Arsitektur 3

<i>Learning Rate</i>	<i>Epoch</i>	Hasil Nilai Maksimal			
		<i>Accuracy</i>	<i>F1 Score</i>	<i>Val Accuracy</i>	Δ <i>Accuracy</i>
0.0001 (10^{-4})	150	90.290%	90.330%	78.280%	12.010%
	200	91.690%	91.640%	71.530%	20.160%
	250	93.340%	93.310%	79.150%	14.190%
	300	94.040%	94.010%	79.230%	14.810%
0.0002 (2×10^{-4})	50	91.330%	91.350%	85.080%	6.250%
	100	96.340%	96.300%	85.360%	10.980%
	150	99.270%	99.230%	85.760%	13.510%
	200	99.770%	99.740%	85.800%	13.970%
	250	99.880%	99.890%	85.900%	13.980%
	300	99.930%	99.940%	85.770%	14.160%
0.00001 (10^{-5})	50	92.860%	92.860%	85.010%	7.850%
	100	97.920%	97.890%	84.960%	12.960%
	150	99.560%	99.550%	84.620%	14.940%
	200	99.750%	99.760%	85.170%	14.580%
	250	99.860%	99.850%	84.950%	14.910%
	300	99.910%	99.920%	84.780%	15.130%

Untuk mempermudah pembacaan Tabel 4.8 dan 4.9, khususnya pembacaan nilai

BAB 4 IMPLEMENTASI DAN PENGUJIAN

akurasi, dibuat sebuah grafik yang ditunjukkan pada Gambar 4.56 hingga 4.59. Grafik tersebut menggambarkan nilai rata-rata akurasi, akurasi validasi, dan selisih antara keduanya. Berikut adalah gambar-gambar tersebut. Kesimpulan mengenai tabel dan gambar dapat dilihat pada bagian setelah Gambar 4.59.



images/3-sum-0.001.png

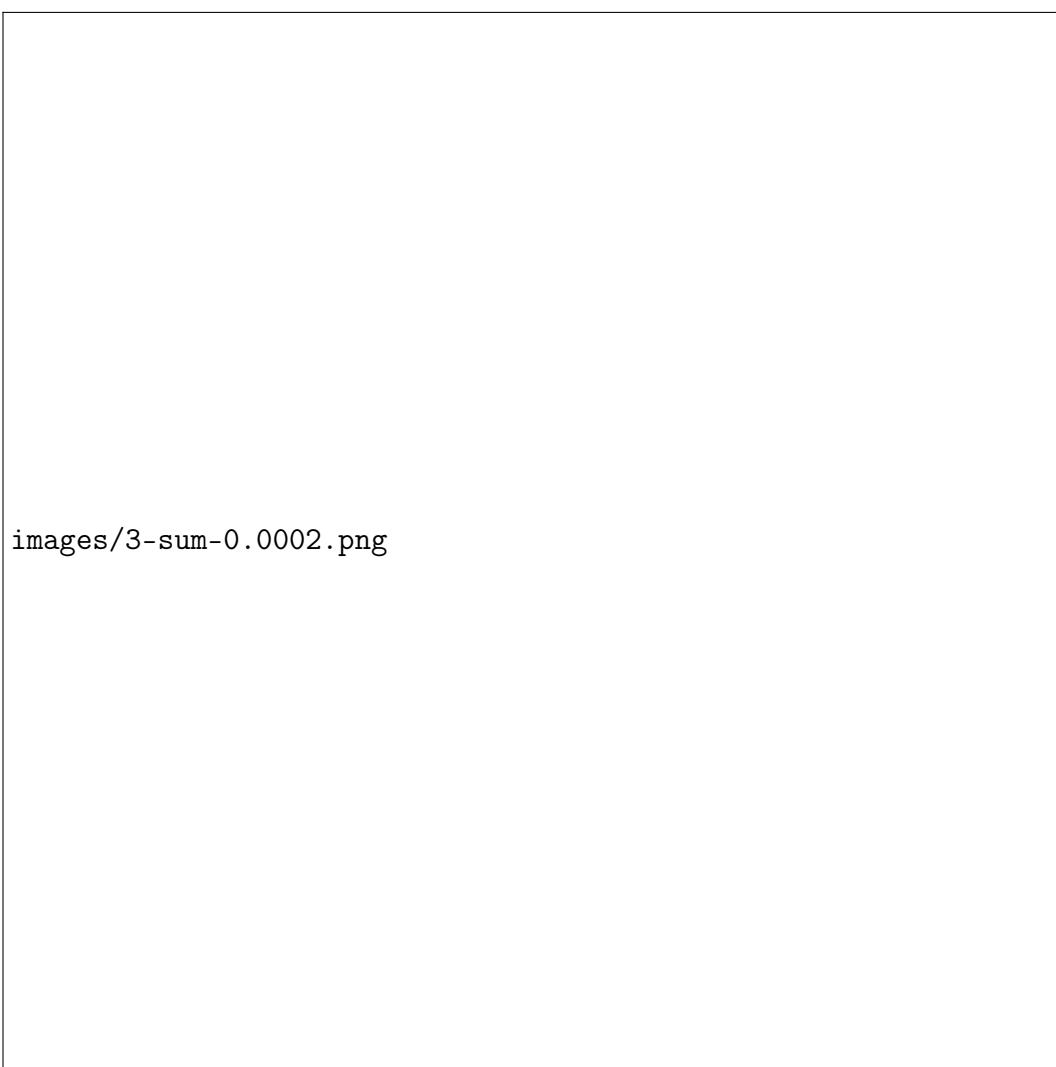
Gambar 4.56 Grafik akurasi, akurasi validasi, dan selisihnya pada pengujian di arsitektur ketiga dengan $learning\ rate = 0.001 (10^{-3})$

BAB 4 IMPLEMENTASI DAN PENGUJIAN



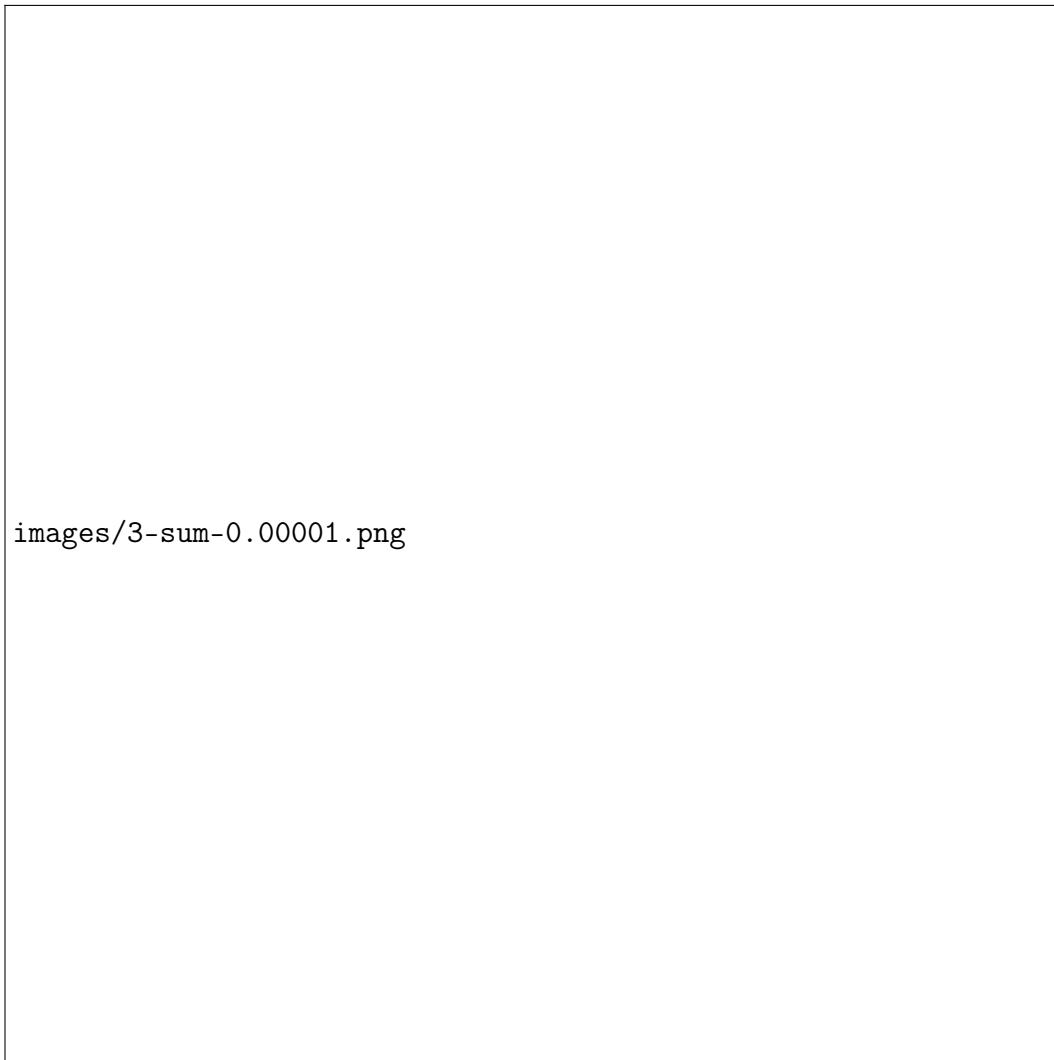
Gambar 4.57 Grafik akurasi, akurasi validasi, dan selisihnya pada pengujian di arsitektur ketiga dengan $learning\ rate = 0.0001 (10^{-4})$

BAB 4 IMPLEMENTASI DAN PENGUJIAN



images/3-sum-0.0002.png

Gambar 4.58 Grafik akurasi, akurasi validasi, dan selisihnya pada pengujian di arsitektur ketiga dengan $learning\ rate = 0.0002 (2 \times 10^{-4})$



Gambar 4.59 Grafik akurasi, akurasi validasi, dan selisihnya pada pengujian di arsitektur ketiga dengan $learning\ rate = 0.00001 (10^{-5})$

Berdasarkan hasil pengujian dengan arsitektur ketiga yang ditunjukkan pada Tabel 4.8 dan 4.9 serta yang digambarkan pada Gambar 4.56 hingga Gambar 4.59, dapat diambil kesimpulan tentang beberapa hal berikut.

1. Dengan jumlah konvolusi yang berkurang dan dengan *filter* yang banyak, nilai *epoch* tidak perlu terlalu banyak untuk menghindari model yang terlalu banyak belajar yang akan mengakibatkan *overfitting*.
2. Nilai *learning rate* $0.001 (10^{-3})$ dan $0.0002 (2 \times 10^{-4})$ kurang cocok digunakan pada arsitektur ini karena selisih rata-rata antara akurasi dan akurasi validasi yang jauh.
3. Nilai *learning rate* $0.0001 (10^{-4})$ dapat menghasilkan akurasi yang tinggi dan dapat dinyatakan memiliki performa yang baik pada arsitektur ini. Namun, dengan nilai *learning rate* tersebut, model sedikit mengalami *overfitting* karena

selisih rata-rata antara akurasi dan akurasi validasi tidak sekecil *learning rate* $0.00001 (10^{-5})$.

4. Nilai *learning rate* $0.00001 (10^{-5})$ memiliki performa yang kurang baik pada arsitektur ini. Meskipun jika dilihat dari grafik akurasi dan akurasi validasinya tidak pernah ada laju pergerakan yang terlalu ekstrem, namun akurasi yang dihasilkan tidak terlalu tinggi dan selalu berada di bawah akurasi yang dihasilkan oleh kombinasi *learning rate* yang lainnya.

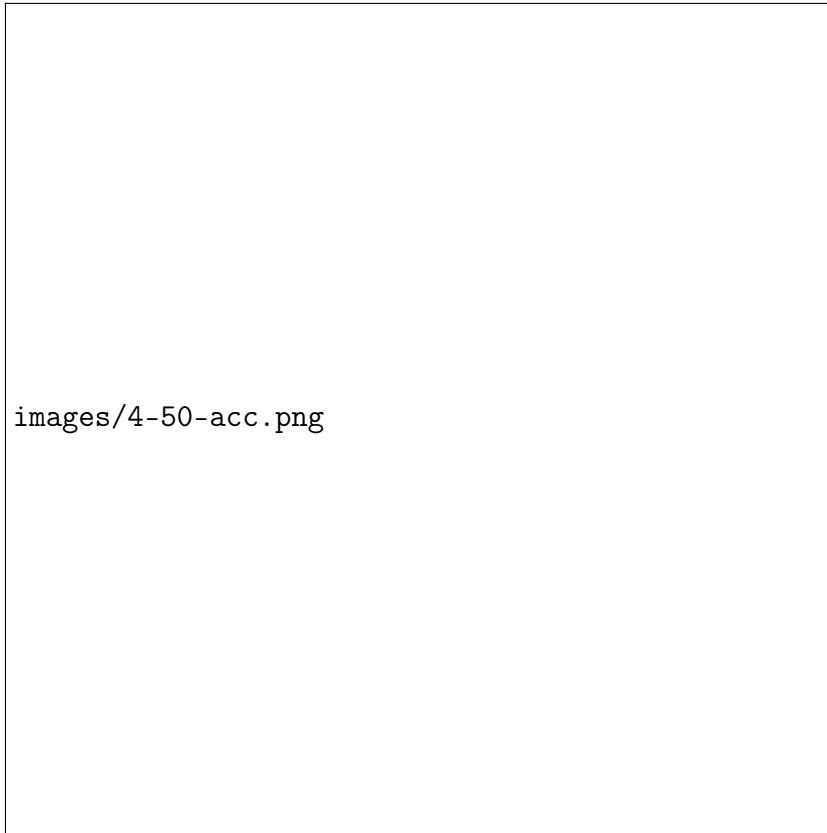
4.5.4 Pengujian pada Arsitektur 4

Bagian ini akan merangkum hasil pengujian pada arsitektur keempat. Penjelasan dibagi per *epoch* dan *learning rate*. Tabel yang merangkum hasil pengujian ada pada Tabel 4.10 dan 4.11 di bagian 4.5.4.7.

4.5.4.1 Nilai *Epoch* = 50

Grafik pengujian dengan nilai *epoch* 50 untuk masing-masing nilai *learning rate* dapat dilihat pada Gambar 4.60 untuk akurasi pengujian dan Gambar 4.61 untuk akurasi validasi pengujian. Dapat dilihat pada kedua gambar berikut, nilai *learning rate* memiliki pengaruh kenaikan akurasi sekaligus memengaruhi kestabilan model. Model dengan *learning rate* $0.001 (10^{-3})$ cenderung tidak stabil, dapat dilihat pada grafik akurasi validasinya. Hal ini menandakan bahwa *learning rate* tersebut kurang cocok digunakan pada kombinasi *epoch* dan arsitektur ini. Selain *learning rate*, jumlah *filter* juga memiliki pengaruh terhadap akurasi. Semakin banyak *filter* yang digunakan, model dapat belajar lebih banyak fitur. Pada arsitektur ini, jumlah konvolusi sama dengan pada arsitektur ketiga, namun jumlah *filter* dikurangi. Dengan *filter* yang lebih sedikit, fitur yang dihasilkan lebih sedikit, sehingga dibutuhkan pengujian dengan nilai *epoch* yang lebih banyak.

BAB 4 IMPLEMENTASI DAN PENGUJIAN



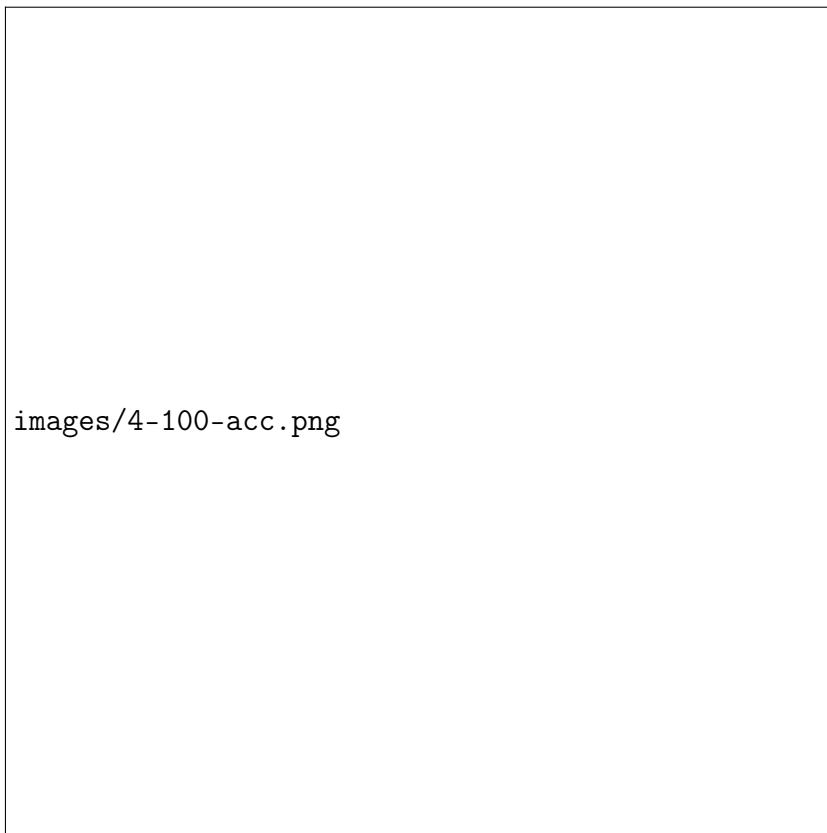
Gambar 4.60 Grafik perubahan akurasi pada pengujian arsitektur 4 untuk $epoch = 50$



Gambar 4.61 Grafik perubahan akurasi validasi pada pengujian arsitektur 4 untuk $epoch = 50$

4.5.4.2 Nilai *Epoch* = 100

Hasil pengujian dalam arsitektur keempat dengan nilai *epoch* 100 dapat dilihat pada Gambar 4.62 dan 4.63. Hasil pengujian menunjukkan bahwa nilai *learning rate* sangat berpengaruh. Pada arsitektur ini, dengan jumlah *epoch* 100, nilai *learning rate* terbaik yang dapat diambil adalah $0.0001 (10^{-4})$ karena menghasilkan akurasi tertinggi dan akurasi validasi yang cenderung stabil. Nilai *learning rate* $0.0002 (2 \times 10^{-4})$ sudah mulai terlihat kurang cocok untuk arsitektur ini, ditunjukkan dengan akurasi validasi yang mulai tidak stabil.



Gambar 4.62 Grafik perubahan akurasi pada pengujian arsitektur 4 untuk *epoch* = 100



images/4-100-valacc.png

Gambar 4.63 Grafik perubahan akurasi validasi pada pengujian arsitektur 4 untuk $epoch = 100$

4.5.4.3 Nilai $Epoch = 150$

Hasil pengujian dengan nilai $epoch$ 150 pada arsitektur keempat dapat dilihat pada Gambar 4.64 dan Gambar 4.65. Bukti bahwa nilai *learning rate* sangat berpengaruh pada laju pergerakan akurasi semakin terlihat pada pengujian ini. Pada nilai *learning rate* 0.0002 (2×10^{-4}) dan 0.001 (10^{-3}), akurasi validasi semakin jelas terlihat ketidakaturannya, yang menandakan bahwa nilai *learning rate* ini terlalu besar untuk $epoch$ 150 pada arsitektur keempat. Pergerakan grafik akurasi masih cenderung naik, sehingga jumlah $epoch$ masih dapat ditambahkan dalam pengujian untuk mencari kapan grafik akurasi akan mulai melandai secara umum.

BAB 4 IMPLEMENTASI DAN PENGUJIAN



Gambar 4.64 Grafik perubahan akurasi pada pengujian arsitektur 4 untuk $epoch = 150$



Gambar 4.65 Grafik perubahan akurasi validasi pada pengujian arsitektur 4 untuk $epoch = 150$

4.5.4.4 Nilai *Epoch* = 200

Gambar 4.66 dan Gambar 4.67 menunjukkan hasil pengujian pada arsitektur keempat dengan nilai *epoch* 200. Selain adanya pengaruh dari *learning rate*, pengaruh nilai *epoch* mulai terlihat juga pada pengujian ini. Model masih menunjukkan peningkatan akurasi, di mana grafik yang dihasilkan belum landai sepenuhnya, tidak seperti pada arsitektur ketiga. Arsitektur ini menggunakan lebih sedikit jumlah *filter*, dan pada kombinasi *epoch* 200, model masih belum belajar terlalu banyak sehingga masih ada peningkatan akurasi. Hal ini menyiratkan bahwa jumlah *filter* juga memengaruhi akurasi. Nilai *epoch* ini dapat dinyatakan tidak cocok untuk *learning rate* 0.0002 (2×10^{-4}) dan 0.001 (10^{-3}) karena menghasilkan akurasi validasi yang tidak stabil.



images/4-200-acc.png

Gambar 4.66 Grafik perubahan akurasi pada pengujian arsitektur 4 untuk *epoch* = 200

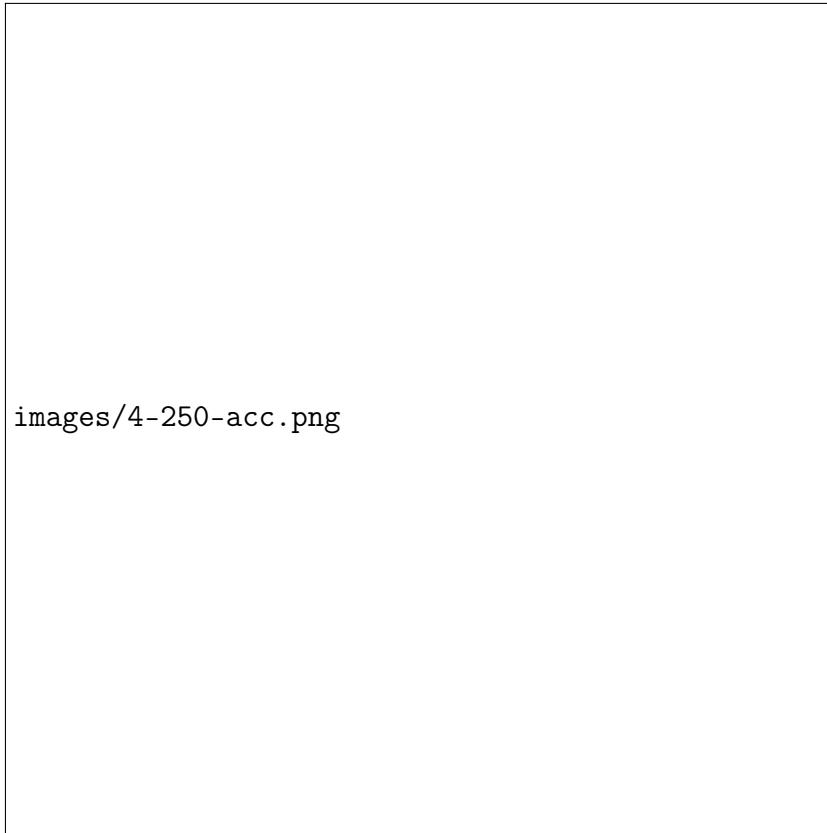


Gambar 4.67 Grafik perubahan akurasi validasi pada pengujian arsitektur 4 untuk $epoch = 200$

4.5.4.5 Nilai *Epoch* = 250

Di bawah ini, ditunjukkan hasil pengujian dengan nilai *epoch* 250 pada arsitektur keempat dengan Gambar 4.68 dan Gambar 4.69. Sejauh ini, dapat diambil kesimpulan bahwa nilai *learning rate* 0.001 (10^{-3}) dan 0.0002 (2×10^{-4}) tidak cocok digunakan pada model, karena meskipun menghasilkan akurasi yang tinggi, model mengalami ketidakstabilan ketika dimasukkan data yang baru. Grafik akurasi validasi juga menegaskan performa model yang tidak baik untuk nilai *epoch* dan *learning rate* tersebut karena semakin tidak beraturan. Grafik akurasi sudah mulai landai, sehingga jika nilai *epoch* ditambahkan, potensi *overfitting* akan bertambah.

BAB 4 IMPLEMENTASI DAN PENGUJIAN



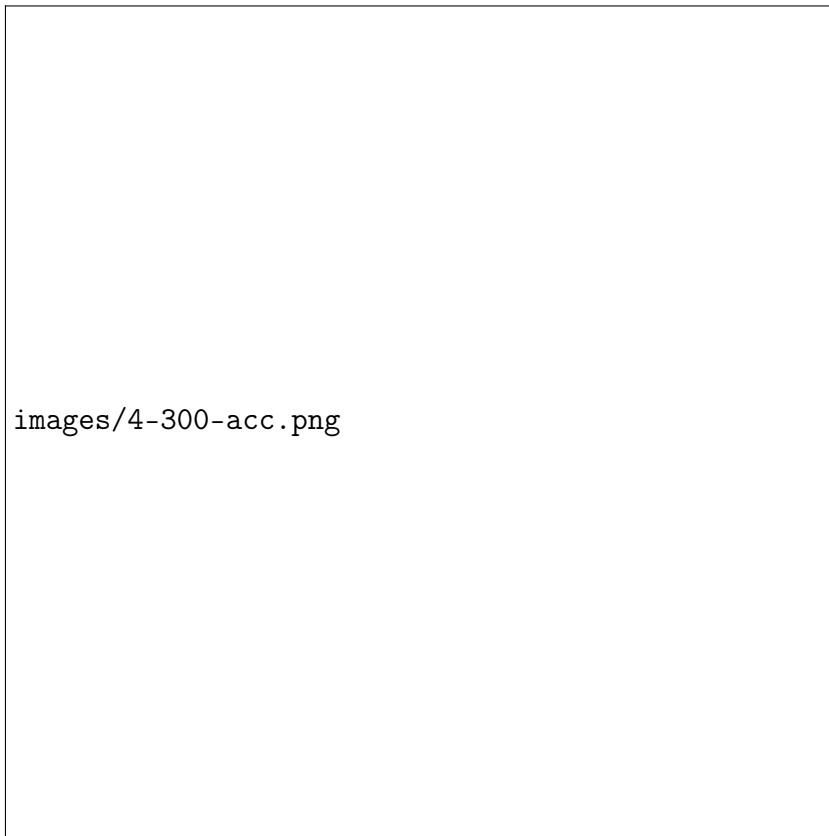
Gambar 4.68 Grafik perubahan akurasi pada pengujian arsitektur 4 untuk $epoch = 250$



Gambar 4.69 Grafik perubahan akurasi validasi pada pengujian arsitektur 4 untuk $epoch = 250$

4.5.4.6 Nilai *Epoch* = 300

Hasil pengujian dengan nilai *epoch* 300 pada arsitektur keempat dapat dilihat pada Gambar 4.70 dan Gambar 4.71. Sebelumnya, telah dinyatakan bahwa nilai *learning rate*, nilai *epoch*, dan jumlah *filter* memiliki pengaruh terhadap laju pergerakan grafik akurasi dan akurasi validasi. Ketika nilai *epoch* optimal, grafik akan mulai berbentuk landai. Pada pengujian dengan arsitektur ini, jumlah *epoch* 300 sudah dapat dinyatakan cukup. Dapat disimpulkan bahwa dengan menggunakan arsitektur ini, *learning rate* $0.00001 (10^{-5})$ menghasilkan model yang stabil, namun tidak dapat mencapai akurasi yang tinggi. Nilai *learning rate* $0.001 (10^{-3})$ dan $0.0002 (2 \times 10^{-4})$ tetap tidak cocok digunakan jika mempertimbangkan laju perubahan pada grafik akurasi validasi. *Learning rate* $0.0001 (10^{-4})$ tidak menunjukkan adanya ketidakstabilan, namun mengalami sedikit *overfitting*.



images/4-300-acc.png

Gambar 4.70 Grafik perubahan akurasi pada pengujian arsitektur 4 untuk *epoch* = 300



images/4-300-valacc.png

Gambar 4.71 Grafik perubahan akurasi validasi pada pengujian arsitektur 4 untuk $epoch = 300$

4.5.4.7 Tabel Hasil Pengujian pada Arsitektur 4

Tabel 4.10 dan 4.11 merangkum seluruh hasil pengujian estimasi *pitch* yang sudah dilakukan pada arsitektur keempat. Kedua tabel masing-masing berisi nilai *mean* (rata-rata) dan nilai maksimum dari setiap parameter percobaan. Kesimpulan yang diambil dari Tabel 4.10 dan 4.11 dapat dilihat setelah Tabel 4.11.

Tabel 4.10 Hasil Pengujian (nilai rata-rata) pada Arsitektur 4

<i>Learning Rate</i>	<i>Epoch</i>	Hasil Nilai Rata-rata			
		<i>Accuracy</i>	<i>F1 Score</i>	<i>Val Accuracy</i>	$\Delta Accuracy$
0.001 (10^{-3})	50	85.756%	85.591%	49.784%	35.971%
	100	87.400%	87.304%	50.290%	37.110%
	150	88.390%	88.333%	49.191%	39.199%
	200	89.271%	89.220%	51.921%	37.350%
	250	90.022%	89.988%	53.946%	36.076%
	300	90.709%	90.682%	53.369%	37.340%

BAB 4 IMPLEMENTASI DAN PENGUJIAN

Tabel 4.10 Hasil Pengujian (nilai rata-rata) pada Arsitektur 4

<i>Learning Rate</i>	<i>Epoch</i>	Hasil Nilai Rata-rata			
		<i>Accuracy</i>	<i>F1 Score</i>	<i>Val Accuracy</i>	$\Delta Accuracy$
0.0001 (10^{-4})	50	81.431%	80.773%	79.448%	1.983%
	100	85.360%	85.055%	81.974%	3.386%
	150	87.345%	87.119%	83.033%	4.312%
	200	89.597%	89.427%	83.589%	6.008%
	250	90.939%	90.801%	83.529%	7.410%
	300	92.196%	92.074%	83.823%	8.372%
0.0002 (2×10^{-4})	50	84.087%	83.647%	80.171%	3.916%
	100	87.935%	87.699%	81.930%	6.005%
	150	90.308%	90.146%	79.878%	10.430%
	200	91.984%	91.868%	78.956%	13.028%
	250	93.400%	93.299%	79.676%	13.724%
	300	94.394%	94.305%	79.822%	14.572%
0.00001 (10^{-5})	50	63.022%	57.498%	65.457%	2.434%
	100	73.429%	70.409%	74.207%	0.779%
	150	76.453%	74.335%	76.807%	0.354%
	200	78.364%	76.785%	78.528%	0.164%
	250	79.957%	78.572%	79.514%	0.443%
	300	81.077%	79.873%	80.628%	0.449%

Tabel 4.11 Hasil Pengujian (nilai maksimal) pada Arsitektur 4

<i>Learning Rate</i>	<i>Epoch</i>	Hasil Nilai Maksimal			
		<i>Accuracy</i>	<i>F1 Score</i>	<i>Val Accuracy</i>	$\Delta Accuracy$
0.001 (10^{-3})	50	88.460%	88.470%	78.720%	9.740%
	100	90.160%	90.160%	75.810%	14.350%

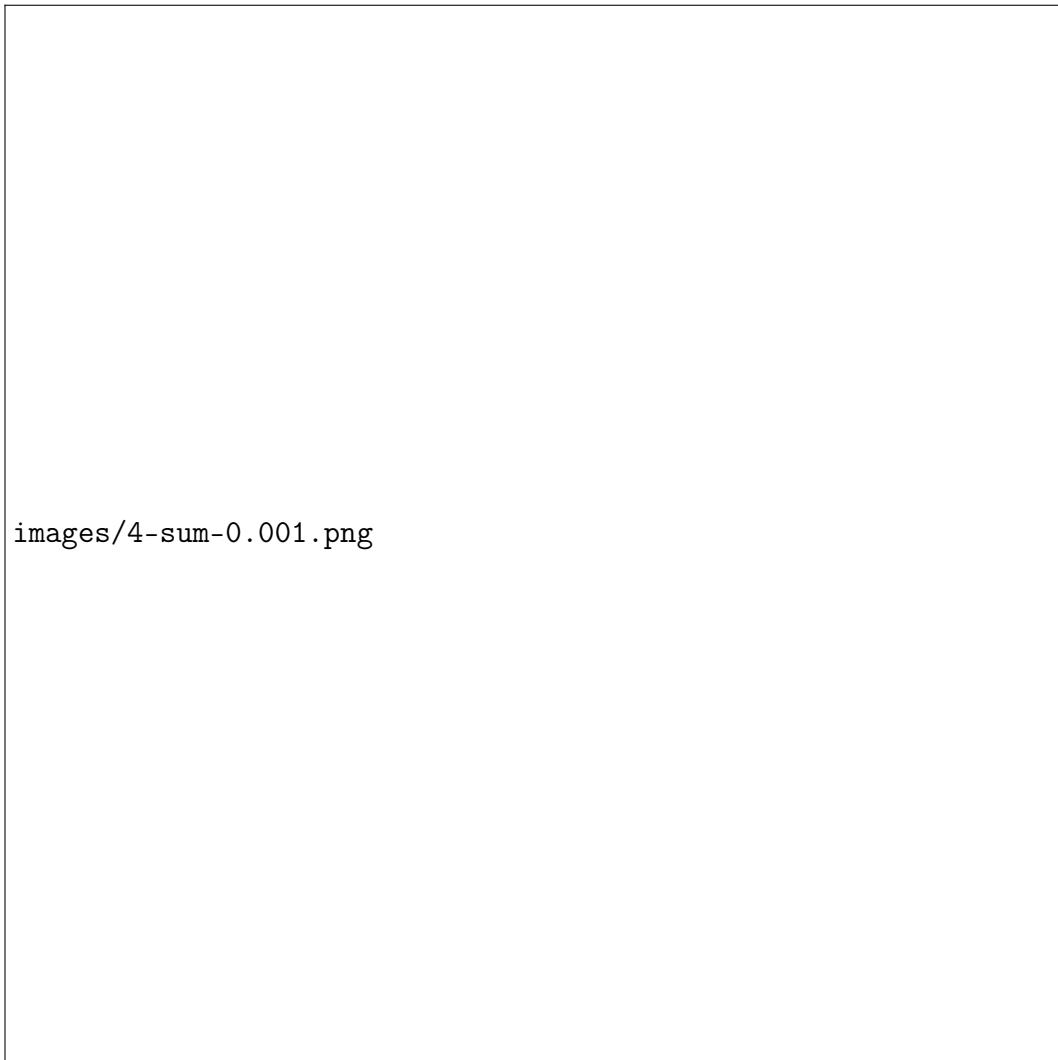
Tabel 4.11 Hasil Pengujian (nilai maksimal) pada Arsitektur 4

<i>Learning Rate</i>	<i>Epoch</i>	Hasil Nilai Maksimal			
		<i>Accuracy</i>	<i>F1 Score</i>	<i>Val Accuracy</i>	Δ <i>Accuracy</i>
0.0001 (10^{-4})	150	91.180%	91.190%	77.680%	13.500%
	200	92.740%	92.740%	78.840%	13.900%
	250	94.160%	94.110%	78.680%	15.480%
	300	95.450%	95.460%	83.310%	12.140%
0.0002 (2×10^{-4})	50	87.930%	87.910%	85.030%	2.900%
	100	90.590%	90.650%	85.550%	5.040%
	150	93.250%	93.260%	85.820%	7.430%
	200	96.440%	96.410%	85.690%	10.750%
	250	98.390%	98.350%	85.940%	12.450%
	300	99.250%	99.180%	85.910%	13.340%
0.00001 (10^{-5})	50	89.520%	89.510%	84.870%	4.650%
	100	93.600%	93.560%	85.330%	8.270%
	150	96.770%	96.730%	85.040%	11.730%
	200	98.730%	98.720%	85.350%	13.380%
	250	99.340%	99.330%	85.350%	13.990%
	300	99.760%	99.770%	84.990%	14.770%

Untuk mempermudah pembacaan Tabel 4.10 dan 4.11, khususnya pembacaan nilai

BAB 4 IMPLEMENTASI DAN PENGUJIAN

akurasi, dibuat sebuah grafik yang ditunjukkan pada Gambar 4.72 hingga 4.75. Grafik tersebut menggambarkan nilai rata-rata akurasi, akurasi validasi, dan selisih antara keduanya. Berikut adalah gambar-gambar tersebut. Kesimpulan mengenai tabel dan gambar dapat dilihat pada bagian setelah Gambar 4.75.



images/4-sum-0.001.png

Gambar 4.72 Grafik akurasi, akurasi validasi, dan selisihnya pada pengujian di arsitektur keempat dengan $learning\ rate = 0.001 (10^{-3})$

BAB 4 IMPLEMENTASI DAN PENGUJIAN



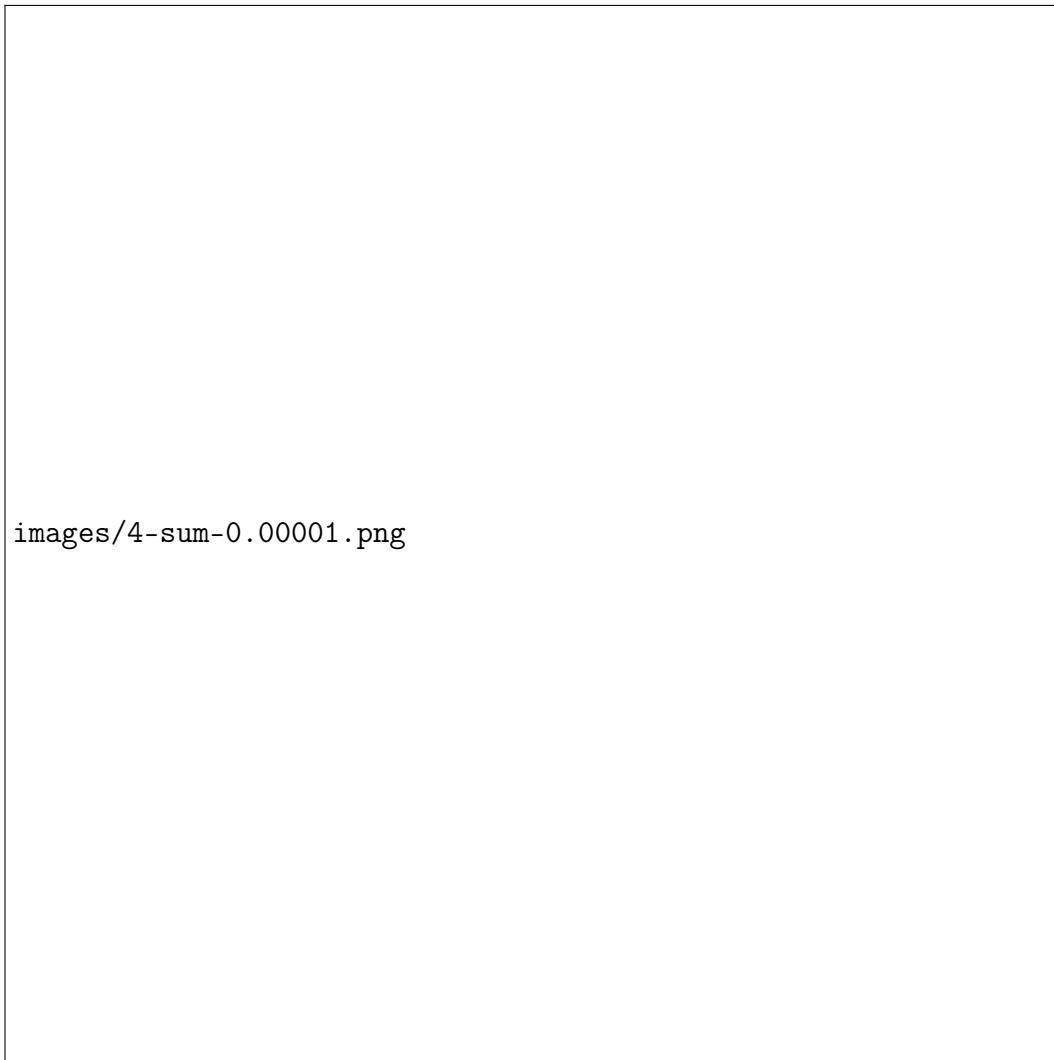
Gambar 4.73 Grafik akurasi, akurasi validasi, dan selisihnya pada pengujian di arsitektur keempat dengan $learning\ rate = 0.0001 (10^{-4})$

BAB 4 IMPLEMENTASI DAN PENGUJIAN



images/4-sum-0.0002.png

Gambar 4.74 Grafik akurasi, akurasi validasi, dan selisihnya pada pengujian di arsitektur keempat dengan $learning\ rate = 0.0002 (2 \times 10^{-4})$



images/4-sum-0.00001.png

Gambar 4.75 Grafik akurasi, akurasi validasi, dan selisihnya pada pengujian di arsitektur keempat dengan $learning\ rate = 0.00001 (10^{-5})$

Berdasarkan hasil pengujian dengan arsitektur keempat yang ditunjukkan pada Tabel 4.10 dan 4.11 serta yang digambarkan pada Gambar 4.72 hingga Gambar 4.75, dapat diambil kesimpulan tentang beberapa hal berikut.

1. Dengan jumlah konvolusi dan *filter* yang kurang, nilai *epoch* perlu ditambah untuk menghasilkan akurasi yang tinggi. Dengan lebih banyak *epoch*, fitur yang dihasilkan akan lebih banyak.
2. Nilai $learning\ rate$ $0.001 (10^{-3})$ dan $0.0002 (2 \times 10^{-4})$ kurang cocok digunakan pada arsitektur ini karena selisih rata-rata antara akurasi dan akurasi validasi yang jauh.
3. Nilai $learning\ rate$ $0.0001 (10^{-4})$ dan $0.00001 (10^{-5})$ dapat menghasilkan akurasi yang tinggi dan dapat dinyatakan memiliki performa yang baik pada arsitektur ini. Namun, dengan nilai $learning\ rate$ $0.0001 (10^{-4})$, model sedikit

mengalami *overfitting* karena selisih rata-rata antara akurasi dan akurasi validasi tidak sekecil *learning rate* $0.00001 (10^{-5})$. Sementara itu, dengan nilai $0.00001 (10^{-5})$, laju pertambahan akurasi cenderung lambat.

4.5.5 Pembahasan Umum Hasil Pengujian Arsitektur CNN

Dari seluruh hasil pengujian arsitektur CNN yang sudah dituliskan pada bagian 4.5.1 hingga 4.5.4, jika dikaitkan dengan rumusan masalah pada bagian 1.2 , maka keterkaitan jumlah konvolusi, jumlah *filter*, *epoch*, dan *learning rate* secara menyeluruh terhadap akurasi sistem dapat dijelaskan sebagai berikut.

1. Semakin banyak dilakukan konvolusi dengan jumlah *filter* yang optimal pada masing-masing *convolutional layer*, arsitektur yang dibangun dapat dinyatakan semakin baik untuk melakukan estimasi *pitch*. Jumlah *filter* menunjukkan jumlah fitur yang diekstrak dari suatu data. Maka dari itu, jika *filter* semakin banyak, fitur yang dapat diekstrak dari data juga semakin banyak. Sementara itu, jumlah konvolusi menentukan berapa kali pengambilan fitur diambil pada suatu data. Kombinasi yang tepat antara jumlah konvolusi dengan jumlah *filter* akan menghasilkan akurasi yang baik. Meskipun begitu, kedua parameter tersebut belum cukup untuk menentukan hasil model mana yang memiliki hasil terbaik. Misalnya, pada arsitektur pertama, jumlah keseluruhan *filter* adalah 4608 dan jumlah keseluruhan *filter* pada arsitektur keempat adalah 1536. Secara konsep, fitur akan lebih banyak diambil pada arsitektur pertama karena memiliki jumlah *filter* yang lebih banyak. Namun, karena ada parameter lain yang perlu dipertimbangkan (*epoch* dan *learning rate*), maka belum dapat diketahui secara pasti apakah arsitektur pertama sudah pasti menghasilkan akurasi yang lebih baik, karena ada sebuah kasus di mana arsitektur keempat dapat menghasilkan rata-rata akurasi yang lebih tinggi daripada arsitektur pertama (kombinasi nilai *epoch* = 100 dan *learning rate* = $0.001 (10^{-3})$, di mana pada arsitektur keempat menghasilkan rata-rata akurasi sebesar 87.4% dengan selisih rata-rata akurasi dan rata-rata akurasi validasi sebesar 37.11% dan pada arsitektur pertama menghasilkan rata-rata akurasi sebesar 87.06% dengan selisih rata-rata akurasi dan rata-rata akurasi validasi sebesar 51.402%). Maka dari itu, perlu dilakukan pencarian nilai *epoch* dan *learning rate* yang optimal untuk dapat memastikan bahwa jumlah konvolusi dan *filter* pada setiap arsitektur dinyatakan menghasilkan akurasi yang baik .
2. Ketika sudah ditemukan jumlah konvolusi dan jumlah *filter* yang optimal pada masing-masing *convolutional layer*, hal lain yang perlu diperhatikan adalah *epoch*, dan *learning rate* yang menentukan iterasi pelatihan model dan seberapa

cepat model diharapkan mencapai konvergensi yang baik. Misalnya, pada arsitektur pertama, ditemukan nilai *epoch* = 300 dan nilai *learning rate* = 0.0001 (10^{-4}) cocok digunakan karena model yang dihasilkan memiliki akurasi yang tinggi (97.425%) dan selisih antara rata-rata akurasi dan rata-rata akurasi validasi yang tidak terlalu jauh (14.383%). Namun, nilai itu belum tentu cocok digunakan pada arsitektur yang lain dengan nilai *epoch* dan *learning rate* yang sama. Misalnya, pada arsitektur keempat, dengan nilai *epoch* dan *learning rate* yang sama, dihasilkan akurasi yang lebih rendah (94.394%) dan selisih antara rata-rata akurasi dan rata-rata akurasi validasi yang lebih tinggi (14.572%).

Secara umum, dapat diambil kesimpulan bahwa kombinasi yang tepat dari jumlah konvolusi, jumlah *filter*, *epoch*, dan *learning rate* akan menghasilkan akurasi sistem yang terbaik. Namun, dari pembahasan hasil arsitektur tersebut, untuk menentukan keputusan pengambilan hasil model yang terbaik, pada dasarnya bergantung dari sudut pandang kepada hasil pengujian. Model yang terbaik tidak dapat ditentukan hanya dalam sebuah kesimpulan tunggal karena ada beberapa hal yang dapat dipertimbangkan lebih lanjut sebelum menyatakan suatu model X merupakan model yang terbaik. Misalnya, jika yang dibutuhkan hanya rata-rata akurasi tinggi, maka nilai rata-rata akurasi validasi dapat diabaikan. Sebaliknya, jika akurasi validasi dipertimbangkan, dalam arti bahwa akurasi validasi tidak boleh melebihi suatu batas nilai (misalnya 1%-5%), maka harus dicari sebuah model yang selisih rata-rata akurasi dan rata-rata akurasi validasinya tidak terlalu jauh dan masih memiliki rata-rata akurasi yang relatif tinggi. Pemikiran ini digunakan dalam pemilihan model terbaik dari seluruh hasil pengujian yang sudah dilakukan untuk pengujian tahap selanjutnya, yaitu pengujian *threshold* dengan metode DTW.

4.5.6 Hasil Pengujian *Threshold* dengan Metode DTW

Berdasarkan hasil pengujian pada arsitektur CNN, akan dipilih beberapa buah kombinasi *epoch* dan *learning rate* yang terbaik untuk dapat dilakukan percobaan pengambilan *threshold*. Pengambilan model terbaik bergantung dari sudut pandang kepada hasil pengujian yang dilakukan, seperti yang sudah dituliskan pada bagian 4.5.5. Maka, keputusan pengambilan model terbaik sebelum melakukan pengujian *threshold* dengan DTW didasarkan kepada dua pertimbangan berikut ini.

1. Model memiliki rata-rata akurasi yang tinggi. Selisih antara rata-rata akurasi dan akurasi validasi tidak diperhatikan, sehingga model dibiarkan mengalami sedikit *overfitting*.

2. Model memiliki rata-rata akurasi yang tidak terlalu tinggi, namun memiliki selisih rata-rata akurasi dan validasi yang kecil. Hal ini berarti mencegah penggunaan model yang *overfitting*.

Berdasarkan pertimbangan tersebut, dipilih dua buah model untuk pengujian *threshold* dengan DTW, yaitu:

1. Model CNN pada arsitektur pertama, *epoch* = 300, *learning rate* = 0.0001 (10^{-4}), dengan rata-rata akurasi 97.425%.
2. Model CNN pada arsitektur keempat, *epoch* = 300, *learning rate* = 0.00001 (10^{-5}), dengan selisih rata-rata akurasi dan akurasi validasi sebesar 0.449% dan rata-rata akurasi sebesar 81.077%.

4.5.6.1 Pengujian pada Model Pertama

Tabel 4.12 menunjukkan hasil pengujian dengan *dynamic time warping* pada model CNN yang pertama. Nilai minimal menunjukkan *distance* terkecil dari seluruh rekaman lagu dengan judul terkait, sedangkan nilai maksimal menunjukkan *distance* terbesar.

Tabel 4.12 Hasil Pengujian *Distance* dengan Model Pertama

No	Judul Lagu	Jumlah Data	Hasil <i>Distance</i>		
			Minimal	Maksimal	Rata-rata
1	Twinkle, Twinkle, Little Star	32	2	2871	1513.813
2	Old MacDonald Had a Farm	31	132	3796	2203.710
3	Happy Birthday	34	297	3856	2443.235
4	Brother John (Are you sleeping?)	35	185	3732	2170.629
5	London Bridge Is Falling Down	29	10	3804	1820.862

Berdasarkan nilai rata-rata, akan diujikan beberapa nilai *threshold* untuk melihat perbandingan data yang dinyatakan mirip dan tidak mirip. Hasil pengujian dengan masing-masing *threshold* tersebut dapat dilihat pada Tabel 4.13. Kolom M menunjukkan jumlah data yang dianggap mirip karena tidak melebihi *threshold*. Kolom TM menunjukkan data yang melebihi *threshold* dan dianggap tidak mirip.

Tabel 4.13 Hasil Pengujian *Threshold* pada Model Pertama

No	Judul Lagu	Nilai <i>Threshold</i>									
		500		1000		1500		2000		2500	
		M	TM	M	TM	M	TM	M	TM	M	TM
1	Twinkle, Twinkle, Little Star	1	31	8	24	16	16	23	9	29	3
2	Old MacDonald Had a Farm	1	30	4	27	9	22	13	18	17	14
3	Happy Birthday	2	32	4	30	8	26	10	24	14	20
4	Brother John (Are you sleeping?)	1	34	6	29	7	28	10	25	20	15
5	London Bridge Is Falling Down	3	26	4	25	9	20	16	13	24	5

4.5.6.2 Pengujian pada Model Kedua

Tabel 4.14 menunjukkan hasil pengujian dengan *dynamic time warping* pada model CNN yang kedua. Nilai minimal merupakan *distance* terkecil pada data lagu tersebut, sedangkan nilai maksimal merupakan *distance* terbesar.

Tabel 4.14 Hasil Pengujian *Distance* dengan Model Kedua

No	Judul Lagu	Jumlah Data	Hasil <i>Distance</i>		
			Minimal	Maksimal	Rata-rata
1	Twinkle, Twinkle, Little Star	32	112	2975	1523.188
2	Old MacDonald Had a Farm	31	133	4093	2154.226
3	Happy Birthday	34	147	3885	2411.176
4	Brother John (Are you sleeping?)	35	25	3738	2125.914
5	London Bridge Is Falling Down	29	9	3952	1980.207

Dari nilai rata-rata yang telah ditampilkan pada Tabel 4.14, akan diujikan beberapa nilai *threshold* untuk melihat perbandingan data yang dinyatakan mirip dan tidak mirip. Hasil pengujian dengan masing-masing *threshold* tersebut dapat dilihat pada Tabel 4.15. Kolom M menunjukkan jumlah data yang dianggap mirip karena tidak melebihi *threshold*. Kolom TM menunjukkan data yang melebihi *threshold* dan dianggap tidak mirip.

Tabel 4.15 Hasil Pengujian *Threshold* pada Model Kedua

No	Judul Lagu	Nilai <i>Threshold</i>									
		500		1000		1500		2000		2500	
		M	TM	M	TM	M	TM	M	TM	M	TM
1	Twinkle, Twinkle, Little Star	2	30	6	26	17	15	22	10	29	3
2	Old MacDonald Had a Farm	1	30	6	25	9	22	13	18	16	15
3	Happy Birthday	2	32	4	30	8	26	10	24	13	21
4	Brother John (Are you sleeping?)	4	31	7	28	7	28	11	24	21	14
5	London Bridge Is Falling Down	1	28	4	25	7	22	14	15	22	7

4.5.7 Pembahasan Hasil Pengujian *Threshold*

Berdasarkan hasil pengujian *threshold* pada Tabel 4.13 dan 4.15, dapat diambil beberapa kesimpulan berikut ini.

1. Nilai *threshold* yang dapat digunakan untuk melakukan pengujian kemiripan dengan optimal berada di rentang 1000-1500.
2. Nilai tersebut dipilih karena dinilai dapat melakukan pengujian kemiripan secara baik, di mana data rekaman dengan kualitas *pitch* kurang baik dapat dikategorikan sebagai rekaman yang tidak mirip dengan *template* pada *dataset*.
3. Jika diambil nilai *threshold* yang terlalu tinggi, maka rekaman yang memiliki kualitas *pitch* kurang baik dapat dianggap mirip. Hal ini akan berdampak pada performa sistem yang kurang baik karena dengan begitu, kualitas penyanyi yang baik tidak dapat ditentukan.
4. Jika diambil nilai *threshold* yang terlalu rendah, maka rekaman masukan cenderung harus identik dengan *template* pada *dataset*, di mana hal ini sangat sulit dicapai pada proses rekaman yang sesungguhnya.

Pengambilan nilai *threshold* pada rentang 1000-1500 dilakukan dengan cara melakukan perbandingan antara hasil estimasi *pitch* dengan data *pitch* dari *dataset*. Perbandingan ini dilakukan untuk melihat berapa banyak *pitch* yang terestimasi dengan tepat. Selain itu, akan dilihat pula berapa banyak jumlah *pitch* yang tidak terestimasi (data yang kurang) dan yang seharusnya tidak terestimasi (data yang lebih). Secara lengkap, hasil pengujian *threshold* ini dapat dilihat pada Lampiran C di Tabel C-1 sampai Tabel C-10 dan Lampiran D di Tabel D-1 sampai Tabel D-10. Pada bagian ini, hanya akan ditampilkan hasil perbandingan.

Berikut ini adalah empat contoh perbandingan hasil estimasi *pitch* dan *pitch* pada *dataset* menggunakan lagu Twinkle-Twinkle Little Star dan Old McDonald Had a Farm pada model pertama dan kedua. Seperti yang sudah dituliskan sebelumnya, tabel perbandingan untuk seluruh lagu secara lengkap dapat dilihat pada Lampiran D.

Tabel 4.16 menunjukkan hasil perbandingan hasil estimasi *pitch* dan *pitch* pada *dataset* dari pengujian lagu Twinkle-Twinkle Little Star pada model pertama. Sekuens *pitch* di *dataset* untuk lagu ini adalah [0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58] dengan panjang data = 14. Berikut ini adalah keterangan setiap kolom pada tabel.

1. *Pitch result* merupakan hasil estimasi *pitch* dari rekaman.
2. *Distance* merupakan hasil *distance* antara rekaman dari *dataset* dengan rekaman yang diuji.
3. *Match* menunjukkan jumlah *pitch* pada *pitch result* yang ada di hasil estimasi dan *dataset*.
4. *Diff1* merupakan jumlah *pitch* pada *pitch result* yang tidak terestimasi pada hasil estimasi.
5. *Diff2* merupakan jumlah *pitch* pada *pitch result* yang terestimasi, padahal seharusnya tidak terestimasi.
6. *Percentage* menunjukkan perhitungan yang digunakan untuk melihat seberapa baik hasil estimasi *pitch* pada data uji. Perhitungan nilai persentase ini dilakukan berdasarkan hasil eksperimen dengan *dataset* yang digunakan, di mana perhitungan ini belum tentu cocok jika digunakan dengan *dataset* atau penelitian lain. Perhitungan dilakukan dengan persamaan berikut.

$$\frac{match - diff1 - diff2}{panjangdataset} \times 100\%$$

Setelah dihitung dengan persamaan tersebut, hasil perhitungan kemudian diskalakan ke nilai 0 sampai 1 (0% hingga 100%) agar tidak muncul angka persentase negatif dengan menggunakan persamaan *min-max normalization*. Persamaan *min-max normalization* dituliskan sebagai berikut.

BAB 4 IMPLEMENTASI DAN PENGUJIAN

$$v'_i = \frac{v_i - \min_A}{\max_A - \min_A} \times (\text{new_max}_A - \text{new_min}_A) + \text{new_min}_A$$

Keterangan :

- v'_i : hasil penskalaan
- v_i : nilai yang akan diskalakan
- \min_A : nilai minimal dari seluruh nilai yang akan diskalakan
- \max_A : nilai maksimal dari seluruh nilai yang akan diskalakan
- new_min_A : nilai minimal penskalaan
- new_max_A : nilai maksimal penskalaan

Tabel 4.16 Hasil Perbandingan *Pitch* pada lagu Twinkle-Twinkle Little Star di Model Pertama

No	<i>Pitch result</i>	Distance	Match	Diff1	Diff2	Percentage
1	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58]	2	14	0	0	100.00%
2	[0, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57]	521	13	1	2	90.48%
3	[0, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59]	550	12	2	1	88.10%
4	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55]	637	11	3	0	85.71%
5	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56]	646	12	2	0	90.48%
6	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56]	739	12	2	0	90.48%
7	[0, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54]	860	10	4	2	76.19%
8	[0, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55]	999	11	3	1	83.33%
9	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57]	1055	13	1	0	95.24%

BAB 4 IMPLEMENTASI DAN PENGUJIAN

Tabel 4.16 Hasil Perbandingan *Pitch* pada lagu Twinkle-Twinkle Little Star di Model Pertama

No	<i>Pitch result</i>	Distance	Match	Diff1	Diff2	Percentage
10	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56]	1150	12	2	0	90.48%
11	[0, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55]	1222	11	3	1	83.33%
12	[0, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55]	1250	11	3	3	78.57%
13	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57]	1298	13	1	0	95.24%
14	[0, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52]	1299	8	6	4	61.90%
15	[0, 44, 45, 46, 48, 49, 50, 51, 52, 53, 54, 55, 56]	1370	11	3	2	80.95%
16	[0, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55]	1450	11	3	2	80.95%
17	[0, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 59, 60]	1557	12	2	3	83.33%
18	[0, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 53]	1680	8	6	6	57.14%
19	[0, 43, 44, 45, 46, 47, 48, 49, 50, 53, 54]	1707	8	6	3	64.29%
20	[0, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 56]	1817	11	3	5	73.81%
21	[0, 48, 49, 50, 52, 53, 54, 55, 56, 57, 58, 59, 60]	1819	11	3	2	80.95%
22	[0, 47, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63]	1847	7	7	5	54.76%
23	[0, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54]	1931	10	4	3	73.81%
24	[0, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64]	2023	6	8	6	47.62%
25	[0, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55]	2054	11	3	2	80.95%

BAB 4 IMPLEMENTASI DAN PENGUJIAN

Tabel 4.16 Hasil Perbandingan *Pitch* pada lagu Twinkle-Twinkle Little Star di Model Pertama

No	<i>Pitch result</i>	<i>Distance</i>	Match	Diff1	Diff2	Percentage
26	[0, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53]	2078	9	5	4	66.67%
27	[0, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62]	2217	8	6	4	61.90%
28	[0, 54, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66]	2343	4	10	8	33.33%
29	[0, 45, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66]	2370	4	10	9	30.95%
30	[0, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68]	2502	3	11	10	23.81%
31	[0, 45, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66]	2578	4	10	9	30.95%
32	[0, 42, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64]	2871	6	8	7	45.24%

Tabel 4.17 menunjukkan hasil perbandingan hasil estimasi *pitch* dan *pitch* pada *dataset* dari pengujian lagu Twinkle-Twinkle Little Star pada model kedua. Keterangan pada kolom di Tabel 4.17 sama dengan Tabel 4.16.

Tabel 4.17 Hasil Perbandingan *Pitch* pada lagu Twinkle-Twinkle Little Star di Model Kedua

No	<i>Pitch result</i>	<i>Distance</i>	Match	Diff1	Diff2	Percentage
1	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57]	112	13	1	0	95.24%
2	[0, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56]	489	12	2	2	85.71%
3	[0, 48, 49, 50, 52, 53, 54, 55, 56, 57, 58, 59]	570	11	3	1	83.33%
4	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55]	676	11	3	0	85.71%
5	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56]	726	12	2	0	90.48%

BAB 4 IMPLEMENTASI DAN PENGUJIAN

Tabel 4.17 Hasil Perbandingan *Pitch* pada lagu Twinkle-Twinkle Little Star di Model Kedua

No	Pitch result	Distance	Match	Diff1	Diff2	Percentage
6	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56]	732	12	2	0	90.48%
7	[0, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54]	1033	10	4	2	76.19%
8	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57]	1037	13	1	0	95.24%
9	[0, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55]	1040	11	3	1	83.33%
10	[0, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54]	1070	10	4	3	73.81%
11	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57]	1158	13	1	0	95.24%
12	[0, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55]	1180	11	3	1	83.33%
13	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56]	1214	12	2	0	90.48%
14	[0, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59]	1249	13	1	3	88.10%
15	[0, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52]	1283	8	6	4	61.90%
16	[0, 45, 46, 48, 49, 50, 51, 52, 53, 54, 55, 56]	1459	11	3	1	83.33%
17	[0, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54]	1474	10	4	2	76.19%
18	[0, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52]	1502	8	6	6	57.14%
19	[0, 48, 49, 50, 52, 53, 54, 55, 56, 57, 58, 59]	1742	11	3	1	83.33%
20	[0, 43, 44, 45, 46, 47, 48, 49, 50]	1764	6	8	3	54.76%
21	[0, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54]	1791	10	4	5	69.05%

BAB 4 IMPLEMENTASI DAN PENGUJIAN

Tabel 4.17 Hasil Perbandingan *Pitch* pada lagu Twinkle-Twinkle Little Star di Model Kedua

No	<i>Pitch result</i>	<i>Distance</i>	Match	Diff1	Diff2	Percentage
22	[0, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63]	1804	6	8	5	50.00%
23	[0, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64]	2028	6	8	6	47.62%
24	[0, 44, 45, 46, 47, 48, 49, 50, 51, 52, 54]	2080	9	5	2	71.43%
25	[0, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62]	2118	8	6	4	61.90%
26	[0, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54]	2130	10	4	2	76.19%
27	[0, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53]	2147	9	5	4	66.67%
28	[0, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66]	2399	3	11	8	28.57%
29	[0, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68]	2429	3	11	10	23.81%
30	[0, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66]	2636	4	10	8	33.33%
31	[0, 57, 58, 59, 60, 62, 63, 64, 65, 66]	2695	3	11	7	30.95%
32	[0, 54, 56, 57, 58, 59, 60, 62, 64]	2975	5	9	4	47.62%

Tabel 4.18 menunjukkan hasil perbandingan hasil estimasi *pitch* dan *pitch* pada *dataset* dari pengujian lagu Old McDonald Had a Farm pada model pertama. Sekuens *pitch* di *dataset* untuk lagu ini adalah [0, 56, 57, 58, 59, 60, 62, 63, 64, 65, 66, 67] dengan panjang data = 12. Keterangan pada kolom di Tabel 4.18 sama dengan Tabel 4.16.

BAB 4 IMPLEMENTASI DAN PENGUJIAN

Tabel 4.18 Hasil Perbandingan *Pitch* pada lagu Old McDonald Had a Farm di Model Pertama

No	<i>Pitch result</i>	Distance	Match	Diff1	Diff2	Percentage
1	[0, 52, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67]	132	12	0	2	94.44%
2	[0, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68]	645	12	0	3	91.67%
3	[0, 49, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64]	756	9	3	3	75.00%
4	[0, 53, 54, 55, 56, 57, 59, 60, 61, 62, 63]	817	7	5	4	61.11%
5	[0, 42, 49, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66]	1068	11	1	3	86.11%
6	[0, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62]	1171	7	5	6	55.56%
7	[0, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70]	1181	8	4	4	66.67%
8	[0, 45, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65]	1200	10	2	2	83.33%
9	[0, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64]	1245	9	3	6	66.67%
10	[0, 48, 49, 50, 51, 52, 54, 55, 56, 57, 58, 59, 60]	1648	6	6	7	47.22%
11	[0, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59]	1668	5	7	6	44.44%
12	[0, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60]	1742	6	6	9	41.67%
13	[0, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58]	1836	4	8	8	33.33%
14	[0, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58]	2151	4	8	9	30.56%
15	[0, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57]	2173	3	9	11	19.44%
16	[0, 45, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58]	2461	4	8	10	27.78%

BAB 4 IMPLEMENTASI DAN PENGUJIAN

Tabel 4.18 Hasil Perbandingan *Pitch* pada lagu Old McDonald Had a Farm di Model Pertama

No	<i>Pitch result</i>	<i>Distance</i>	Match	Diff1	Diff2	Percentage
17	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56]	2466	2	10	10	16.67%
18	[0, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52]	2716	1	11	11	8.33%
19	[0, 44, 45, 46, 47, 49, 50, 51, 52, 53, 54, 55]	2736	1	11	11	8.33%
20	[0, 42, 43, 44, 45, 47, 48, 49, 50, 51, 52, 55]	2808	1	11	11	8.33%
21	[0, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53]	2940	1	11	10	11.11%
22	[0, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54]	2990	1	11	14	0.00%
23	[0, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53]	2999	1	11	12	5.56%
24	[0, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53]	3082	1	11	12	5.56%
25	[0, 48, 49, 50, 51, 53, 54, 55, 56, 57, 58]	3101	4	8	7	36.11%
26	[0, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52]	3116	1	11	11	8.33%
27	[0, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 54]	3175	1	11	13	2.78%
28	[0, 39, 43, 44, 45, 46, 47, 48, 49, 50, 52, 53, 54, 59]	3468	2	10	12	11.11%
29	[0, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 60]	3486	2	10	13	8.33%
30	[0, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52]	3542	1	11	11	8.33%
31	[0, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51]	3796	1	11	11	8.33%

Tabel 4.19 menunjukkan hasil perbandingan hasil estimasi *pitch* dan *pitch* pada

BAB 4 IMPLEMENTASI DAN PENGUJIAN

dataset dari pengujian lagu Old McDonald Had a Farm pada model pertama. Keterangan pada kolom di Tabel 4.19 sama dengan Tabel 4.16.

Tabel 4.19 Hasil Perbandingan *Pitch* pada lagu Old McDonald Had a Farm di Model Kedua

No	<i>Pitch result</i>	<i>Distance</i>	Match	Diff1	Diff2	Percentage
1	[0, 52, 56, 57, 58, 59, 62, 63, 64, 65, 66, 67]	133	11	1	1	91.67%
2	[0, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68]	688	12	0	2	94.44%
3	[0, 49, 56, 57, 58, 59, 61, 62, 63, 64, 65, 66]	744	10	2	2	83.33%
4	[0, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63]	813	8	4	4	66.67%
5	[0, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64]	871	9	3	2	77.78%
6	[0, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65]	926	10	2	1	86.11%
7	[0, 59, 60, 61, 62, 63, 64, 66, 67, 68, 69, 70]	1044	8	4	4	66.67%
8	[0, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64]	1100	9	3	6	66.67%
9	[0, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62]	1211	7	5	6	55.56%
10	[0, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59]	1579	5	7	6	44.44%
11	[0, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60]	1703	6	6	9	41.67%
12	[0, 48, 49, 50, 51, 52, 54, 55, 56, 57, 58, 59, 60]	1817	6	6	7	47.22%
13	[0, 48, 49, 50, 53, 54, 55, 56, 57, 58]	1953	4	8	6	38.89%
14	[0, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58]	2014	4	8	9	30.56%
15	[0, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58]	2299	4	8	9	30.56%

BAB 4 IMPLEMENTASI DAN PENGUJIAN

Tabel 4.19 Hasil Perbandingan *Pitch* pada lagu Old McDonald Had a Farm di Model Kedua

No	Pitch result	Distance	Match	Diff1	Diff2	Percentage
16	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56]	2361	2	10	10	16.67%
17	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56]	2593	2	10	10	16.67%
18	[0, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53]	2662	1	11	12	5.56%
19	[0, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55]	2687	1	11	12	5.56%
20	[0, 42, 43, 44, 45, 47, 48, 49, 50, 51, 52]	2785	1	11	10	11.11%
21	[0, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 54]	2816	1	11	12	5.56%
22	[0, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53]	2821	1	11	11	8.33%
23	[0, 48, 49, 50, 51, 53, 54, 55, 56, 57, 58]	2888	4	8	7	36.11%
24	[0, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53]	2928	1	11	10	11.11%
25	[0, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54]	2983	1	11	14	0.00%
26	[0, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52]	3036	1	11	12	5.56%
27	[0, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52]	3146	1	11	12	5.56%
28	[0, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52]	3321	1	11	11	8.33%
29	[0, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54]	3363	1	11	13	2.78%
30	[0, 43, 44, 45, 46, 47, 48, 49, 51, 52, 53, 54]	3403	1	11	11	8.33%
31	[0, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51]	4093	1	11	10	11.11%

Dari keempat contoh hasil perbandingan tersebut, dapat dilihat bahwa dengan rentang *threshold* 1000 hingga 1500, persentase estimasi cukup tinggi. Hal ini menyatakan bahwa rekaman pada data uji tersebut mirip dengan *dataset*. Sementara itu, ketika rentang *threshold* dinaikkan, misalnya hingga 2000, nilai ini tidak lagi dapat melakukan seleksi rekaman dengan baik karena persentase estimasi untuk nilai *distance* yang >1500 mulai menunjukkan hasil yang tidak baik. Maka dari itu, rentang nilai *threshold* dipilih antara 1000-1500.

4.6 Analisis Kesalahan

Ada beberapa kesalahan yang terjadi selama proses pengujian sistem estimasi *pitch* dan pengujian kemiripan. Beberapa kesalahan yang akan dibahas adalah sebagai berikut.

1. Berdasarkan hasil pengujian kemiripan (bagian 4.5.6 dan 4.5.7), ditemukan bahwa data rekaman lagu dinyanyikan dalam oktaf yang pada umumnya berbeda-beda. Hal ini dapat menyebabkan kesalahan sistem dalam mengestimasi *pitch* dari sebuah lagu karena adanya ketidakseragaman *pitch* untuk satu lagu yang sama. Sistem harus mempelajari banyak *range* oktaf untuk satu lagu yang sama, bahkan beberapa *pitch class* yang berbeda karena menggunakan tangga nada yang berbeda. Hal ini menimbulkan turunnya akurasi sistem. Padahal, jika dilihat dari sudut pandang pembawaan karya musik, *pitch class* yang sama dalam oktaf yang berbeda dapat dianggap mirip. Misalnya, *pitch class* G3, G4, G5 dapat dianggap sama karena sama-sama merupakan *pitch class* G, hanya oktafnya saja yang berbeda. Selain itu, perbedaan tangga nada juga berpengaruh. Misalnya, seseorang bernyanyi dengan nada do, re, mi pada tangga nada C. *Pitch* do, re, mi pada tangga nada C dilambangkan dengan *pitch class* C, D, dan E. Pada rekaman yang dijadikan *template*, rekaman dinyanyikan dalam nada yang sama (do, re, mi), namun dalam tangga nada yang berbeda, misalnya tangga nada G, di mana nada do, re, mi dilambangkan dengan *pitch class* G, A, B.
2. Metode *dynamic time warping* yang diujikan pada bagian 4.5.6 dan 4.5.7 belum bisa menghubungkan nada yang sama dalam oktaf yang berbeda dalam proses pengujian kemiripan. Nada tersebut seharusnya dapat dikategorikan sebagai sesuatu yang mirip. Penjelasan persamaan nada antara oktaf yang berbeda dapat dilihat pada bagian ???. Hal ini dikarenakan *pitch label* yang digunakan adalah dalam bentuk nomor not MIDI, di mana metode *dynamic time warping* tidak dapat mengelompokkan nada-nada yang sama meskipun dalam oktaf yang berbeda. Karena adanya perbedaan oktaf ini, seseorang yang bernyanyi pada

oktaf ketiga dapat memiliki *distance* yang jauh bila dibandingkan dengan *template* rekaman di *dataset* yang bernyanyi pada nada yang sama di oktaf keempat.

3. Berdasarkan hasil pengujian estimasi *pitch* (bagian 4.5.1 sampai bagian 4.5.4) yang menggunakan spektrogram sebagai masukan, ditemukan bahwa sebaran energi pada rekaman lagu sangat berpengaruh dalam pembuatan spektrogram. Jika penyanyi memiliki energi yang tinggi dan *pitch* yang baik, maka spektrogram yang dihasilkan memiliki kualitas yang baik. Jika penyanyi yang sama tidak bernyanyi dengan jelas, sebaran energi akan kecil dan citra spektrogram yang dihasilkan pun menjadi gelap. Hal ini dapat membuat sistem mengalami kesalahan melakukan estimasi *pitch* karena citra spektrogram yang gelap meskipun sebenarnya merepresentasikan *pitch* yang sama, oleh sistem akan dianggap sesuatu yang berbeda. Gambar 4.76 dan 4.77 menunjukkan dua contoh pengaruh energi penyanyi pada rekaman dalam spektogram. Gambar 4.76 mengindikasikan bahwa penyanyi bernyanyi dengan energi yang tinggi, sebaliknya pada Gambar 4.77 penyanyi bernyanyi dengan energi yang rendah.



Gambar 4.76 Contoh spektrogram dengan sebaran energi tinggi

BAB 4 IMPLEMENTASI DAN PENGUJIAN



Gambar 4.77 Contoh spektogram dengan sebaran energi rendah

BAB 5 KESIMPULAN DAN SARAN

Bab ini berisi kesimpulan yang dilandasi oleh penelitian dan pengujian yang telah dilakukan. Selain itu, bab ini juga dilengkapi dengan saran yang dapat digunakan atau dipertimbangkan sebagai konsiderasi untuk penelitian di masa depan.

5.1 Kesimpulan

Kesimpulan dari pembuatan sistem estimasi *pitch* dengan menggunakan metode *convolutional neural network* dan pengujian kemiripan dengan menggunakan metode *dynamic time warping* melalui pengujian yang telah dilakukan berdasarkan tujuan penelitian adalah sebagai berikut.

1. Metode *convolutional neural network* dapat melakukan estimasi *pitch* pada objek suara penyanyi dengan akurasi rata-rata tertinggi sebesar 97.425% pada arsitektur pertama (empat kali konvolusi dan jumlah *filter* 4608 yang dibagi menjadi 1024, 512, 1024, 2048 untuk masing-masing konvolusi), *epoch* = 300, dan *learning rate* = 0.0001 (10^{-4}) dengan selisih antara rata-rata akurasi dan rata-rata akurasi validasi sebesar 14.383%. Meskipun pada hasil ini *overfitting* masih terjadi (ada selisih antara rata-rata akurasi dan rata-rata akurasi validasi), namun dalam proses pengujian, model ini memberikan hasil pengujian terbaik dibandingkan dengan model yang memiliki selisih antara rata-rata akurasi dan rata-rata akurasi validasi yang lebih kecil. Model ini juga memiliki rata-rata akurasi tertinggi dari seluruh hasil pengujian yang dilakukan, sehingga model arsitektur pertama dengan kombinasi parameter yang sudah dituliskan di atas digunakan sebagai model utama dalam penelitian ini.
2. Arsitektur *convolutional neural network* yang berbeda-beda dapat memengaruhi hasil akurasi karena adanya perbedaan jumlah konvolusi untuk melakukan ekstraksi fitur. Semakin banyak dilakukan konvolusi dengan jumlah *filter* yang optimal pada masing-masing *convolutional layer*, arsitektur yang dibangun dapat dinyatakan semakin baik untuk melakukan estimasi *pitch*. Contohnya, pada arsitektur pertama dan kedua, rata-rata akurasi untuk *epoch* = 300 dan *learning rate* = 0.0001 (10^{-4}) memiliki perbedaan, di mana pada arsitektur pertama dihasilkan rata-rata akurasi sebesar 97.425% dan pada arsitektur kedua dihasilkan rata-rata akurasi 93.019%. Jumlah *filter* yang lebih banyak (arsitektur pertama memiliki jumlah *filter* = 4608; arsitektur kedua memiliki jumlah *filter* = 1920) akan membuat arsitektur lebih banyak belajar, namun harus dikombinasikan dengan nilai *epoch* dan *learning rate* yang tepat

agar mencapai nilai yang optimal dan tidak membuat model menjadi *overfitting* atau bahkan *underfitting*.

3. Nilai *threshold* yang optimal untuk menentukan kemiripan sebuah rekaman lagu dengan metode *dynamic time warping* berada dalam rentang 1000-1500. Jika *threshold* yang digunakan terlalu kecil, maka rekaman masukan cenderung harus identik dengan rekaman *template*, di mana kondisi ini sulit dicapai karena proses rekaman suara tidak mudah. Nilai *threshold* tersebut dipilih agar kesalahan-kesalahan yang tidak terlalu besar dalam rekaman dapat ditoleransi. Jika nilai *threshold* terlalu besar, rekaman yang memiliki banyak kesalahan akan dianggap mirip.

5.2 Saran

Saran untuk pengembangan sistem estimasi *pitch* pada penelitian selanjutnya antara lain sebagai berikut.

1. Menggunakan *dataset* dengan rekaman audio yang lebih seragam, baik secara *pitch* maupun *tempo* (kecepatan pembawaan suatu karya musik). Rekaman audio yang lebih seragam diharapkan dapat mengurangi kompleksitas arsitektur *convolutional neural network* yang dibangun. *Dataset* dari MIR-QBSH yang digunakan dalam penelitian ini memiliki banyak keragaman *pitch* dan *tempo* dalam satu lagu yang sama. Keragaman *pitch* yang dimaksud adalah keragaman penggunaan tangga nada serta oktaf yang digunakan saat menyanyikan lagu. Meskipun jika dipandang dari sudut pandang pembawaan musik hal ini tidak terlalu memiliki pengaruh signifikan selama *pitch* yang dibawakan tepat, namun hal ini membuat *convolutional neural network* harus banyak mempelajari data yang berbeda-beda dalam satu lagu yang sama karena *pitch label* didasarkan pada nomor not MIDI. Walaupun seorang penyanyi bernyanyi pada *pitch* yang sama namun dalam oktaf berbeda, *convolutional neural network* yang digunakan dalam penelitian ini menganggap kedua *pitch* tersebut berbeda karena memiliki nomor not MIDI berbeda dan citra spektrogram yang juga berbeda. Kecepatan bernyanyi yang berbeda-beda juga menyulitkan estimasi *pitch* karena potongan-potongan lagu menjadi tidak seragam, ada yang pendek dan ada yang panjang.
2. Menggunakan *dataset* yang tidak memiliki terlalu banyak *pitch label* dengan nilai 0. Jika *dataset* memiliki banyak *pitch label* dengan nilai 0, dapat dikembangkan sebuah algoritme untuk *voice and non-voice detection* untuk mengolah fragmen audio dengan *pitch label* 0 sebelum melakukan estimasi *pitch*.

3. Jika *dataset* lain yang digunakan juga tidak dapat memberikan data rekaman dengan tangga nada yang sama, maka pada proses *preprocessing*, dapat ditambahkan proses *musical key estimation* [?, ?] untuk mengetahui tangga nada yang digunakan dalam rekaman sebelum dimasukkan ke proses estimasi *pitch*. Jika *dataset* lain yang digunakan memberikan data rekaman dengan tangga nada yang sama namun oktaf yang berbeda, pada *preprocessing* dapat ditambahkan mekanisme untuk mengelompokkan *pitch class* yang sama dalam oktaf berbeda.
4. Mencari literatur terkait pembuatan arsitektur *convolutional neural network* yang dapat melakukan *generalization* dengan lebih baik. Dapat juga dipertimbangkan untuk menggabungkan/mengganti metode *convolutional neural network* dengan metode lain, misalnya dengan *recurrent neural network*, *long-short-term memory*, atau *convolutional-recurrent neural network*.

DAFTAR REFERENSI

- [1] Bendjilali Ridha Ilyas, Beladgham Mohammed, Merit Khaled and Taleb-Ahmed Abdelmalik, "*Illumination-robust face recognition based on deep convolutional neural networks architecture*", *Indonesian Journal of Electrical Engineering and Computer Science*, May 2020.
- [2] Samar S. Mohammed, Wael A. Mohamed, A.T. Khalil and A.S. Mora, "*Deep Learning Face Detection and Recognition*", *Iternational journal of electronics and telecommunications*, June 2019.
- [3] KB. Pranav, J. Manikandan, "*Design and Evaluation of a Real-Time Face Recognition System using Convolutional Neural Networks*", Article in *Procedia Computer Science 171 (2020) 1651–1659*,2020.
- [4] Coskun Musab,Ucar Aysegul, Yildirm Ozal and Demir Yakup, "*Face Recognition Based on Convolutional Neural Network*", *Conference Paper*, November, 2021.
- [5] Zhiming Xie, Junjie Li and Hui Shi, "*A Face Recognition Method Based on CNN*", *2019 High Performance Computing and Computational Intelligence Conference*,2019
- [6] C. Beumier, "Face Recognition," *Natl. Sci. Technol. Counc.*, pp. 92–99, 2001.
- [7] Bhardwaj, Anjali. *What is a Perceptron? – Basics of Neural Networks.* <https://towardsdatascience.com/what-is-a-perceptron-basics-of-neural-networks-c4cfea20c590> (accessed September 13,2021 12:30:34)
- [8] Donges, Niklas. *Gradient Descent: An Introduction to 1 of Machine Learning's Most Popular Algorithms.* <https://builtin.com/data-science/gradient-descent> (accessed September 14, 2021 11:40:23)
- [9] Pandey, Paul. *Understanding the Mathematics behind Gradient Descent.* <https://towardsdatascience.com/understanding-the-mathematics-behind-gradient-descent-dde5dc9be06e> (accessed September 14, 2021 11:46:30)
- [10] Shah, Saily. *Cost Function is No Rocket Science!* <https://www.analyticsvidhya.com/blog/2021/02/cost-function-is-no-rocket-science/> (accessed September 14, 2021 12:45:12)

DAFTAR REFERENSI

- [11] B.Dickson. "What are convolutional neural networks (CNN)?".bdtechtalk.com. <https://bdtechtalks.com/2020/01/06/convolutional-neural-networks-cnn-convnets/> (accessed May 4, 2021 14:55:23)
- [12] IBM Cloud Education. *Convolutional Neural Networks* <https://www.ibm.com/cloud/learn/convolutional-neural-networks> (accessed October 25, 2021 13:30:30)
- [13] IBM Cloud Education. *Neural Networks*. <https://www.ibm.com/cloud/learn/neural-networks> (accessed September 13, 2021 17:17:23)
- bib
- [14] Jeong, Jiwon. *The Most Intuitive and Easiest Guide for Convolutional Neural Network* <https://towardsdatascience.com/the-most-intuitive-and-easiest-guide-for-convolutional-neural-network-3607be47480> (accessed October 25, 2021 13:36 54)
- [15] Saxena, Shipra. *Introduction to Batch Normalization* <https://www.analyticsvidhya.com/blog/2021/03/introduction-to-batch-normalization/> (accessed October 25, 2021 17:43:42)
- [16] Mustafa. *Optimizers in Deep Learning* <https://medium.com/mlearning-ai/optimizers-in-deep-learning-7bf81fed78a0> (accessed October 31, 2021 19:49:50)
- [17] Simonyan. Karen and Zisserman. Andrew, "VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE SCALE IMAGE RECOGNITION" conference paper at ICLR 2015, April 10 2015
- [18] Sharma, Pulkit. "A Comprehensive Tutorial to learn Convolutional Neural Networks from Scratch (deeplearning.ai Course #4)". <https://www.analyticsvidhya.com/blog/2018/12/guide-convolutional-neural-network-cnn/> (accessed October 25, 2021 23:54:50)
- [19] Kumar, Ahlad. "Lecture: CNN Architectures (AlexNet, VGGNet, Inception ResNet)". <https://www.youtube.com/watch?v=CN NzI8HIIU> (accessed October 26, 2021 13:24:20)
- [20] Rosebrock, Adrian. 2017. *Deep Learning for Computer Vision with Python*. PyImageSearch

DAFTAR REFERENSI

- [21] Rosebrock, Adrian.2017.*Deep Learning for Computer Vision with Python*. pp. 20 - 22.
- [22] Ruder, Sebastian.2016.*An overview of gradient descent optimization algorithms*.<https://www.notion.so/Gradient-with-momentum-b1b93b72ff71402ebd6c47b49a408f81> (accessed November 1, 2021 (16:28:00))
- [23] Kumawat, Neha. *Continuing on Adaptive Method: ADADELTA and RMSProp*. <https://insideaiml.com/blog/Adagrad-and-Adadelta-optimizer:-In-depth-explanation-1052> (accessed January 9,2022 (21:42:00))
- [24] Maiza, Ashref. *The Unknown Benefits of using a Soft-F1 Loss in Classification Systems*. <https://towardsdatascience.com/the-unknown-benefits-of-using-a-soft-f1-loss-in-classification-systems-753902c0105d> (accessed November 7 2021, 00:24:00)
- [25] Ilyas, Ridwan. *Penjelasan Algoritma Backpropagation Mengupdate Bobot | Jurnal Kelas*. <https://www.youtube.com/watch?v=OeowRowEbBc> (accessed November 7, 2021 11:41:00)
- [26] Gupta, Dishashree. *Fundamentals of Deep Learning – Activation Functions and When to Use Them?*.<https://www.analyticsvidhya.com/blog/2020/01/fundamentals-deep-learning-activation-functions-when-to-use-them/> (accessed November 7, 2021 11:45:00)
- [27] Naik, Krish. *Deep Learning-All Optimizers In One Video-SGD with Momentum,Adagrad,Adadelta,RMSprop,Adam Optimizers*. <https://www.youtube.com/watch?v=TudQZtgpoHk> (accessed November 7, 2021 13:08:00)
- [28] opencv. *Color conversions*. https://docs.opencv.org/3.4.15/de/d25/imgproc_colorconversions.html 26 : 00)
- [29] I. Goodfellow, Y. Bengio, and A. Courville,*Deep Learning, An MIT Press book*,2016. [Online]. Available : <http://www.deeplearningbook.org> (accessed November 7,2021 16:57:00)
- [30] S. Khan, H. Rahmani, Syed Afaq Ali Shah, M. Bennamoun, "A Guide to Convolutional Neural Networks for Computer

DAFTAR REFERENSI

- Vision", 1sted, GrardMedioniandSvenDickinson, Ed.California : MorganandClaypool, 2018, pp.45, 53, 56, 67 – 80.*
- [31] Kapoor, Namrata. *Weight Initialization Techniques-What best works for you.*<https://www.numpyninja.com/post/weight-initialization-techniques> (accessed November 7,2021 20:15:00)
- [32] Taunk, Dhaval. *L1 vs L2 Regularization: The intuitive difference.*<https://medium.com/analytics-vidhya/l1-vs-l2-regularization-which-is-better-d01068e6658c> (accessed November 7, 2021 21:40:00)
- [33] H. Habibi Aghdam and E. Jahani Heravi, *Guide to Convolutional Neural Networks*, 1 st ed. Switzerland: Springer International Publishing AG, 2017, pp. 108-111, 118-120.
- [34] Lina, Qobylatul. *Implementasi Deep Learning Menggunakan Convolutional Neural Network untuk Klasifikasi Gambar (Mata Juling dan Mata Normal) dengan R.* <https://medium.com/@16611110/implementasi-deep-learning-menggunakan-convolutional-neural-network-untuk-klasifikasi-gambar-mata-87dcc0ad26e0> (accessed November 16, 2021 22:39:00)
- [35] H. Kinsley and D. Kukieła, “*Neural Networks from Scratch in Python*”, 1 st ed. Harrison Kinsley, 2020, pp. 108, 333-358.
- [36] Bhandari Aniruddha, *Image Augmentation on the fly using Keras ImageDataGenerator.* <https://www.analyticsvidhya.com/blog/2020/08/image-augmentation-on-the-fly-using-keras-imagedatagenerator/> (accessed November 24, 2021 18:23:00)
- [37] Sudhakar, Shreenidhi. *Histogram Equalization.* <https://towardsdatascience.com/histogram-equalization-5d1013626e64> (accessed November 25,2021 12:16:00)
- [38] Frickle, Tobin. *Rotation by Shearing.* <https://www.ocf.berkeley.edu/~fricke/projects/israel/paeth/> 15 : 00)
- [39] R.C. Gonzalez and R.E. Woods, *Digital Image Processing*, 3 rd ed. New Jersey: Pearson Prentice Hall, 2008, pp.91, pp.122-126

LAMPIRAN A

NOMOR NOT MIDI, NAMA NOT, DAN *CENTER FREQUENCY*

Tabel A merangkum nomor not MIDI seperti yang ditentukan dalam standar MIDI dan dicocokkan dengan *Middle C* (nomor not 60) sebagai C4. Persamaan yang menghubungkan nomor not MIDI dan frekuensi dasar dengan asumsi *equal tuning* berdasarkan A4 = 440 Hz, dapat dilihat pada Persamaan ??.

Tabel A Nomor not MIDI, nama not, dan *center frequency*

Nomor Not MIDI	Nama Not	<i>Center frequency</i>
127	G9	12,543.85
126	F♯9 / G♭9	11,839.82
125	F9	11,175.30
124	E9	10,548.08
123	D♯9 / E♭9	9,956.06
122	D9	9,397.27
121	C♯9 / D♭9	8,869.84
120	C9	8,372.02
119	B8	7,902.13
118	A♯8 / B♭8	7,458.62
117	A8	7,040
116	G♯8 / A♭8	6,644.88
115	G8	6,271.93
114	F♯8 / G♭8	5,919.91
113	F8	5,587.65
112	E8	5,274.04
111	D♯8 / E♭8	4,978.03
110	D8	4,698.64
109	C♯8 / D♭8	4,434.92
108	C8	4,186.01
107	B7	3,951.07
106	A♯7 / B♭7	3,729.31
105	A7	3,520
104	G♯7 / A♭7	3,322.44

LAMPIRAN A
Nomor Not MIDI, Nama Not, dan *Center Frequency*

Tabel A Nomor not MIDI, nama not, dan *center frequency*

Nomor Not MIDI	Nama Not	<i>Center frequency</i>
103	G7	3,135.96
102	F♯7 / G♭7	2,959.96
101	F7	2,793.83
100	E7	2,637.02
99	D♯7 / E♭7	2,489.02
98	D7	2,349.32
97	C♯7 / D♭7	2,217.46
96	C7	2,093
95	B6	1,975.53
94	A♯6 / B♭6	1,864.66
93	A6	1,760
92	G♯6 / A♭6	1,661.22
91	G6	1,567.98
90	F♯6 / G♭6	1,479.98
89	F6	1,396.91
88	E6	1,318.51
87	D♯6 / E♭6	1,244.51
86	D6	1,174.66
85	C♯6 / D♭6	1,108.73
84	C6	1046.5
83	B5	987.77
82	A♯5 / B♭5	932.33
81	A5	880
80	G♯5 / A♭5	830.61
79	G5	783.99
78	F♯5 / G♭5	739.99
77	F5	698.46
76	E5	659.26
75	D♯5 / E♭5	622.25
74	D5	587.33
73	C♯5 / D♭5	554.37
72	C5	523.25

LAMPIRAN A
Nomor Not MIDI, Nama Not, dan *Center Frequency*

Tabel A Nomor not MIDI, nama not, dan *center frequency*

Nomor Not MIDI	Nama Not	Center frequency
71	B4	493.88
70	A♯4 / B♭4	466.16
69	A4 (<i>concert pitch</i>)	440
68	G♯4 / A♭4	415.3
67	G4	392
66	F♯4 / G♭4	369.99
65	F4	349.23
64	E4	329.63
63	D♯4 / E♭4	311.13
62	D4	293.66
61	C♯4 / D♭4	277.18
60	C4 (<i>middle C</i>)	261.63
59	B3	246.94
58	A♯3 / B♭3	233.08
57	A3	220
56	G♯3 / A♭3	207.65
55	G3	196
54	F♯3 / G♭3	185
53	F3	174.61
52	E3	164.81
51	D♯3 / E♭3	155.56
50	D3	146.83
49	C♯3 / D♭3	138.59
48	C3	130.81
47	B2	123.47
46	A♯2 / B♭2	116.54
45	A2	110
44	G♯2 / A♭2	103.83
43	G2	98
42	F♯2 / G♭2	92.50
41	F2	87.31
40	E2	82.41

LAMPIRAN A
Nomor Not MIDI, Nama Not, dan *Center Frequency*

Tabel A Nomor not MIDI, nama not, dan *center frequency*

Nomor Not MIDI	Nama Not	Center frequency
39	D♯2 / E♭2	77.78
38	D2	73.42
37	C♯2 / D♭2	69.30
36	C2	65.41
35	B1	61.74
34	A♯1 / B♭1	58.27
33	A1	55
32	G♯1 / A♭1	51.91
31	G1	49
30	F♯1 / G♭1	46.25
29	F1	43.65
28	E1	41.20
27	D♯1 / E♭1	38.89
26	D1	36.71
25	C♯1 / D♭1	34.65
24	C1	32.70
23	B0	30.87
22	A♯0/B♭0	29.14
21	A0	27.50
20	G♯0/ A♭0	25.96
19	G0	24.50
18	F♯0 / G♭0	23.12
17	F0	21.83
16	E0	20.60
15	D♯0 / E♭0	19.45
14	D0	18.35
13	C♯0 / D♭0	17.32
12	C0	16.35
11	-	15.43
10	-	14.57
9	-	13.75
8	-	12.98

LAMPIRAN A
Nomor Not MIDI, Nama Not, dan *Center Frequency*

Tabel A Nomor not MIDI, nama not, dan *center frequency*

Nomor Not MIDI	Nama Not	<i>Center frequency</i>
7	-	12.25
6	-	11.56
5	-	10.91
4	-	10.30
3	-	9.72
2	-	9.18
1	-	8.66
0	-	8.18

LAMPIRAN B

HASIL PENGUJIAN ARSITEKTUR CONVOLUTIONAL NEURAL NETWORK

Bagian ini menunjukkan hasil pengujian arsitektur *convolutional neural network* untuk masing-masing kombinasi *epoch*, *learning rate*, jumlah *filter*, dan jumlah konvolusi. Ada empat buah arsitektur yang diuji. Pada setiap arsitektur pertama hingga keempat, diujikan kombinasi *item* pengujian yang sudah dituliskan di atas. Total kombinasi yang diujikan pada setiap arsitektur adalah 96 kombinasi yang rangkuman hasil pengujianya sudah dituliskan pada bagian 4.5 . Hasil pengujian yang menunjukkan perubahan per *epoch* pada setiap arsitektur disimpan ke dalam berkas .xlsx. Berkas tersebut dapat diakses secara *online* pada pranala berikut dengan terlebih dahulu memohon *access request* kepada penulis. Pranala yang dapat diakses adalah <http://bit.ly/CNN-Test-Results-1117002>.

LAMPIRAN C

HASIL PENGUJIAN *DISTANCE DENGAN DYNAMIC TIME WARPING DAN EQUIVALENT PITCH CLASS*

Bagian ini menunjukkan hasil pengujian *distance* untuk *dynamic time warping* (DTW). Pengujian dilakukan kepada lima buah lagu, dengan masing-masing jumlah data yang dapat dilihat pada Tabel 4.3. Pengujian dilakukan kepada dua model, seperti yang sudah dijelaskan pada bagian 4.5.6. Setelah dilakukan pengujian *distance*, baru diambil kesimpulan berapa *threshold* yang digunakan untuk melakukan pengujian kemiripan.

Tabel C-1 menunjukkan hasil pengujian *distance* dengan DTW di lagu Twinkle-Twinkle Little Star pada model pertama.

Tabel C-1 Hasil pengujian *distance* pada lagu Twinkle-Twinkle Little Star di model pertama

No	Unique pitch	Equivalent pitch class	Distance
1	0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58	A \sharp 2 / B \flat 2, B2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3, F3, F \sharp 3 / G \flat 3, G3, G \sharp 3 / A \flat 3, A3, A \sharp 3 / B \flat 3	2
2	0, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57	G \sharp 2 / A \flat 2, A2, A \sharp 2 / B \flat 2, B2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3, F3, F \sharp 3 / G \flat 3, G3, G \sharp 3 / A \flat 3, A3	521
3	0, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59	C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3, F3, F \sharp 3 / G \flat 3, G3, G \sharp 3 / A \flat 3, A3, A \sharp 3 / B \flat 3, B3	550
4	0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55	A \sharp 2 / B \flat 2, B2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3, F3, F \sharp 3 / G \flat 3, G3	637
5	0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56	A \sharp 2 / B \flat 2, B2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3, F3, F \sharp 3 / G \flat 3, G3, G \sharp 3 / A \flat 3	646
6	0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56	A \sharp 2 / B \flat 2, B2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3, F3, F \sharp 3 / G \flat 3, G3, G \sharp 3 / A \flat 3	739
7	0, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54	G \sharp 2 / A \flat 2, A2, A \sharp 2 / B \flat 2, B2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3, F3, F \sharp 3 / G \flat 3	860
8	0, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55	A2, A \sharp 2 / B \flat 2, B2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3, F3, F \sharp 3 / G \flat 3, G3	999
9	0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57	A \sharp 2 / B \flat 2, B2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3, F3, F \sharp 3 / G \flat 3, G3, G \sharp 3 / A \flat 3, A3	1055
10	0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56	A \sharp 2 / B \flat 2, B2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3, F3, F \sharp 3 / G \flat 3, G3, G \sharp 3 / A \flat 3	1150
11	0, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55	A2, A \sharp 2 / B \flat 2, B2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3, F3, F \sharp 3 / G \flat 3, G3	1222
12	0, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55	G2, G \sharp 2 / A \flat 2, A2, A \sharp 2 / B \flat 2, B2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3, F3, F \sharp 3 / G \flat 3, G3	1250
13	0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57	A \sharp 2 / B \flat 2, B2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3, F3, F \sharp 3 / G \flat 3, G3, G \sharp 3 / A \flat 3, A3	1298

LAMPIRAN C

Hasil Pengujian *Distance* dengan *Dynamic Time Warping* dan *Equivalent Pitch Class*

Tabel C-1 Hasil pengujian *distance* pada lagu Twinkle-Twinkle Little Star di model pertama

No	Unique pitch	Equivalent pitch class	Distance
14	0, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52	F♯2 / G♭2, G2, G♯2 / A♭2, A2, A♯2 / B♭2, B2, C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3	1299
15	0, 44, 45, 46, 48, 49, 50, 51, 52, 53, 54, 55, 56	G♯2 / A♭2, A2, A♯2 / B♭2, C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F3, F♯3 / G♭3, G3, G♯3 / A♭3	1370
16	0, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55	G♯2 / A♭2, A2, A♯2 / B♭2, B2, C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F3, F♯3 / G♭3, G3	1450
17	0, 45, 46, 47, 48, 49, 50, 51, 52, 54, 55, 56, 57, 59, 60	A2, A♯2 / B♭2, B2, C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F♯3 / G♭3, G3, G♯3 / A♭3, A3, B3, C4	1557
18	0, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 53	E2, F2, F♯2 / G♭2, G2, G♯2 / A♭2, A2, A♯2 / B♭2, B2, C3, C♯3 / D♭3, D3, D♯3 / E♭3, F3	1680
19	0, 43, 44, 45, 46, 47, 48, 49, 50, 53, 54	G2, G♯2 / A♭2, A2, A♯2 / B♭2, B2, C3, C♯3 / D♭3, D3, F3, F♯3 / G♭3	1707
20	0, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 56	F2, F♯2 / G♭2, G2, G♯2 / A♭2, A2, A♯2 / B♭2, B2, C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F3, F♯3 / G♭3, G3, G♯3 / A♭3	1817
21	0, 48, 49, 50, 52, 53, 54, 55, 56, 57, 58, 59, 60	C3, C♯3 / D♭3, D3, E3, F3, F♯3 / G♭3, G3, G♯3 / A♭3, A3, A♯3 / B♭3, B3, C4	1819
22	0, 47, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63	B2, F♯3 / G♭3, G3, G♯3 / A♭3, A3, A♯3 / B♭3, B3, C4, C♯4 / D♭4, D4, D♯4 / E♭4	1847
23	0, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54	G2, G♯2 / A♭2, A2, A♯2 / B♭2, B2, C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F3, F♯3 / G♭3	1931
24	0, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64	F♯3 / G♭3, G3, G♯3 / A♭3, A3, A♯3 / B♭3, B3, C4, C♯4 / D♭4, D4, D♯4 / E♭4, E4	2023
25	0, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55	G♯2 / A♭2, A2, A♯2 / B♭2, B2, C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F3, F♯3 / G♭3, G3	2054
26	0, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53	F♯2 / G♭2, G2, G♯2 / A♭2, A2, A♯2 / B♭2, B2, C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F3	2078
27	0, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62	E3, F3, F♯3 / G♭3, G3, G♯3 / A♭3, A3, A♯3 / B♭3, B3, C4, C♯4 / D♭4, D4	2217
28	0, 54, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66	F♯3 / G♭3, A3, A♯3 / B♭3, B3, C4, C♯4 / D♭4, D4, D♯4 / E♭4, E4, F4, F♯4 / G♭4	2343
29	0, 45, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66	A2, G♯3 / A♭3, A3, A♯3 / B♭3, B3, C4, C♯4 / D♭4, D4, D♯4 / E♭4, E4, F4, F♯4 / G♭4	2370
30	0, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68	A3, A♯3 / B♭3, B3, C4, C♯4 / D♭4, D4, D♯4 / E♭4, E4, F4, F♯4 / G♭4, G4, G♯4 / A♭4	2502
31	0, 45, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66	A2, G♯3 / A♭3, A3, A♯3 / B♭3, B3, C4, C♯4 / D♭4, D4, D♯4 / E♭4, E4, F4, F♯4 / G♭4	2578
32	0, 42, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64	F♯2 / G♭2, F♯3 / G♭3, G3, G♯3 / A♭3, A3, A♯3 / B♭3, B3, C4, C♯4 / D♭4, D4, D♯4 / E♭4, E4	2871

Tabel C-2 menunjukkan hasil pengujian *distance* dengan DTW di lagu Old McDonald pada model pertama.

LAMPIRAN C

Hasil Pengujian *Distance* dengan *Dynamic Time Warping* dan *Equivalent Pitch Class*

Tabel C-2 Hasil pengujian *distance* pada lagu Old McDonald di model pertama

No	Unique pitch	Equivalent pitch class	Distance
1	0, 52, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67	E3, G \sharp 3 / A \flat 3, A3, A \sharp 3 / B \flat 3, B3, C4, C \sharp 4 / D \flat 4, D4, D \sharp 4 / E \flat 4, E4, F4, F \sharp 4 / G \flat 4, G4	132
2	0, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68	G3, G \sharp 3 / A \flat 3, A3, A \sharp 3 / B \flat 3, B3, C4, C \sharp 4 / D \flat 4, D4, D \sharp 4 / E \flat 4, E4, F4, F \sharp 4 / G \flat 4, G4, G \sharp 4 / A \flat 4	645
3	0, 49, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64	C \sharp 3 / D \flat 3, G3, G \sharp 3 / A \flat 3, A3, A \sharp 3 / B \flat 3, B3, C4, C \sharp 4 / D \flat 4, D4, D \sharp 4 / E \flat 4, E4	756
4	0, 53, 54, 55, 56, 57, 59, 60, 61, 62, 63	F3, F \sharp 3 / G \flat 3, G3, G \sharp 3 / A \flat 3, A3, B3, C4, C \sharp 4 / D \flat 4, D4, D \sharp 4 / E \flat 4	817
5	0, 42, 49, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66	F \sharp 2 / G \flat 2, C \sharp 3 / D \flat 3, G \sharp 3 / A \flat 3, A3, A \sharp 3 / B \flat 3, B3, C4, C \sharp 4 / D \flat 4, D4, D \sharp 4 / E \flat 4, E4, F4, F \sharp 4 / G \flat 4	1068
6	0, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62	D \sharp 3 / E \flat 3, E3, F3, F \sharp 3 / G \flat 3, G3, G \sharp 3 / A \flat 3, A3, A \sharp 3 / B \flat 3, B3, C4, C \sharp 4 / D \flat 4, D4	1171
7	0, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70	C4, C \sharp 4 / D \flat 4, D4, D \sharp 4 / E \flat 4, E4, F4, F \sharp 4 / G \flat 4, G4, G \sharp 4 / A \flat 4, A4, A \sharp 4 / B \flat 4	1181
8	0, 45, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65	A2, G \sharp 3 / A \flat 3, A3, A \sharp 3 / B \flat 3, B3, C4, C \sharp 4 / D \flat 4, D4, D \sharp 4 / E \flat 4, E4, F4	1200
9	0, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64	D \sharp 3 / E \flat 3, E3, F3, F \sharp 3 / G \flat 3, G3, G \sharp 3 / A \flat 3, A3, A \sharp 3 / B \flat 3, B3, C4, C \sharp 4 / D \flat 4, D4, D \sharp 4 / E \flat 4, E4	1245
10	0, 48, 49, 50, 51, 52, 54, 55, 56, 57, 58, 59, 60	C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3, F \sharp 3 / G \flat 3, G3, G \sharp 3 / A \flat 3, A3, A \sharp 3 / B \flat 3, B3, C4	1648
11	0, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59	D3, D \sharp 3 / E \flat 3, E3, F3, F \sharp 3 / G \flat 3, G3, G \sharp 3 / A \flat 3, A3, A \sharp 3 / B \flat 3, B3	1668
12	0, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60	B2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3, F3, F \sharp 3 / G \flat 3, G3, G \sharp 3 / A \flat 3, A3, A \sharp 3 / B \flat 3, B3, C4	1742
13	0, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58	C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3, F3, F \sharp 3 / G \flat 3, G3, G \sharp 3 / A \flat 3, A3, A \sharp 3 / B \flat 3	1836
14	0, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58	B2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3, F3, F \sharp 3 / G \flat 3, G3, G \sharp 3 / A \flat 3, A3, A \sharp 3 / B \flat 3	2151
15	0, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57	A2, A \sharp 2 / B \flat 2, B2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3, F3, F \sharp 3 / G \flat 3, G3, G \sharp 3 / A \flat 3, A3	2173
16	0, 45, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58	A2, B2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3, F3, F \sharp 3 / G \flat 3, G3, G \sharp 3 / A \flat 3, A3, A \sharp 3 / B \flat 3	2461
17	0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56	A \sharp 2 / B \flat 2, B2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3, F3, F \sharp 3 / G \flat 3, G3, G \sharp 3 / A \flat 3	2466
18	0, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52	F \sharp 2 / G \flat 2, G2, G \sharp 2 / A \flat 2, A2, A \sharp 2 / B \flat 2, B2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3	2716
19	0, 44, 45, 46, 47, 49, 50, 51, 52, 53, 54, 55	G \sharp 2 / A \flat 2, A2, A \sharp 2 / B \flat 2, B2, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3, F3, F \sharp 3 / G \flat 3, G3	2736
20	0, 42, 43, 44, 45, 47, 48, 49, 50, 51, 52, 55	F \sharp 2 / G \flat 2, G2, G \sharp 2 / A \flat 2, A2, B2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3, G3	2808
21	0, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53	G \sharp 2 / A \flat 2, A2, A \sharp 2 / B \flat 2, B2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3, F3	2940
22	0, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54	F2, F \sharp 2 / G \flat 2, G2, G \sharp 2 / A \flat 2, A2, A \sharp 2 / B \flat 2, B2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3, F3, F \sharp 3 / G \flat 3	2990

LAMPIRAN C

Hasil Pengujian *Distance* dengan *Dynamic Time Warping* dan *Equivalent Pitch Class*

Tabel C-2 Hasil pengujian *distance* pada lagu Old McDonald di model pertama

No	Unique pitch	Equivalent pitch class	Distance
23	0, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53	F♯2 / G♭2, G2, G♯2 / A♭2, A2, A♯2 / B♭2, B2, C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F3	2999
24	0, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53	F♯2 / G♭2, G2, G♯2 / A♭2, A2, A♯2 / B♭2, B2, C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F3	3082
25	0, 48, 49, 50, 51, 53, 54, 55, 56, 57, 58	C3, C♯3 / D♭3, D3, D♯3 / E♭3, F3, F♯3 / G♭3, G3, G♯3 / A♭3, A3, A♯3 / B♭3	3101
26	0, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52	F♯2 / G♭2, G2, G♯2 / A♭2, A2, A♯2 / B♭2, B2, C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3	3116
27	0, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 54	F2, F♯2 / G♭2, G2, G♯2 / A♭2, A2, A♯2 / B♭2, B2, C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F♯3 / G♭3	3175
28	0, 39, 43, 44, 45, 46, 47, 48, 49, 50, 52, 53, 54, 59	D♯2 / E♭2, G2, G♯2 / A♭2, A2, A♯2 / B♭2, B2, C3, C♯3 / D♭3, D3, E3, F3, F♯3 / G♭3, B3	3468
29	0, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 60	F♯2 / G♭2, G2, G♯2 / A♭2, A2, A♯2 / B♭2, B2, C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F3, F♯3 / G♭3, C4	3486
30	0, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52	F♯2 / G♭2, G2, G♯2 / A♭2, A2, A♯2 / B♭2, B2, C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3	3542
31	0, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51	F2, F♯2 / G♭2, G2, G♯2 / A♭2, A2, A♯2 / B♭2, B2, C3, C♯3 / D♭3, D3, D♯3 / E♭3	3796

Tabel **C-3** menunjukkan hasil pengujian *distance* dengan DTW di lagu Happy Birthday pada model pertama.

Tabel C-3 Hasil pengujian *distance* pada lagu Happy Birthday di model pertama

No	Unique pitch	Equivalent pitch class	Distance
1	0, 56, 57, 58, 59, 60, 61, 63, 64, 65, 66	G♯3 / A♭3, A3, A♯3 / B♭3, B3, C4, C♯4 / D♭4, D♯4 / E♭4, E4, F4, F♯4 / G4	297
2	0, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67	A♯3 / B♭3, B3, C4, C♯4 / D♭4, D4, D♯4 / E♭4, E4, F4, F♯4 / G♭4, G4	339
3	0, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66	F3, F♯3 / G♭3, G3, G♯3 / A♭3, A3, A♯3 / B♭3, B3, C4, C♯4 / D♭4, D4, D♯4 / E♭4, E4, F4, F♯4 / G♭4	805
4	0, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67	A3, A♯3 / B♭3, B3, C4, C♯4 / D♭4, D4, D♯4 / E♭4, E4, F4, F♯4 / G♭4, G4	982
5	0, 54, 55, 56, 57, 59, 60, 61, 62, 64, 65, 67, 68	F♯3 / G♭3, G3, G♯3 / A♭3, A3, B3, C4, C♯4 / D♭4, D4, E4, F4, G4, G♯4 / A♭4	1270
6	0, 45, 50, 54, 55, 56, 57, 58, 59, 60, 61, 62, 64, 66, 67	A2, D3, F♯3 / G♭3, G3, G♯3 / A♭3, A3, A♯3 / B♭3, B3, C4, C♯4 / D♭4, D4, E4, F♯4 / G♭4, G4	1366
7	0, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67	A3, A♯3 / B♭3, B3, C4, C♯4 / D♭4, D4, D♯4 / E♭4, E4, F4, F♯4 / G♭4, G4	1407
8	0, 58, 59, 60, 61, 62, 64, 65, 67, 68	A♯3 / B♭3, B3, C4, C♯4 / D♭4, D4, E4, F4, G4, G♯4 / A♭4	1450
9	0, 51, 53, 54, 55, 56, 57, 59, 60	D♯3 / E♭3, F3, F♯3 / G♭3, G3, G♯3 / A♭3, A3, B3, C4	1524
10	0, 51, 52, 53, 54, 55, 56, 57, 58, 59, 61	D♯3 / E♭3, E3, F3, F♯3 / G♭3, G3, G♯3 / A♭3, A3, A♯3 / B♭3, B3, C♯4 / D♭4	1780

LAMPIRAN C

Hasil Pengujian *Distance* dengan *Dynamic Time Warping* dan *Equivalent Pitch Class*

Tabel C-3 Hasil pengujian *distance* pada lagu Happy Birthday di model pertama

No	Unique pitch	Equivalent pitch class	Distance
11	0, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57	C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F3, F♯3 / G♭3, G3, G♯3 / A♭3, A3	2275
12	0, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60	A2, A♯2 / B♭2, B2, C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F3, F♯3 / G♭3, G3, G♯3 / A♭3, A3, A♯3 / B♭3, B3, C4	2435
13	0, 45, 46, 47, 48, 49, 51, 52, 53, 54, 56, 57, 58, 59	A2, A♯2 / B♭2, B2, C3, C♯3 / D♭3, D♯3 / E♭3, E3, F3, F♯3 / G♭3, G♯3 / A♭3, A3, A♯3 / B♭3, B3	2473
14	0, 42, 48, 49, 50, 51, 52, 54, 55, 56, 61	F♯2 / G♭2, C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F♯3 / G♭3, G3, G♯3 / A♭3, C♯4 / D♭4	2479
15	0, 45, 46, 47, 48, 50, 51, 52, 53, 54	A2, A♯2 / B♭2, B2, C3, D3, D♯3 / E♭3, E3, F3, F♯3 / G♭3	2641
16	0, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58	A2, A♯2 / B♭2, B2, C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F3, F♯3 / G♭3, G3, G♯3 / A♭3, A3, A♯3 / B♭3	2662
17	0, 46, 47, 48, 49, 50, 51, 52, 53, 54	A♯2 / B♭2, B2, C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F3, F♯3 / G♭3	2722
18	0, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 56, 57, 58	G♯2 / A♭2, A2, A♯2 / B♭2, B2, C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F3, F♯3 / G♭3, G♯3 / A♭3, A3, A♯3 / B♭3	2841
19	0, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58	G2, G♯2 / A♭2, A2, A♯2 / B♭2, B2, C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F3, F♯3 / G♭3, G3, G♯3 / A♭3, A3, A♯3 / B♭3	2847
20	0, 46, 47, 48, 49, 50, 52, 53, 54, 55	A♯2 / B♭2, B2, C3, C♯3 / D♭3, D3, E3, F3, F♯3 / G♭3, G3	2883
21	0, 44, 45, 46, 47, 48, 50, 51, 52, 53, 54, 55	G♯2 / A♭2, A2, A♯2 / B♭2, B2, C3, D3, D♯3 / E♭3, E3, F3, F♯3 / G♭3, G3	2903
22	0, 45, 46, 47, 48, 49, 50, 52, 53, 54, 55	A2, A♯2 / B♭2, B2, C3, C♯3 / D♭3, D3, E3, F3, F♯3 / G♭3, G3	2931
23	0, 44, 45, 46, 47, 48, 49, 50, 51, 52, 54, 55, 56, 57, 58	G♯2 / A♭2, A2, A♯2 / B♭2, B2, C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F3, F♯3 / G♭3, G3, G♯3 / A♭3, A3, A♯3 / B♭3	2967
24	0, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55	G♯2 / A♭2, A2, A♯2 / B♭2, B2, C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F3, F♯3 / G♭3, G3	2970
25	0, 44, 45, 46, 47, 49, 50, 51, 52, 53	G♯2 / A♭2, A2, A♯2 / B♭2, B2, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F3	3032
26	0, 44, 45, 46, 47, 49, 50, 51, 52, 54, 55, 56, 57, 58	G♯2 / A♭2, A2, A♯2 / B♭2, B2, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F3, F♯3 / G♭3, G3, G♯3 / A♭3, A3, A♯3 / B♭3	3064
27	0, 45, 46, 47, 48, 49, 51, 52, 53, 54, 55, 56, 58, 59	A2, A♯2 / B♭2, B2, C3, C♯3 / D♭3, D♯3 / E♭3, E3, F3, F♯3 / G♭3, G3, G♯3 / A♭3, A♯3 / B♭3, B3	3235
28	0, 45, 46, 47, 48, 49, 50, 51, 52, 53	A2, A♯2 / B♭2, B2, C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F3	3283
29	0, 42, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56	F♯2 / G♭2, G♯2 / A♭2, A2, A♯2 / B♭2, B2, C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F3, F♯3 / G♭3, G3, G♯3 / A♭3	3290
30	0, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54	G♯2 / A♭2, A2, A♯2 / B♭2, B2, C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F3, F♯3 / G♭3	3322
31	0, 43, 44, 45, 46, 48, 49, 50, 51, 54, 55, 56	G2, G♯2 / A♭2, A2, A♯2 / B♭2, C3, C♯3 / D♭3, D3, D♯3 / E♭3, F3, F♯3 / G♭3, G3, G♯3 / A♭3	3411
32	0, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54	G♯2 / A♭2, A2, A♯2 / B♭2, B2, C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F3, F♯3 / G♭3	3541

LAMPIRAN C

Hasil Pengujian *Distance* dengan *Dynamic Time Warping* dan *Equivalent Pitch Class*

Tabel C-3 Hasil pengujian *distance* pada lagu Happy Birthday di model pertama

No	Unique pitch	Equivalent pitch class	Distance
33	0, 41, 42, 43, 44, 45, 46, 47, 48, 49, 51, 53, 54, 55	F2, F \sharp 2 / G \flat 2, G2, G \sharp 2 / A \flat 2, A2, A \sharp 2 / B \flat 2, B2, C3, C \sharp 3 / D \flat 3, D \sharp 3 / E \flat 3, F3, F \sharp 3 / G \flat 3, G3	3787
34	0, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52	G2, G \sharp 2 / A \flat 2, A2, A \sharp 2 / B \flat 2, B2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3	3856

Tabel C-4 menunjukkan hasil pengujian *distance* dengan DTW di lagu Brother John pada model pertama.

Tabel C-4 Hasil pengujian *distance* pada lagu Brother John di model pertama

No	Unique pitch	Equivalent pitch class	Distance
1	0, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68	A \sharp 3 / B \flat 3, B3, C4, C \sharp 4 / D \flat 4, D4, D \sharp 4 / E \flat 4, E4, F4, F \sharp 4 / G \flat 4, G4, G \sharp 4 / A \flat 4	185
2	0, 54, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67	F \sharp 3 / G \flat 3, G \sharp 3 / A \flat 3, A3, A \sharp 3 / B \flat 3, B3, C4, C \sharp 4 / D \flat 4, D4, D \sharp 4 / E \flat 4, E4, F4, F \sharp 4 / G \flat 4, G4	510
3	0, 47, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67	B2, G \sharp 3 / A \flat 3, A3, A \sharp 3 / B \flat 3, B3, C4, C \sharp 4 / D \flat 4, D4, D \sharp 4 / E \flat 4, E4, F4, F \sharp 4 / G \flat 4, G4	519
4	0, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69	B3, C4, C \sharp 4 / D \flat 4, D4, D \sharp 4 / E \flat 4, E4, F4, F \sharp 4 / G \flat 4, G4, G \sharp 4 / A \flat 4, A4	534
5	0, 44, 56, 58, 59, 61, 62, 63, 64, 66, 67, 69	G \sharp 2 / A \flat 2, G \sharp 3 / A \flat 3, A \sharp 3 / B \flat 3, B3, C4, C \sharp 4 / D \flat 4, D4, D \sharp 4 / E \flat 4, E4, F4, F \sharp 4 / G \flat 4, G4, A4	669
6	0, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70	B3, C4, C \sharp 4 / D \flat 4, D4, D \sharp 4 / E \flat 4, E4, F4, F \sharp 4 / G \flat 4, G4, G \sharp 4 / A \flat 4, A4, A \sharp 4 / B \flat 4	705
7	0, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67	G \sharp 3 / A \flat 3, A3, A \sharp 3 / B \flat 3, B3, C4, C \sharp 4 / D \flat 4, D4, D \sharp 4 / E \flat 4, E4, F4, F \sharp 4 / G \flat 4, G4	1076
8	0, 47, 48, 49, 51, 52, 54, 55, 56, 57, 58, 59, 60, 61, 64	B2, C3, C \sharp 3 / D \flat 3, D \sharp 3 / E \flat 3, E3, F \sharp 3 / G \flat 3, G3, G \sharp 3 / A \flat 3, A3, A \sharp 3 / B \flat 3, B3, C4, C \sharp 4 / D \flat 4, E4	1657
9	0, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 60, 61	G2, G \sharp 2 / A \flat 2, A2, A \sharp 2 / B \flat 2, B2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3, F3, F \sharp 3 / G \flat 3, G3, G \sharp 3 / A \flat 3, A3, A \sharp 3 / B \flat 3, C4, C \sharp 4 / D \flat 4	1770
10	0, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58	A2, A \sharp 2 / B \flat 2, B2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3, F3, F \sharp 3 / G \flat 3, G3, G \sharp 3 / A \flat 3, A3	1986
11	0, 48, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59	C3, D3, D \sharp 3 / E \flat 3, E3, F3, F \sharp 3 / G \flat 3, G3, G \sharp 3 / A \flat 3, A3, A \sharp 3 / B \flat 3, B3	2008
12	0, 45, 46, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59	A2, A \sharp 2 / B \flat 2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3, F3, F \sharp 3 / G \flat 3, G3, G \sharp 3 / A \flat 3, A3, A \sharp 3 / B \flat 3, B3	2031
13	0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58	A \sharp 2 / B \flat 2, B2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3, F3, F \sharp 3 / G \flat 3, G3, G \sharp 3 / A \flat 3, A3, A \sharp 3 / B \flat 3	2149
14	0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58	A \sharp 2 / B \flat 2, B2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3, F3, F \sharp 3 / G \flat 3, G3, G \sharp 3 / A \flat 3, A3, A \sharp 3 / B \flat 3	2238
15	0, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 61, 62	B2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3, F3, F \sharp 3 / G \flat 3, G3, G \sharp 3 / A \flat 3, A3, A \sharp 3 / B \flat 3, C \sharp 4 / D \flat 4	2327
16	0, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57	G2, G \sharp 2 / A \flat 2, A2, A \sharp 2 / B \flat 2, B2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3, F3, F \sharp 3 / G \flat 3, G3, G \sharp 3 / A \flat 3, A3	2414

LAMPIRAN C

Hasil Pengujian *Distance* dengan *Dynamic Time Warping* dan *Equivalent Pitch Class*

Tabel C-4 Hasil pengujian *distance* pada lagu Brother John di model pertama

No	Unique pitch	Equivalent pitch class	Distance
17	0, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58	B2, C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F3, F♯3 / G♭3, G3, G♯3 / A♭3, A3, A♯3 / B♭3	2415
18	0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57	A♯2 / B♭2, B2, C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F3, F♯3 / G♭3, G3, G♯3 / A♭3, A3	2431
19	0, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 61	A2, A♯2 / B♭2, B2, C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F3, F♯3 / G♭3, G3, G♯3 / A♭3, A3, C♯4 / D♭4	2457
20	0, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69	B3, C4, C♯4 / D♭4, D4, D♯4 / E♭4, E4, F4, F♯4 / G♭4, G4, G♯4 / A♭4, A4	2493
21	0, 47, 48, 49, 50, 51, 52, 53, 54, 55, 57, 59	B2, C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F3, F♯3 / G♭3, G3, A3, B3	2520
22	0, 49, 50, 51, 52, 53, 54, 55, 56, 57	C♯3 / D♭3, D3, D♯3 / E♭3, E3, F3, F♯3 / G♭3, G3, G♯3 / A♭3, A3	2590
23	0, 45, 46, 47, 49, 50, 51, 52, 53, 54, 55	A2, A♯2 / B♭2, B2, C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F3, F♯3 / G♭3, G3	2651
24	0, 44, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 58, 61	G♯2 / A♭2, B2, C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F3, F♯3 / G♭3, G3, G♯3 / A♭3, A3, C♯4 / D♭4	2664
25	0, 44, 45, 46, 47, 48, 49, 50, 51	G♯2 / A♭2, A2, A♯2 / B♭2, B2, C3, C♯3 / D♭3, D3, D♯3 / E♭3	2697
26	0, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58	A2, A♯2 / B♭2, B2, C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F3, F♯3 / G♭3, G3, G♯3 / A♭3, A3, A♯3 / B♭3	2702
27	0, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55	G♯2 / A♭2, A2, A♯2 / B♭2, B2, C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F3, F♯3 / G♭3, G3	2706
28	0, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53	G2, G♯2 / A♭2, A2, A♯2 / B♭2, B2, C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F3	2767
29	0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56	A♯2 / B♭2, B2, C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F3, F♯3 / G♭3, G3, G♯3 / A♭3	2829
30	0, 42, 43, 44, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56	F♯2 / G♭2, G2, G♯2 / A♭2, A♯2 / B♭2, B2, C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F3, F♯3 / G♭3, G3, G♯3 / A♭3	2859
31	0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56	A♯2 / B♭2, B2, C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F3, F♯3 / G♭3, G3, G♯3 / A♭3	2921
32	0, 43, 44, 46, 47, 48, 49, 50, 51, 52, 53	G2, G♯2 / A♭2, A♯2 / B♭2, B2, C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F3	2981
33	0, 44, 45, 46, 47, 48, 49, 50, 51, 52, 54	G♯2 / A♭2, A2, A♯2 / B♭2, B2, C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F♯3 / G♭3	3217
34	0, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51	F2, F♯2 / G♭2, G2, G♯2 / A♭2, A2, A♯2 / B♭2, B2, C3, C♯3 / D♭3, D3, D♯3 / E♭3	3562
35	0, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57	G2, G♯2 / A♭2, A2, A♯2 / B♭2, B2, C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F3, F♯3 / G♭3, G3, G♯3 / A♭3, A3	3732

Tabel [C-5](#) menunjukkan hasil pengujian *distance* dengan DTW di lagu London Bridge is Falling Down pada model pertama.

LAMPIRAN C

Hasil Pengujian *Distance* dengan *Dynamic Time Warping* dan *Equivalent Pitch Class*

Tabel C-5 Hasil pengujian *distance* pada lagu London Bridge is Falling Down di model pertama

No	Unique pitch	Equivalent pitch class	Distance
1	0, 56, 58, 59, 60, 61, 62, 64	G♯3 / Ab3, A♯3 / B♭3, B3, C4, C♯4 / D♭4, D4, E4	10
2	0, 55, 56, 58, 59, 60, 61, 63	G3, G♯3 / Ab3, A♯3 / B♭3, B3, C4, C♯4 / D♭4, D♯4 / E♭4	389
3	0, 49, 57, 58, 59, 60, 61, 62, 63, 64	C♯3 / Db3, A3, A♯3 / B♭3, B3, C4, C♯4 / D♭4, D4, D♯4 / E♭4, E4	435
4	0, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64	G3, G♯3 / Ab3, A3, A♯3 / B♭3, B3, C4, C♯4 / D♭4, D4, D♯4 / E♭4, E4	842
5	0, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62	F3, F♯3 / G♭3, G3, G♯3 / Ab3, A3, A♯3 / B♭3, B3, C4, C♯4 / D♭4, D4	1068
6	0, 53, 54, 55, 56, 57, 58, 59, 60, 61	F3, F♯3 / G♭3, G3, G♯3 / Ab3, A3, A♯3 / B♭3, B3, C4, C♯4 / D♭4	1156
7	0, 45, 53, 54, 55, 56, 57, 58, 59, 60	A2, F3, F♯3 / G♭3, G3, G♯3 / Ab3, A3, A♯3 / B♭3, B3, C4	1195
8	0, 50, 51, 52, 53, 54, 55, 56, 57, 58	D3, D♯3 / E♭3, E3, F3, F♯3 / G♭3, G3, G♯3 / Ab3, A3, A♯3 / B♭3	1287
9	0, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60	D♯3 / Eb3, E3, F3, F♯3 / G♭3, G3, G♯3 / Ab3, A3, A♯3 / B♭3, B3, C4	1427
10	0, 50, 51, 52, 53, 54, 55, 56, 57, 58	D3, D♯3 / Eb3, E3, F3, F♯3 / G♭3, G3, G♯3 / Ab3, A3, A♯3 / B♭3	1503
11	0, 50, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61	D3, E3, F3, F♯3 / G♭3, G3, G♯3 / Ab3, A3, A♯3 / B♭3, B3, C4, C♯4 / D♭4	1615
12	0, 42, 43, 49, 50, 51, 52, 53, 54, 55, 56	F♯2 / G♭2, G2, C♯3 / Db3, D3, D♯3 / Eb3, E3, F3, F♯3 / G♭3, G3, G♯3 / Ab3	1616
13	0, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58	C3, C♯3 / Db3, D3, D♯3 / Eb3, E3, F3, F♯3 / G♭3, G3, G♯3 / Ab3, A3, A♯3 / B♭3	1811
14	0, 48, 49, 51, 52, 53, 54, 55, 56, 57	C3, C♯3 / Db3, D♯3 / Eb3, E3, F3, F♯3 / G♭3, G3, G♯3 / Ab3, A3	1826
15	0, 48, 49, 50, 51, 52, 53, 54, 55, 56	C3, C♯3 / Db3, D3, D♯3 / Eb3, E3, F3, F♯3 / G♭3, G3, G♯3 / Ab3	1916
16	0, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57	B2, C3, C♯3 / Db3, D3, D♯3 / Eb3, E3, F3, F♯3 / G♭3, G3, G♯3 / Ab3, A3	1980
17	0, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 58	A2, A♯2 / B♭2, B2, C3, C♯3 / Db3, D3, D♯3 / Eb3, E3, F3, F♯3 / G♭3, G3, A♯3 / B♭3	2009
18	0, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57	C3, C♯3 / Db3, D3, D♯3 / Eb3, E3, F3, F♯3 / G♭3, G3, G♯3 / Ab3, A3	2080
19	0, 44, 45, 46, 48, 49, 50, 51, 52, 53	G♯2 / Ab2, A2, A♯2 / B♭2, C3, C♯3 / Db3, D3, D♯3 / Eb3, E3, F3	2093
20	0, 45, 46, 48, 49, 50, 51, 52, 53, 54	A2, A♯2 / B♭2, C3, C♯3 / Db3, D3, D♯3 / Eb3, E3, F3, F♯3 / G♭3	2173
21	0, 36, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53	C2, G♯2 / Ab2, A2, A♯2 / B♭2, B2, C3, C♯3 / Db3, D3, D♯3 / Eb3, E3, F3	2226
22	0, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58	C3, C♯3 / Db3, D3, D♯3 / Eb3, E3, F3, F♯3 / G♭3, G3, G♯3 / Ab3, A3, A♯3 / B♭3	2237
23	0, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64	G3, G♯3 / Ab3, A3, A♯3 / B♭3, B3, C4, C♯4 / D♭4, D4, D♯4 / Eb4, E4	2273

LAMPIRAN C

Hasil Pengujian *Distance* dengan *Dynamic Time Warping* dan *Equivalent Pitch Class*

Tabel C-5 Hasil pengujian *distance* pada lagu London Bridge is Falling Down di model pertama

No	Unique pitch	Equivalent pitch class	Distance
24	0, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55	A2, A \sharp 2 / B \flat 2, B2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3, F3, F \sharp 3 / G \flat 3, G3	2359
25	0, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53	G \sharp 2 / A \flat 2, A2, A \sharp 2 / B \flat 2, B2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3, F3	2573
26	0, 45, 46, 47, 48, 49, 50, 51, 52, 53	A2, A \sharp 2 / B \flat 2, B2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3, F3	2647
27	0, 45, 46, 47, 48, 49, 50, 51, 52, 53	A2, A \sharp 2 / B \flat 2, B2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3, F3	2765
28	0, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56	B2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3, F3, F \sharp 3 / G \flat 3, G3, G \sharp 3 / A \flat 3	3490
29	0, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56	A2, A \sharp 2 / B \flat 2, B2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3, F3, F \sharp 3 / G \flat 3, G3, G \sharp 3 / A \flat 3	3804

Tabel **C-6** menunjukkan hasil pengujian *distance* dengan DTW di lagu Twinkle-Twinkle Little Star pada model kedua.

Tabel C-6 Hasil pengujian *distance* pada lagu Twinkle-Twinkle Little Star di model kedua

No	Unique pitch	Equivalent pitch class	Distance
1	0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57	A \sharp 2 / B \flat 2, B2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3, F3, F \sharp 3 / G \flat 3, G3, G \sharp 3 / A \flat 3, A3	112
2	0, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56	G \sharp 2 / A \flat 2, A2, A \sharp 2 / B \flat 2, B2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3, F3, F \sharp 3 / G \flat 3, G3, G \sharp 3 / A \flat 3	489
3	0, 48, 49, 50, 52, 53, 54, 55, 56, 57, 58, 59	C3, C \sharp 3 / D \flat 3, D3, E3, F3, F \sharp 3 / G \flat 3, G3, G \sharp 3 / A \flat 3, A3, A \sharp 3 / B \flat 3, B3	570
4	0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55	A \sharp 2 / B \flat 2, B2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3, F3, F \sharp 3 / G \flat 3, G3	676
5	0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56	A \sharp 2 / B \flat 2, B2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3, F3, F \sharp 3 / G \flat 3, G3, G \sharp 3 / A \flat 3	726
6	0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56	A \sharp 2 / B \flat 2, B2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3, F3, F \sharp 3 / G \flat 3, G3, G \sharp 3 / A \flat 3	732
7	0, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54	G \sharp 2 / A \flat 2, A2, A \sharp 2 / B \flat 2, B2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3, F3, F \sharp 3 / G \flat 3	1033
8	0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57	A \sharp 2 / B \flat 2, B2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3, F3, F \sharp 3 / G \flat 3, G3, G \sharp 3 / A \flat 3, A3	1037
9	0, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55	A2, A \sharp 2 / B \flat 2, B2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3, F3, F \sharp 3 / G \flat 3, G3	1040
10	0, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54	G2, G \sharp 2 / A \flat 2, A2, A \sharp 2 / B \flat 2, B2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3, F3, F \sharp 3 / G \flat 3	1070
11	0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57	A \sharp 2 / B \flat 2, B2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3, F3, F \sharp 3 / G \flat 3, G3, G \sharp 3 / A \flat 3, A3	1158
12	0, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55	A2, A \sharp 2 / B \flat 2, B2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3, F3, F \sharp 3 / G \flat 3, G3	1180
13	0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56	A \sharp 2 / B \flat 2, B2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3, F3, F \sharp 3 / G \flat 3, G3, G \sharp 3 / A \flat 3	1214

LAMPIRAN C

Hasil Pengujian *Distance* dengan *Dynamic Time Warping* dan *Equivalent Pitch Class*

Tabel C-6 Hasil pengujian *distance* pada lagu Twinkle-Twinkle Little Star di model kedua

No	Unique pitch	Equivalent pitch class	Distance
14	0, 44, 45, 46, 47, 48, 49, 50, 51, 52, 54, 55, 56, 57, 58, 59	G \sharp 2 / A \flat 2, A2, A \sharp 2 / B \flat 2, B2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3, F \sharp 3 / G \flat 3, G3, G \sharp 3 / A \flat 3, A3, A \sharp 3 / B \flat 3, B3	1249
15	0, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52	F \sharp 2 / G \flat 2, G2, G \sharp 2 / A \flat 2, A2, A \sharp 2 / B \flat 2, B2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3	1283
16	0, 45, 46, 48, 49, 50, 51, 52, 53, 54, 55, 56	A2, A \sharp 2 / B \flat 2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3, F3, F \sharp 3 / G \flat 3, G3, G \sharp 3 / A \flat 3	1459
17	0, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54	G \sharp 2 / A \flat 2, A2, A \sharp 2 / B \flat 2, B2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3, F3, F \sharp 3 / G \flat 3	1474
18	0, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52	E2, F2, F \sharp 2 / G \flat 2, G2, G \sharp 2 / A \flat 2, A2, A \sharp 2 / B \flat 2, B2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3	1502
19	0, 48, 49, 50, 52, 53, 54, 55, 56, 57, 58, 59	C3, C \sharp 3 / D \flat 3, D3, E3, F3, F \sharp 3 / G \flat 3, G3, G \sharp 3 / A \flat 3, A3, A \sharp 3 / B \flat 3, B3	1742
20	0, 43, 44, 45, 46, 47, 48, 49, 50	G2, G \sharp 2 / A \flat 2, A2, A \sharp 2 / B \flat 2, B2, C3, C \sharp 3 / D \flat 3, D3	1764
21	0, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54	F2, F \sharp 2 / G \flat 2, G2, G \sharp 2 / A \flat 2, A2, A \sharp 2 / B \flat 2, B2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3, F3, F \sharp 3 / G \flat 3	1791
22	0, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63	F \sharp 3 / G \flat 3, G3, G \sharp 3 / A \flat 3, A3, A \sharp 3 / B \flat 3, B3, C4, C \sharp 4 / D \flat 4, D4, D \sharp 4 / E \flat 4	1804
23	0, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64	F \sharp 3 / G \flat 3, G3, G \sharp 3 / A \flat 3, A3, A \sharp 3 / B \flat 3, B3, C4, C \sharp 4 / D \flat 4, D4, D \sharp 4 / E \flat 4, E4	2028
24	0, 44, 45, 46, 47, 48, 49, 50, 51, 52, 54	G \sharp 2 / A \flat 2, A2, A \sharp 2 / B \flat 2, B2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3, F \sharp 3 / G \flat 3	2080
25	0, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62	E3, F3, F \sharp 3 / G \flat 3, G3, G \sharp 3 / A \flat 3, A3, A \sharp 3 / B \flat 3, B3, C4, C \sharp 4 / D \flat 4, D4	2118
26	0, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54	G \sharp 2 / A \flat 2, A2, A \sharp 2 / B \flat 2, B2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3, F3, F \sharp 3 / G \flat 3	2130
27	0, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53	F \sharp 2 / G \flat 2, G2, G \sharp 2 / A \flat 2, A2, A \sharp 2 / B \flat 2, B2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3, F3	2147
28	0, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66	A3, A \sharp 3 / B \flat 3, B3, C4, C \sharp 4 / D \flat 4, D4, D \sharp 4 / E \flat 4, E4, F4, F \sharp 4 / G \flat 4	2399
29	0, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68	A3, A \sharp 3 / B \flat 3, B3, C4, C \sharp 4 / D \flat 4, D4, D \sharp 4 / E \flat 4, E4, F4, F \sharp 4 / G \flat 4, G4, G \sharp 4 / A \flat 4	2429
30	0, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66	G \sharp 3 / A \flat 3, A3, A \sharp 3 / B \flat 3, B3, C4, C \sharp 4 / D \flat 4, D4, D \sharp 4 / E \flat 4, E4, F4, F \sharp 4 / G \flat 4	2636
31	0, 57, 58, 59, 60, 62, 63, 64, 65, 66	A3, A \sharp 3 / B \flat 3, B3, C4, D4, D \sharp 4 / E \flat 4, E4, F4, F \sharp 4 / G \flat 4	2695
32	0, 54, 56, 57, 58, 59, 60, 62, 64	F \sharp 3 / G \flat 3, G \sharp 3 / A \flat 3, A3, A \sharp 3 / B \flat 3, B3, C4, D4, E4	2975

Tabel C-7 menunjukkan hasil pengujian *distance* dengan DTW di lagu Old McDonald pada model kedua.

LAMPIRAN C

Hasil Pengujian *Distance* dengan *Dynamic Time Warping* dan *Equivalent Pitch Class*

Tabel C-7 Hasil pengujian *distance* pada lagu Old McDonald di model kedua

No	Unique pitch	Equivalent pitch class	Distance
1	0, 52, 56, 57, 58, 59, 62, 63, 64, 65, 66, 67	E3, G \sharp 3 / Ab3, A3, A \sharp 3 / Bb3, B3, D4, D \sharp 4 / Eb4, E4, F4, F \sharp 4 / Gb4, G4	133
2	0, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68	G \sharp 3 / Ab3, A3, A \sharp 3 / Bb3, B3, C4, C \sharp 4 / Db4, D4, D \sharp 4 / Eb4, E4, F4, F \sharp 4 / Gb4, G4, G \sharp 4 / Ab4	688
3	0, 49, 56, 57, 58, 59, 61, 62, 63, 64, 65, 66	C \sharp 3 / Db3, G \sharp 3 / Ab3, A3, A \sharp 3 / Bb3, B3, C \sharp 4 / Db4, D4, D \sharp 4 / Eb4, E4, F4, F \sharp 4 / Gb4	744
4	0, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63	F3, F \sharp 3 / Gb3, G3, G \sharp 3 / Ab3, A3, A \sharp 3 / Bb3, B3, C4, C \sharp 4 / Db4, D4, D \sharp 4 / Eb4	813
5	0, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64	G3, G \sharp 3 / Ab3, A3, A \sharp 3 / Bb3, B3, C4, C \sharp 4 / Db4, D4, D \sharp 4 / Eb4, E4	871
6	0, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65	G \sharp 3 / Ab3, A3, A \sharp 3 / Bb3, B3, C4, C \sharp 4 / Db4, D4, D \sharp 4 / Eb4, E4, F4	926
7	0, 59, 60, 61, 62, 63, 64, 66, 67, 68, 69, 70	B3, C4, C \sharp 4 / Db4, D4, D \sharp 4 / Eb4, E4, F \sharp 4 / Gb4, G4, G \sharp 4 / Ab4, A4, A \sharp 4 / Bb4	1044
8	0, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64	D \sharp 3 / Eb3, E3, F3, F \sharp 3 / Gb3, G3, G \sharp 3 / Ab3, A3, A \sharp 3 / Bb3, B3, C4, C \sharp 4 / Db4, D4, D \sharp 4 / Eb4, E4	1100
9	0, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62	D \sharp 3 / Eb3, E3, F3, F \sharp 3 / Gb3, G3, G \sharp 3 / Ab3, A3, A \sharp 3 / Bb3, B3, C4, C \sharp 4 / Db4, D4	1211
10	0, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59	D3, D \sharp 3 / Eb3, E3, F3, F \sharp 3 / Gb3, G3, G \sharp 3 / Ab3, A3, A \sharp 3 / Bb3, B3	1579
11	0, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60	B2, C3, C \sharp 3 / Db3, D3, D \sharp 3 / Eb3, E3, F3, F \sharp 3 / Gb3, G3, G \sharp 3 / Ab3, A3, A \sharp 3 / Bb3, B3, C4	1703
12	0, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60	C3, C \sharp 3 / Db3, D3, D \sharp 3 / Eb3, E3, F \sharp 3 / Gb3, G3, G \sharp 3 / Ab3, A3, A \sharp 3 / Bb3, B3, C4	1817
13	0, 48, 49, 50, 53, 54, 55, 56, 57, 58	C3, C \sharp 3 / Db3, D3, F3, F \sharp 3 / Gb3, G3, G \sharp 3 / Ab3, A3, A \sharp 3 / Bb3	1953
14	0, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58	B2, C3, C \sharp 3 / Db3, D3, D \sharp 3 / Eb3, E3, F3, F \sharp 3 / Gb3, G3, G \sharp 3 / Ab3, A3, A \sharp 3 / Bb3	2014
15	0, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58	B2, C3, C \sharp 3 / Db3, D3, D \sharp 3 / Eb3, E3, F3, F \sharp 3 / Gb3, G3, G \sharp 3 / Ab3, A3, A \sharp 3 / Bb3	2299
16	0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56	A \sharp 2 / Bb2, B2, C3, C \sharp 3 / Db3, D3, D \sharp 3 / Eb3, E3, F3, F \sharp 3 / Gb3, G3, G \sharp 3 / Ab3	2361
17	0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56	A \sharp 2 / Bb2, B2, C3, C \sharp 3 / Db3, D3, D \sharp 3 / Eb3, E3, F3, F \sharp 3 / Gb3, G3, G \sharp 3 / Ab3	2593
18	0, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53	F \sharp 2 / Gb2, G2, G \sharp 2 / Ab2, A2, A \sharp 2 / Bb2, B2, C3, C \sharp 3 / Db3, D3, D \sharp 3 / Eb3, E3, F3	2662
19	0, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55	G \sharp 2 / Ab2, A2, A \sharp 2 / Bb2, B2, C3, C \sharp 3 / Db3, D3, D \sharp 3 / Eb3, E3, F3	2687
20	0, 42, 43, 44, 45, 47, 48, 49, 50, 51, 52	F \sharp 2 / Gb2, G2, G \sharp 2 / Ab2, A2, B2, C3, C \sharp 3 / Db3, D3, D \sharp 3 / Eb3, E3	2785
21	0, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53	F \sharp 2 / Gb2, G2, G \sharp 2 / Ab2, A2, A \sharp 2 / Bb2, B2, C3, C \sharp 3 / Db3, D3, D \sharp 3 / Eb3, E3, F3	2816
22	0, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53	G2, G \sharp 2 / Ab2, A2, A \sharp 2 / Bb2, B2, C3, C \sharp 3 / Db3, D3, D \sharp 3 / Eb3, E3, F3	2821

LAMPIRAN C

Hasil Pengujian *Distance* dengan *Dynamic Time Warping* dan *Equivalent Pitch Class*

Tabel C-7 Hasil pengujian *distance* pada lagu Old McDonald di model kedua

No	Unique pitch	Equivalent pitch class	Distance
23	0, 48, 49, 50, 51, 53, 54, 55, 56, 57, 58	C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, F3, F \sharp 3 / G \flat 3, G3, G \sharp 3 / A \flat 3, A3, A \sharp 3 / B \flat 3	2888
24	0, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53	G \sharp 2 / A \flat 2, A2, A \sharp 2 / B \flat 2, B2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3, F3	2928
25	0, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54	F2, F \sharp 2 / G \flat 2, G2, G \sharp 2 / A \flat 2, A2, A \sharp 2 / B \flat 2, B2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3, F3, F \sharp 3 / G \flat 3	2983
26	0, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52	F2, F \sharp 2 / G \flat 2, G2, G \sharp 2 / A \flat 2, A2, A \sharp 2 / B \flat 2, B2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3	3036
27	0, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52	F2, F \sharp 2 / G \flat 2, G2, G \sharp 2 / A \flat 2, A2, A \sharp 2 / B \flat 2, B2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3	3146
28	0, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52	F \sharp 2 / G \flat 2, G2, G \sharp 2 / A \flat 2, A2, A \sharp 2 / B \flat 2, B2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3	3321
29	0, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54	F \sharp 2 / G \flat 2, G2, G \sharp 2 / A \flat 2, A2, A \sharp 2 / B \flat 2, B2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3, F3, F \sharp 3 / G \flat 3	3363
30	0, 43, 44, 45, 46, 47, 48, 49, 51, 52, 53, 54	G2, G \sharp 2 / A \flat 2, A2, A \sharp 2 / B \flat 2, B2, C3, C \sharp 3 / D \flat 3, D \sharp 3 / E \flat 3, E3, F3, F \sharp 3 / G \flat 3	3403
31	0, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51	F \sharp 2 / G \flat 2, G2, G \sharp 2 / A \flat 2, A2, A \sharp 2 / B \flat 2, B2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3	4093

Tabel **C-8** menunjukkan hasil pengujian *distance* dengan DTW di lagu Happy Birthday pada model kedua.

Tabel C-8 Hasil pengujian *distance* pada lagu Happy Birthday di model kedua

No	Unique pitch	Equivalent pitch class	Distance
1	0, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67	A \sharp 3 / B \flat 3, B3, C4, C \sharp 4 / D \flat 4, D4, D \sharp 4 / E \flat 4, E4, F4, F \sharp 4 / G \flat 4, G4	147
2	0, 56, 57, 58, 59, 60, 61, 63, 64, 65, 66	G \sharp 3 / A \flat 3, A3, A \sharp 3 / B \flat 3, B3, C4, C \sharp 4 / D \flat 4, D \sharp 4 / E \flat 4, E4, F4, F \sharp 4 / G \flat 4	294
3	0, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66	F3, F \sharp 3 / G \flat 3, G3, G \sharp 3 / A \flat 3, A3, A \sharp 3 / B \flat 3, B3, C4, C \sharp 4 / D \flat 4, D4, D \sharp 4 / E \flat 4, E4, F4, F \sharp 4 / G \flat 4	813
4	0, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67	A \sharp 3 / B \flat 3, B3, C4, C \sharp 4 / D \flat 4, D4, D \sharp 4 / E \flat 4, E4, F4, F \sharp 4 / G \flat 4, G4	947
5	0, 57, 58, 59, 60, 61, 62, 63, 64, 66, 67, 68	A3, A \sharp 3 / B \flat 3, B3, C4, C \sharp 4 / D \flat 4, D4, D \sharp 4 / E \flat 4, E4, F \sharp 4 / G \flat 4, G4, G \sharp 4 / A \flat 4	1056
6	0, 59, 60, 61, 62, 64, 65, 67, 68	B3, C4, C \sharp 4 / D \flat 4, D4, E4, F4, G4, G \sharp 4 / A \flat 4	1081
7	0, 54, 55, 56, 57, 59, 60, 62, 67	F \sharp 3 / G \flat 3, G3, G \sharp 3 / A \flat 3, A3, B3, C4, D4, G4	1329
8	0, 47, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 66, 67, 68	B2, F \sharp 3 / G \flat 3, G3, G \sharp 3 / A \flat 3, A3, A \sharp 3 / B \flat 3, B3, C4, C \sharp 4 / D \flat 4, D4, D \sharp 4 / E \flat 4, E4, F \sharp 4 / G \flat 4, G4, G \sharp 4 / A \flat 4	1417
9	0, 53, 54, 55, 56, 57, 58, 59, 60	F3, F \sharp 3 / G \flat 3, G3, G \sharp 3 / A \flat 3, A3, A \sharp 3 / B \flat 3, B3, C4	1747
10	0, 51, 52, 53, 54, 55, 56, 57, 58, 59	D \sharp 3 / E \flat 3, E3, F3, F \sharp 3 / G \flat 3, G3, G \sharp 3 / A \flat 3, A3, A \sharp 3 / B \flat 3, B3	1841

LAMPIRAN C

Hasil Pengujian *Distance* dengan *Dynamic Time Warping* dan *Equivalent Pitch Class*

Tabel C-8 Hasil pengujian *distance* pada lagu Happy Birthday di model kedua

No	Unique pitch	Equivalent pitch class	Distance
11	0, 46, 47, 48, 49, 51, 52, 53, 54, 55, 56, 57, 58, 59	A♯2 / B♭2, B2, C3, C♯3 / D♭3, D♯3 / E♭3, E3, F3, F♯3 / G♭3, G3, G♯3 / A♭3, A3, A♯3 / B♭3, B3	2415
12	0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60	A♯2 / B♭2, B2, C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F3, F♯3 / G♭3, G3, G♯3 / A♭3, A3, A♯3 / B♭3, B3, C4	2433
13	0, 48, 49, 50, 51, 53, 54, 56, 57	C3, C♯3 / D♭3, D3, D♯3 / E♭3, F3, F♯3 / G♭3, G♯3 / A♭3, A3	2439
14	0, 48, 49, 50, 51, 52, 54, 55, 56	C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F♯3 / G♭3, G3, G♯3 / A♭3	2636
15	0, 46, 47, 48, 49, 50, 51, 52, 53, 54	A♯2 / B♭2, B2, C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F3, F♯3 / G♭3	2643
16	0, 45, 46, 47, 48, 50, 51, 52, 53, 54	A2, A♯2 / B♭2, B2, C3, D3, D♯3 / E♭3, E3, F3, F♯3 / G♭3	2650
17	0, 45, 46, 47, 48, 50, 51, 52, 53, 54, 55, 56, 57, 58	A2, A♯2 / B♭2, B2, C3, D3, D♯3 / E♭3, E3, F3, F♯3 / G♭3, G3, G♯3 / A♭3, A3, A♯3 / B♭3	2675
18	0, 46, 47, 48, 49, 50, 52, 53, 54, 55	A♯2 / B♭2, B2, C3, C♯3 / D♭3, D3, E3, F3, F♯3 / G♭3, G3	2753
19	0, 46, 47, 48, 49, 50, 52, 53, 54, 55	A♯2 / B♭2, B2, C3, C♯3 / D♭3, D3, E3, F3, F♯3 / G♭3, G3	2837
20	0, 44, 45, 46, 47, 48, 50, 51, 52, 53, 55, 56, 58	G♯2 / A♭2, A2, A♯2 / B♭2, B2, C3, D3, D♯3 / E♭3, E3, F3, G3, G♯3 / A♭3, A♯3 / B♭3	2840
21	0, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55	G♯2 / A♭2, A2, A♯2 / B♭2, B2, C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F3, F♯3 / G♭3, G3	2877
22	0, 44, 45, 46, 47, 49, 50, 51, 52, 54, 55, 56, 57, 58	G♯2 / A♭2, A2, A♯2 / B♭2, B2, C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F3, F♯3 / G♭3, G3, G♯3 / A♭3, A3, A♯3 / B♭3	2905
23	0, 44, 45, 46, 47, 48, 49, 50, 51, 52, 54, 55, 56, 57, 58	G♯2 / A♭2, A2, A♯2 / B♭2, B2, C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F3, F♯3 / G♭3, G3, G♯3 / A♭3, A3, A♯3 / B♭3	2929
24	0, 44, 45, 46, 47, 49, 50, 51, 52, 53	G♯2 / A♭2, A2, A♯2 / B♭2, B2, C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F3	2995
25	0, 46, 47, 48, 49, 51, 52, 53, 54, 55, 56, 57, 58, 59	A♯2 / B♭2, B2, C3, C♯3 / D♭3, D♯3 / E♭3, E3, F3, F♯3 / G♭3, G3, G♯3 / A♭3, A3, A♯3 / B♭3, B3	3023
26	0, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58	G2, G♯2 / A♭2, A2, A♯2 / B♭2, B2, C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F3, F♯3 / G♭3, G3, G♯3 / A♭3, A3, A♯3 / B♭3	3040
27	0, 43, 44, 45, 46, 48, 49, 50, 51, 52, 54, 55, 56	G2, G♯2 / A♭2, A2, A♯2 / B♭2, C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F3, F♯3 / G♭3, G3, G♯3 / A♭3	3122
28	0, 45, 46, 47, 48, 50, 51, 52, 53, 54	A2, A♯2 / B♭2, B2, C3, D3, D♯3 / E♭3, E3, F3, F♯3 / G♭3	3145
29	0, 45, 46, 47, 48, 49, 50, 51, 52	A2, A♯2 / B♭2, B2, C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3	3300
30	0, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54	G♯2 / A♭2, A2, A♯2 / B♭2, B2, C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F3, F♯3 / G♭3	3311
31	0, 43, 44, 45, 46, 48, 49, 50, 51, 54, 55	G2, G♯2 / A♭2, A2, A♯2 / B♭2, C3, C♯3 / D♭3, D3, D♯3 / E♭3, F3, F♯3 / G♭3, G3	3338
32	0, 44, 45, 46, 47, 49, 50, 51, 52, 53, 54	G♯2 / A♭2, A2, A♯2 / B♭2, B2, C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F3, F♯3 / G♭3	3364
33	0, 41, 42, 43, 44, 45, 46, 47, 48, 49, 51, 52, 54, 55	F2, F♯2 / G♭2, G2, G♯2 / A♭2, A2, A♯2 / B♭2, B2, C3, C♯3 / D♭3, D♯3 / E♭3, E3, F3, F♯3 / G♭3, G3	3753

LAMPIRAN C

Hasil Pengujian *Distance* dengan *Dynamic Time Warping* dan *Equivalent Pitch Class*

Tabel C-8 Hasil pengujian *distance* pada lagu Happy Birthday di model kedua

No	Unique pitch	Equivalent pitch class	Distance
34	0, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52	G2, G \sharp 2 / A \flat 2, A2, A \sharp 2 / B \flat 2, B2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3	3885

Tabel **C-9** menunjukkan hasil pengujian *distance* dengan DTW di lagu Brother John pada model kedua.

Tabel C-9 Hasil pengujian *distance* pada lagu Brother John di model kedua

No	Unique pitch	Equivalent pitch class	Distance
1	0, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68	A \sharp 3 / B \flat 3, B3, C4, C \sharp 4 / D \flat 4, D4, D \sharp 4 / E \flat 4, E4, F4, F \sharp 4 / G \flat 4, G4, G \sharp 4 / A \flat 4	25
2	0, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67	A3, A \sharp 3 / B \flat 3, B3, C4, C \sharp 4 / D \flat 4, D4, D \sharp 4 / E \flat 4, E4, F4, F \sharp 4 / G \flat 4, G4	450
3	0, 58, 59, 61, 62, 63, 64, 66, 67, 68	A \sharp 3 / B \flat 3, B3, C \sharp 4 / D \flat 4, D4, D \sharp 4 / E \flat 4, E4, F \sharp 4 / G \flat 4, G4, G \sharp 4 / A \flat 4	466
4	0, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67	A \sharp 3 / B \flat 3, B3, C4, C \sharp 4 / D \flat 4, D4, D \sharp 4 / E \flat 4, E4, F4, F \sharp 4 / G \flat 4, G4	469
5	0, 57, 58, 59, 60, 61, 62, 63, 64, 65, 67	A3, A \sharp 3 / B \flat 3, B3, C4, C \sharp 4 / D \flat 4, D4, D \sharp 4 / E \flat 4, E4, F4, G4	537
6	0, 49, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68	C \sharp 3 / D \flat 3, B3, C4, C \sharp 4 / D \flat 4, D4, D \sharp 4 / E \flat 4, E4, F4, F \sharp 4 / G \flat 4, G4, G \sharp 4 / A \flat 4	607
7	0, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70	B3, C4, C \sharp 4 / D \flat 4, D4, D \sharp 4 / E \flat 4, E4, F4, F \sharp 4 / G \flat 4, G4, G \sharp 4 / A \flat 4, A4, A \sharp 4 / B \flat 4	878
8	0, 48, 50, 51, 52, 54, 55, 56, 57, 58, 59, 60, 61, 64	C3, D3, D \sharp 3 / E \flat 3, E3, F \sharp 3 / G \flat 3, G3, G \sharp 3 / A \flat 3, A3, A \sharp 3 / B \flat 3, B3, C4, C \sharp 4 / D \flat 4, E4	1649
9	0, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 60	A2, A \sharp 2 / B \flat 2, B2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3, F3, F \sharp 3 / G \flat 3, G3, G \sharp 3 / A \flat 3, A3, A \sharp 3 / B \flat 3, C4	1693
10	0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59	A \sharp 2 / B \flat 2, B2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3, F3, F \sharp 3 / G \flat 3, G3, G \sharp 3 / A \flat 3, A3, A \sharp 3 / B \flat 3, B3	1899
11	0, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57	A2, A \sharp 2 / B \flat 2, B2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3, F3, F \sharp 3 / G \flat 3, G3, G \sharp 3 / A \flat 3, A3	1994
12	0, 50, 51, 52, 53, 54, 55, 56, 57, 58	D3, D \sharp 3 / E \flat 3, E3, F3, F \sharp 3 / G \flat 3, G3, G \sharp 3 / A \flat 3, A3, A \sharp 3 / B \flat 3	2013
13	0, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57	A2, A \sharp 2 / B \flat 2, B2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3, F3, F \sharp 3 / G \flat 3, G3, G \sharp 3 / A \flat 3, A3	2234
14	0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58	A \sharp 2 / B \flat 2, B2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3, F3, F \sharp 3 / G \flat 3, G3, G \sharp 3 / A \flat 3, A3, A \sharp 3 / B \flat 3	2256
15	0, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69	B3, C4, C \sharp 4 / D \flat 4, D4, D \sharp 4 / E \flat 4, E4, F4, F \sharp 4 / G \flat 4, G4, G \sharp 4 / A \flat 4, A4	2285
16	0, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 62	B2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3, F3, F \sharp 3 / G \flat 3, G3, G \sharp 3 / A \flat 3, A3, D4	2326
17	0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58	A \sharp 2 / B \flat 2, B2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3, F3, F \sharp 3 / G \flat 3, G3, G \sharp 3 / A \flat 3, A3, A \sharp 3 / B \flat 3	2338

LAMPIRAN C

Hasil Pengujian *Distance* dengan *Dynamic Time Warping* dan *Equivalent Pitch Class*

Tabel C-9 Hasil pengujian *distance* pada lagu Brother John di model kedua

No	Unique pitch	Equivalent pitch class	Distance
18	0, 47, 48, 49, 50, 51, 52, 53, 54, 55, 57, 58	B2, C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F3, F♯3 / G♭3, G3, A3, A♯3 / B♭3	2415
19	0, 42, 43, 44, 45, 46, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57	F♯2 / G♭2, G2, G♯2 / A♭2, A2, A♯2 / B♭2, C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F3, F♯3 / G♭3, G3, G♯3 / A♭3, A3	2421
20	0, 48, 49, 50, 51, 52, 53, 54, 55, 56	C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F3, F♯3 / G♭3, G3, G♯3 / A♭3	2468
21	0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56	A♯2 / B♭2, B2, C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F3, F♯3 / G♭3, G3, G♯3 / A♭3	2498
22	0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57	A♯2 / B♭2, B2, C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F3, F♯3 / G♭3, G3, G♯3 / A♭3, A3	2562
23	0, 49, 50, 51, 52, 53, 54, 55, 56, 57	C♯3 / D♭3, D3, D♯3 / E♭3, E3, F3, F♯3 / G♭3, G3, G♯3 / A♭3, A3	2583
24	0, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58	A2, A♯2 / B♭2, B2, C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F3, F♯3 / G♭3, G3, G♯3 / A♭3, A3, A♯3 / B♭3	2598
25	0, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53	G♯2 / A♭2, A2, A♯2 / B♭2, B2, C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F3	2612
26	0, 47, 48, 49, 50, 51, 52, 53, 54, 55	B2, C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F3, F♯3 / G♭3, G3	2656
27	0, 45, 46, 47, 49, 50, 51, 52, 53, 54, 55	A2, A♯2 / B♭2, B2, C3 / D♭3, D3, D♯3 / E♭3, E3, F3, F♯3 / G♭3, G3	2712
28	0, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55	G♯2 / A♭2, A2, A♯2 / B♭2, B2, C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F3, F♯3 / G♭3, G3	2775
29	0, 42, 43, 44, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56	F♯2 / G♭2, G2, G♯2 / A♭2, A♯2 / B♭2, B2, C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F3, F♯3 / G♭3, G3, G♯3 / A♭3	2794
30	0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56	A♯2 / B♭2, B2, C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F3, F♯3 / G♭3, G3, G♯3 / A♭3	2804
31	0, 44, 45, 46, 47, 48, 49, 50, 51	G♯2 / A♭2, A2, A♯2 / B♭2, B2, C3, C♯3 / D♭3, D3, D♯3 / E♭3	2839
32	0, 43, 44, 46, 47, 48, 49, 50, 51, 52, 53	G2, G♯2 / A♭2, A♯2 / B♭2, B2, C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F3	3048
33	0, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54	G♯2 / A♭2, A2, A♯2 / B♭2, B2, C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F3, F♯3 / G♭3	3220
34	0, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50	F2, F♯2 / G♭2, G2, G♯2 / A♭2, A2, A♯2 / B♭2, B2, C3, C♯3 / D♭3, D3	3545
35	0, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57	G2, G♯2 / A♭2, A2, A♯2 / B♭2, B2, C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F3, F♯3 / G♭3, G3, G♯3 / A♭3, A3	3738

Tabel C-10 menunjukkan hasil pengujian *distance* dengan DTW di lagu London Bridge is Falling Down pada model kedua.

Tabel C-10 Hasil pengujian *distance* pada lagu London Bridge is Falling Down di model kedua

No	Unique pitch	Equivalent pitch class	Distance
1	0, 56, 58, 59, 60, 61, 62, 64	G♯3 / A♭3, A♯3 / B♭3, B3, C4, C♯4 / D♭4, D4, E4	9

LAMPIRAN C

Hasil Pengujian *Distance* dengan *Dynamic Time Warping* dan *Equivalent Pitch Class*

Tabel C-10 Hasil pengujian *distance* pada lagu London Bridge is Falling Down di model kedua

No	Unique pitch	Equivalent pitch class	Distance
2	0, 57, 58, 59, 60, 61, 62, 64	A3, A♯3 / B♭3, B3, C4, C♯4 / D♭4, D4, E4	621
3	0, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64	G3, G♯3 / A♭3, A3, A♯3 / B♭3, B3, C4, C♯4 / D♭4, D4, D♯4 / E♭4, E4	800
4	0, 53, 54, 56, 57, 58, 59, 60, 62	F3, F♯3 / G♭3, G♯3 / A♭3, A3, A♯3 / B♭3, B3, C4, D4	884
5	0, 53, 54, 55, 56, 57, 58, 59, 60	F3, F♯3 / G♭3, G3, G♯3 / A♭3, A3, A♯3 / B♭3, B3, C4	1061
6	0, 55, 56, 58, 59, 60, 61, 62, 63	G3, G♯3 / A♭3, A♯3 / B♭3, B3, C4, C♯4 / D♭4, D4, D♯4 / E♭4	1342
7	0, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61	D3, D♯3 / E♭3, E3, F3, F♯3 / G♭3, G3, G♯3 / A♭3, A3, A♯3 / B♭3, B3, C4, C♯4 / D♭4	1365
8	0, 48, 49, 50, 51, 52, 53, 54, 55, 57	C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F3, F♯3 / G♭3, G3, A3	1539
9	0, 53, 54, 55, 56, 57, 58, 59, 60	F3, F♯3 / G♭3, G3, G♯3 / A♭3, A3, A♯3 / B♭3, B3, C4	1586
10	0, 50, 51, 52, 53, 54, 55, 56, 57, 58	D3, D♯3 / E♭3, E3, F3, F♯3 / G♭3, G3, G♯3 / A♭3, A3, A♯3 / B♭3	1734
11	0, 50, 51, 52, 53, 54, 55, 56, 57, 58	D3, D♯3 / E♭3, E3, F3, F♯3 / G♭3, G3, G♯3 / A♭3, A3, A♯3 / B♭3	1797
12	0, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58	C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F3, F♯3 / G♭3, G3, G♯3 / A♭3, A3, A♯3 / B♭3	1812
13	0, 55, 56, 57, 58, 59, 60, 61, 62, 64	G3, G♯3 / A♭3, A3, A♯3 / B♭3, B3, C4, C♯4 / D♭4, D4, E4	1860
14	0, 48, 49, 51, 52, 53, 54, 55, 56, 57	C3, C♯3 / D♭3, D♯3 / E♭3, E3, F3, F♯3 / G♭3, G3, G♯3 / A♭3, A3	1896
15	0, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60	D♯3 / E♭3, E3, F3, F♯3 / G♭3, G3, G♯3 / A♭3, A3, A♯3 / B♭3, B3, C4	2072
16	0, 48, 49, 50, 51, 52, 53, 54, 55, 56	C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F3, F♯3 / G♭3, G3, G♯3 / A♭3	2085
17	0, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56	A2, A♯2 / B♭2, B2, C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F3, F♯3 / G♭3, G3, G♯3 / A♭3	2095
18	0, 45, 46, 48, 49, 50, 51, 52, 53, 54	A2, A♯2 / B♭2, C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F3, F♯3 / G♭3	2148
19	0, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57	C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F3, F♯3 / G♭3, G3, G♯3 / A♭3, A3	2172
20	0, 45, 46, 47, 48, 49, 50, 51, 52, 54, 55	A2, A♯2 / B♭2, B2, C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F♯3 / G♭3, G3	2369
21	0, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53	G♯2 / A♭2, A2, A♯2 / B♭2, B2, C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F3	2370
22	0, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57	C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F3, F♯3 / G♭3, G3, G♯3 / A♭3	2421
23	0, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53	G♯2 / A♭2, A2, A♯2 / B♭2, B2, C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F3	2568
24	0, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57	B2, C3, C♯3 / D♭3, D3, D♯3 / E♭3, E3, F3, F♯3 / G♭3, G3, G♯3 / A♭3, A3	2580

LAMPIRAN C

Hasil Pengujian *Distance* dengan *Dynamic Time Warping* dan *Equivalent Pitch Class*

Tabel C-10 Hasil pengujian *distance* pada lagu London Bridge is Falling Down di model kedua

No	Unique pitch	Equivalent pitch class	Distance
25	0, 44, 45, 46, 48, 49, 50, 51, 52, 53	G \sharp 2 / A \flat 2, A2, A \sharp 2 / B \flat 2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3, F3	2660
26	0, 45, 46, 47, 48, 49, 50, 51, 52, 53	A2, A \sharp 2 / B \flat 2, B2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3, F3	2671
27	0, 45, 46, 47, 48, 49, 50, 51, 52, 53	A2, A \sharp 2 / B \flat 2, B2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3, F3	3284
28	0, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56	A2, A \sharp 2 / B \flat 2, B2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3, F3, F \sharp 3 / G \flat 3, G3, G \sharp 3 / A \flat 3	3673
29	0, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56	B2, C3, C \sharp 3 / D \flat 3, D3, D \sharp 3 / E \flat 3, E3, F3, F \sharp 3 / G \flat 3, G3, G \sharp 3 / A \flat 3	3952

LAMPIRAN D

PERBANDINGAN HASIL ESTIMASI *PITCH* DENGAN DATASET

Bagian ini menunjukkan perbandingan antara hasil estimasi *pitch* dengan *pitch* yang ada di *dataset*. Hal ini dilakukan untuk mendapatkan nilai *threshold* yang optimal untuk pengujian kemiripan dengan melihat berapa banyak data yang *match*, kurang, dan lebih.

Keterangan kolom pada Tabel [D-1](#) hingga Tabel [D-10](#) adalah sebagai berikut.

1. *Pitch result* merupakan hasil estimasi *pitch* dari rekaman.
2. *Dist* merupakan hasil *distance* antara rekaman dari *dataset* dengan rekaman yang diuji.
3. *Match* menunjukkan kelompok *pitch* pada *pitch result* yang ada di hasil estimasi dan *dataset*.
4. *Diff1* merupakan kelompok *pitch* pada *pitch result* yang tidak terestimasi pada hasil estimasi.
5. *Diff2* merupakan kelompok *pitch* pada *pitch result* yang terestimasi, padahal seharusnya tidak terestimasi.
6. *Len_Match* menunjukkan jumlah *pitch* pada *pitch result* yang ada di hasil estimasi dan *dataset*.
7. *Len_Diff1* merupakan jumlah *pitch* pada *pitch result* yang tidak terestimasi pada hasil estimasi.
8. *Len_Diff2* merupakan jumlah *pitch* pada *pitch result* yang terestimasi, padahal seharusnya tidak terestimasi.
9. *Percentage* menunjukkan perhitungan yang digunakan untuk melihat seberapa baik hasil estimasi *pitch* pada data uji. Persamaan untuk melakukan perhitungan ini dapat dilihat pada bagian [4.5.7](#).

Tabel [D-1](#) menunjukkan perbandingan *pitch detected* dan *pitch* di *dataset* pada lagu Twinkle-Twinkle Little Star di model pertama. Sekuens *pitch* dari *dataset* berjumlah 14 data: [0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58].

LAMPIRAN D
Perbandingan Hasil Estimasi *Pitch* dengan *Dataset*

Tabel D-1 Hasil Perbandingan *Pitch* pada lagu Twinkle-Twinkle Little Star di Model Pertama

No	<i>Pitch Result</i>	Dist	Match	Diff1	Diff2	Len_Match	Len_Diff1	Len_Diff2	Percentage
1	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58]	2	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58]	[]	[]	14	0	0	100.00%
2	[0, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57]	521	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57]	[58]	[44, 45]	13	1	2	90.48%
3	[0, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59]	550	[0, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58]	[46, 47]	[59]	12	2	1	88.10%
4	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55]	637	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55]	[56, 57, 58]	[]	11	3	0	85.71%
5	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56]	646	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56]	[57, 58]	[]	12	2	0	90.48%
6	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56]	739	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56]	[57, 58]	[]	12	2	0	90.48%
7	[0, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54]	860	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54]	[56, 57, 58, 55]	[44, 45]	10	4	2	76.19%
8	[0, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55]	999	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55]	[56, 57, 58]	[45]	11	3	1	83.33%
9	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57]	1055	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57]	[58]	[]	13	1	0	95.24%
10	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56]	1150	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56]	[57, 58]	[]	12	2	0	90.48%
11	[0, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55]	1222	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55]	[56, 57, 58]	[45]	11	3	1	83.33%
12	[0, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55]	1250	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55]	[56, 57, 58]	[43, 44, 45]	11	3	3	78.57%

LAMPIRAN D
Perbandingan Hasil Estimasi *Pitch* dengan *Dataset*

Tabel D-1 Hasil Perbandingan *Pitch* pada lagu Twinkle-Twinkle Little Star di Model Pertama

No	<i>Pitch Result</i>	Dist	Match	Diff1	Diff2	Len_Match	Len_Diff1	Len_Diff2	Percentage
13	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57]	1298	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57]	[58]	[]	13	1	0	95.24%
14	[0, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52]	1299	[0, 46, 47, 48, 49, 50, 51, 52]	[53, 54, 55, 56, 57, 58]	[42, 43, 44, 45]	8	6	4	61.90%
15	[0, 44, 45, 46, 48, 49, 50, 51, 52, 53, 54, 55, 56]	1370	[0, 46, 48, 49, 50, 51, 52, 53, 54, 55, 56]	[57, 58, 47]	[44, 45]	11	3	2	80.95%
16	[0, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55]	1450	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55]	[56, 57, 58]	[44, 45]	11	3	2	80.95%
17	[0, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 59, 60]	1557	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57]	[58, 53]	[59, 60, 45]	12	2	3	83.33%
18	[0, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 53]	1680	[0, 46, 47, 48, 49, 50, 51, 53]	[52, 54, 55, 56, 57, 58]	[40, 41, 42, 43, 44, 45]	8	6	6	57.14%
19	[0, 43, 44, 45, 46, 47, 48, 49, 50, 53, 54]	1707	[0, 46, 47, 48, 49, 50, 53, 54]	[51, 52, 55, 56, 57, 58]	[43, 44, 45]	8	6	3	64.29%
20	[0, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 56]	1817	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 56]	[57, 58, 55]	[41, 42, 43, 44, 45]	11	3	5	73.81%
21	[0, 48, 49, 50, 52, 53, 54, 55, 56, 57, 58, 59, 60]	1819	[0, 48, 49, 50, 52, 53, 54, 55, 56, 57, 58]	[51, 46, 47]	[59, 60]	11	3	2	80.95%
22	[0, 47, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63]	1847	[0, 47, 54, 55, 56, 57, 58]	[46, 48, 49, 50, 51, 52, 53]	[59, 60, 61, 62, 63]	7	7	5	54.76%
23	[0, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54]	1931	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54]	[56, 57, 58, 55]	[43, 44, 45]	10	4	3	73.81%

LAMPIRAN D

Perbandingan Hasil Estimasi *Pitch* dengan *Dataset*

Tabel D-1 Hasil Perbandingan *Pitch* pada lagu Twinkle-Twinkle Little Star di Model Pertama

No	<i>Pitch Result</i>	Dist	Match	Diff1	Diff2	Len_Match	Len_Diff1	Len_Diff2	Percentage
24	[0, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64]	2023	[0, 54, 55, 56, 57, 58]	[46, 47, 48, 49, 50, 51, 52, 53]	[64, 59, 60, 61, 62, 63]	6	8	6	47.62%
25	[0, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55]	2054	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55]	[56, 57, 58]	[44, 45]	11	3	2	80.95%
26	[0, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53]	2078	[0, 46, 47, 48, 49, 50, 51, 52, 53]	[54, 55, 56, 57, 58]	[42, 43, 44, 45]	9	5	4	66.67%
27	[0, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62]	2217	[0, 52, 53, 54, 55, 56, 57, 58]	[46, 47, 48, 49, 50, 51]	[59, 60, 61, 62]	8	6	4	61.90%
28	[0, 54, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66]	2343	[0, 57, 58, 54]	[46, 47, 48, 49, 50, 51, 52, 53, 55, 56]	[64, 65, 66, 59, 60, 61, 62, 63]	4	10	8	33.33%
29	[0, 45, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66]	2370	[0, 57, 56, 58]	[46, 47, 48, 49, 50, 51, 52, 53, 54, 55]	[64, 65, 66, 45, 59, 60, 61, 62, 63]	4	10	9	30.95%
30	[0, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68]	2502	[0, 57, 58]	[46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56]	[64, 65, 66, 67, 68, 59, 60, 61, 62, 63]	3	11	10	23.81%
31	[0, 45, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66]	2578	[0, 57, 56, 58]	[46, 47, 48, 49, 50, 51, 52, 53, 54, 55]	[64, 65, 66, 45, 59, 60, 61, 62, 63]	4	10	9	30.95%
32	[0, 42, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64]	2871	[0, 54, 55, 56, 57, 58]	[46, 47, 48, 49, 50, 51, 52, 53]	[64, 42, 59, 60, 61, 62, 63]	6	8	7	45.24%

Tabel D-2 menunjukkan perbandingan *pitch detected* dan *pitch* di *dataset* pada lagu Old McDonald Had a Farm di model pertama. Sekuens *pitch* dari *dataset* berjumlah 12 data: [0, 56, 57, 58, 59, 60, 62, 63, 64, 65, 66, 67].

LAMPIRAN D

Perbandingan Hasil Estimasi *Pitch* dengan *Dataset*

Tabel D-2 Hasil Perbandingan *Pitch* pada lagu Old McDonald Had a Farm di Model Pertama

No	<i>Pitch Result</i>	<i>Dist</i>	<i>Match</i>	<i>Diff1</i>	<i>Diff2</i>	<i>Len_Match</i>	<i>Len_Diff1</i>	<i>Len_Diff2</i>	<i>Percentage</i>
1	[0, 52, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67]	132	[0, 65, 66, 64, 67, 56, 57, 58, 59, 60, 62, 63]	[]	[52, 61]	12	0	2	94.44%
2	[0, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68]	645	[0, 65, 66, 64, 67, 56, 57, 58, 59, 60, 62, 63]	[]	[68, 61, 55]	12	0	3	91.67%
3	[0, 49, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64]	756	[0, 64, 56, 57, 58, 59, 60, 62, 63]	[65, 66, 67]	[49, 61, 55]	9	3	3	75.00%
4	[0, 53, 54, 55, 56, 57, 59, 60, 61, 62, 63]	817	[0, 56, 57, 59, 60, 62, 63]	[64, 65, 66, 67, 58]	[61, 53, 54, 55]	7	5	4	61.11%
5	[0, 42, 49, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66]	1068	[0, 65, 66, 64, 56, 57, 58, 59, 60, 62, 63]	[67]	[49, 42, 61]	11	1	3	86.11%
6	[0, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62]	1171	[0, 56, 57, 58, 59, 60, 62]	[64, 65, 66, 67, 63]	[51, 52, 53, 54, 55, 61]	7	5	6	55.56%
7	[0, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70]	1181	[0, 65, 66, 64, 67, 60, 62, 63]	[56, 57, 58, 59]	[61, 68, 69, 70]	8	4	4	66.67%
8	[0, 45, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65]	1200	[0, 65, 64, 56, 57, 58, 59, 60, 62, 63]	[66, 67]	[61, 45]	10	2	2	83.33%
9	[0, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64]	1245	[0, 64, 56, 57, 58, 59, 60, 62, 63]	[65, 66, 67]	[51, 52, 53, 54, 55, 61]	9	3	6	66.67%
10	[0, 48, 49, 50, 51, 52, 54, 55, 56, 57, 58, 59, 60]	1648	[0, 56, 57, 58, 59, 60]	[64, 65, 66, 67, 62, 63]	[48, 49, 50, 51, 52, 54, 55]	6	6	7	47.22%
11	[0, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59]	1668	[0, 56, 57, 58, 59]	[64, 65, 66, 67, 60, 62, 63]	[50, 51, 52, 53, 54, 55]	5	7	6	44.44%

LAMPIRAN D

Perbandingan Hasil Estimasi *Pitch* dengan *Dataset*

Tabel D-2 Hasil Perbandingan *Pitch* pada lagu Old McDonald Had a Farm di Model Pertama

No	<i>Pitch Result</i>	Dist	Match	Diff1	Diff2	Len_Match	Len_Diff1	Len_Diff2	Percentage
12	[0, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60]	1742	[0, 56, 57, 58, 59, 60]	[64, 65, 66, 67, 62, 63]	[47, 48, 49, 50, 51, 52, 53, 54, 55]	6	6	9	41.67%
13	[0, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58]	1836	[0, 57, 56, 58]	[64, 65, 66, 67, 59, 60, 62, 63]	[48, 49, 50, 51, 52, 53, 54, 55]	4	8	8	33.33%
14	[0, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58]	2151	[0, 57, 56, 58]	[64, 65, 66, 67, 59, 60, 62, 63]	[47, 48, 49, 50, 51, 52, 53, 54, 55]	4	8	9	30.56%
15	[0, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57]	2173	[0, 57, 56]	[64, 65, 66, 67, 58, 59, 60, 62, 63]	[45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55]	3	9	11	19.44%
16	[0, 45, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58]	2461	[0, 57, 56, 58]	[64, 65, 66, 67, 59, 60, 62, 63]	[45, 47, 48, 49, 50, 51, 52, 53, 54, 55]	4	8	10	27.78%
17	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56]	2466	[0, 56]	[64, 65, 66, 67, 57, 58, 59, 60, 62, 63]	[46, 47, 48, 49, 50, 51, 52, 53, 54, 55]	2	10	10	16.67%
18	[0, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52]	2716	[0]	[64, 65, 66, 67, 56, 57, 58, 59, 60, 62, 63]	[42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52]	1	11	11	8.33%
19	[0, 44, 45, 46, 47, 49, 50, 51, 52, 53, 54, 55]	2736	[0]	[64, 65, 66, 67, 56, 57, 58, 59, 60, 62, 63]	[44, 45, 46, 47, 49, 50, 51, 52, 53, 54, 55]	1	11	11	8.33%
20	[0, 42, 43, 44, 45, 47, 48, 49, 50, 51, 52, 55]	2808	[0]	[64, 65, 66, 67, 56, 57, 58, 59, 60, 62, 63]	[42, 43, 44, 45, 47, 48, 49, 50, 51, 52, 55]	1	11	11	8.33%
21	[0, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53]	2940	[0]	[64, 65, 66, 67, 56, 57, 58, 59, 60, 62, 63]	[44, 45, 46, 47, 48, 49, 50, 51, 52, 53]	1	11	10	11.11%
22	[0, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54]	2990	[0]	[64, 65, 66, 67, 56, 57, 58, 59, 60, 62, 63]	[41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54]	1	11	14	0.00%

LAMPIRAN D

Perbandingan Hasil Estimasi *Pitch* dengan *Dataset*

Tabel D-2 Hasil Perbandingan *Pitch* pada lagu Old McDonald Had a Farm di Model Pertama

No	<i>Pitch Result</i>	Dist	Match	<i>Diff1</i>	<i>Diff2</i>	<i>Len_Match</i>	<i>Len_Diff1</i>	<i>Len_Diff2</i>	Percentage
23	[0, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53]	2999	[0]	[64, 65, 66, 67, 56, 57, 58, 59, 60, 62, 63]	[42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53]	1	11	12	5.56%
24	[0, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53]	3082	[0]	[64, 65, 66, 67, 56, 57, 58, 59, 60, 62, 63]	[42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53]	1	11	12	5.56%
25	[0, 48, 49, 50, 51, 53, 54, 55, 56, 57, 58]	3101	[0, 57, 56, 58]	[64, 65, 66, 67, 59, 60, 62, 63]	[48, 49, 50, 51, 53, 54, 55]	4	8	7	36.11%
26	[0, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52]	3116	[0]	[64, 65, 66, 67, 56, 57, 58, 59, 60, 62, 63]	[42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52]	1	11	11	8.33%
27	[0, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 54]	3175	[0]	[64, 65, 66, 67, 56, 57, 58, 59, 60, 62, 63]	[41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 54]	1	11	13	2.78%
28	[0, 39, 43, 44, 45, 46, 47, 48, 49, 50, 52, 53, 54, 59]	3468	[0, 59]	[64, 65, 66, 67, 56, 57, 58, 60, 62, 63]	[39, 43, 44, 45, 46, 47, 48, 49, 50, 52, 53, 54]	2	10	12	11.11%
29	[0, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 60]	3486	[0, 60]	[64, 65, 66, 67, 56, 57, 58, 59, 62, 63]	[42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54]	2	10	13	8.33%
30	[0, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52]	3542	[0]	[64, 65, 66, 67, 56, 57, 58, 59, 60, 62, 63]	[42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52]	1	11	11	8.33%
31	[0, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51]	3796	[0]	[64, 65, 66, 67, 56, 57, 58, 59, 60, 62, 63]	[41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51]	1	11	11	8.33%

Tabel D-3 menunjukkan perbandingan *pitch detected* dan *pitch* di *dataset* pada lagu Happy Birthday di model pertama. Sekuens *pitch* dari *dataset* berjumlah 12 data: [0, 56, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67].

LAMPIRAN D

Perbandingan Hasil Estimasi *Pitch* dengan *Dataset*

Tabel D-3 Hasil Perbandingan *Pitch* pada lagu Happy Birthday di Model Pertama

No	<i>Pitch Result</i>	Dist	Match	Diff1	Diff2	Len_Match	Len_Diff1	Len_Diff2	Percentage
1	[0, 56, 57, 58, 59, 60, 61, 63, 64, 65, 66]	297	[0, 65, 66, 64, 56, 58, 59, 60, 61, 63]	[67, 62]	[57]	10	2	1	86.11%
2	[0, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67]	339	[0, 65, 66, 64, 67, 58, 59, 60, 61, 62, 63]	[56]	[]	11	1	0	94.44%
3	[0, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66]	805	[0, 65, 66, 64, 56, 58, 59, 60, 61, 62, 63]	[67]	[57, 53, 54, 55]	11	1	4	83.33%
4	[0, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67]	982	[0, 65, 66, 64, 67, 58, 59, 60, 61, 62, 63]	[56]	[57]	11	1	1	91.67%
5	[0, 54, 55, 56, 57, 59, 60, 61, 62, 64, 65, 67, 68]	1270	[0, 65, 64, 67, 56, 59, 60, 61, 62]	[66, 58, 63]	[57, 68, 54, 55]	9	3	4	72.22%
6	[0, 45, 50, 54, 55, 56, 57, 58, 59, 60, 61, 62, 64, 66, 67]	1366	[0, 64, 66, 67, 56, 58, 59, 60, 61, 62]	[65, 63]	[45, 50, 54, 55, 57]	10	2	5	75.00%
7	[0, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67]	1407	[0, 65, 66, 64, 67, 58, 59, 60, 61, 62, 63]	[56]	[57]	11	1	1	91.67%
8	[0, 58, 59, 60, 61, 62, 64, 65, 67, 68]	1450	[0, 65, 64, 67, 58, 59, 60, 61, 62]	[56, 66, 63]	[68]	9	3	1	80.56%
9	[0, 51, 53, 54, 55, 56, 57, 59, 60]	1524	[0, 56, 59, 60]	[64, 65, 66, 67, 58, 61, 62, 63]	[51, 53, 54, 55, 57]	4	8	5	41.67%
10	[0, 51, 52, 53, 54, 55, 56, 57, 58, 59, 61]	1780	[0, 56, 58, 59, 61]	[64, 65, 66, 67, 60, 62, 63]	[51, 52, 53, 54, 55, 57]	5	7	6	44.44%
11	[0, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57]	2275	[0, 56]	[64, 65, 66, 67, 58, 59, 60, 61, 62, 63]	[48, 49, 50, 51, 52, 53, 54, 55, 57]	2	10	9	19.44%
12	[0, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60]	2435	[0, 56, 58, 59, 60]	[64, 65, 66, 67, 61, 62, 63]	[45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 57]	5	7	12	27.78%

LAMPIRAN D

Perbandingan Hasil Estimasi *Pitch* dengan *Dataset*

Tabel D-3 Hasil Perbandingan *Pitch* pada lagu Happy Birthday di Model Pertama

No	<i>Pitch Result</i>	Dist	Match	Diff1	Diff2	Len_Match	Len_Diff1	Len_Diff2	Percentage
13	[0, 45, 46, 47, 48, 49, 51, 52, 53, 54, 56, 57, 58, 59]	2473	[0, 56, 59, 58]	[64, 65, 66, 67, 60, 61, 62, 63]	[45, 46, 47, 48, 49, 51, 52, 53, 54, 57]	4	8	10	27.78%
14	[0, 42, 48, 49, 50, 51, 52, 54, 55, 56, 61]	2479	[0, 56, 61]	[64, 65, 66, 67, 58, 59, 60, 62, 63]	[42, 48, 49, 50, 51, 52, 54, 55]	3	9	8	27.78%
15	[0, 45, 46, 47, 48, 50, 51, 52, 53, 54]	2641	[0]	[64, 65, 66, 67, 56, 58, 59, 60, 61, 62, 63]	[45, 46, 47, 48, 50, 51, 52, 53, 54]	1	11	9	13.89%
16	[0, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58]	2662	[0, 56, 58]	[64, 65, 66, 67, 59, 60, 61, 62, 63]	[45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 57]	3	9	12	16.67%
17	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54]	2722	[0]	[64, 65, 66, 67, 56, 58, 59, 60, 61, 62, 63]	[46, 47, 48, 49, 50, 51, 52, 53, 54]	1	11	9	13.89%
18	[0, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 56, 57, 58]	2841	[0, 56, 58]	[64, 65, 66, 67, 59, 60, 61, 62, 63]	[44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 57]	3	9	12	16.67%
19	[0, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58]	2847	[0, 56, 58]	[64, 65, 66, 67, 59, 60, 61, 62, 63]	[43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 57]	3	9	14	11.11%
20	[0, 46, 47, 48, 49, 50, 52, 53, 54, 55]	2883	[0]	[64, 65, 66, 67, 56, 58, 59, 60, 61, 62, 63]	[46, 47, 48, 49, 50, 52, 53, 54, 55]	1	11	9	13.89%
21	[0, 44, 45, 46, 47, 48, 50, 51, 52, 53, 54, 55]	2903	[0]	[64, 65, 66, 67, 56, 58, 59, 60, 61, 62, 63]	[44, 45, 46, 47, 48, 50, 51, 52, 53, 54, 55]	1	11	11	8.33%
22	[0, 45, 46, 47, 48, 49, 50, 52, 53, 54, 55]	2931	[0]	[64, 65, 66, 67, 56, 58, 59, 60, 61, 62, 63]	[45, 46, 47, 48, 49, 50, 52, 53, 54, 55]	1	11	10	11.11%
23	[0, 44, 45, 46, 47, 48, 49, 50, 51, 52, 54, 55, 56, 57, 58]	2967	[0, 56, 58]	[64, 65, 66, 67, 59, 60, 61, 62, 63]	[44, 45, 46, 47, 48, 49, 50, 51, 52, 54, 55, 57]	3	9	12	16.67%

LAMPIRAN D

Perbandingan Hasil Estimasi *Pitch* dengan *Dataset*

Tabel D-3 Hasil Perbandingan *Pitch* pada lagu Happy Birthday di Model Pertama

No	<i>Pitch Result</i>	Dist	Match	<i>Diff1</i>	<i>Diff2</i>	<i>Len_Match</i>	<i>Len_Diff1</i>	<i>Len_Diff2</i>	Percentage
24	[0, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55]	2970	[0]	[64, 65, 66, 67, 56, 58, 59, 60, 61, 62, 63]	[44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55]	1	11	12	5.56%
25	[0, 44, 45, 46, 47, 49, 50, 51, 52, 53]	3032	[0]	[64, 65, 66, 67, 56, 58, 59, 60, 61, 62, 63]	[44, 45, 46, 47, 49, 50, 51, 52, 53]	1	11	9	13.89%
26	[0, 44, 45, 46, 47, 49, 50, 51, 52, 54, 55, 56, 57, 58]	3064	[0, 56, 58]	[64, 65, 66, 67, 59, 60, 61, 62, 63]	[44, 45, 46, 47, 49, 50, 51, 52, 54, 55, 57]	3	9	11	19.44%
27	[0, 45, 46, 47, 48, 49, 51, 52, 53, 54, 55, 56, 58, 59]	3235	[0, 56, 59, 58]	[64, 65, 66, 67, 60, 61, 62, 63]	[45, 46, 47, 48, 49, 51, 52, 53, 54, 55]	4	8	10	27.78%
28	[0, 45, 46, 47, 48, 49, 50, 51, 52, 53]	3283	[0]	[64, 65, 66, 67, 56, 58, 59, 60, 61, 62, 63]	[45, 46, 47, 48, 49, 50, 51, 52, 53]	1	11	9	13.89%
29	[0, 42, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56]	3290	[0, 56]	[64, 65, 66, 67, 58, 59, 60, 61, 62, 63]	[42, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55]	2	10	13	8.33%
30	[0, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54]	3322	[0]	[64, 65, 66, 67, 56, 58, 59, 60, 61, 62, 63]	[44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54]	1	11	11	8.33%
31	[0, 43, 44, 45, 46, 48, 49, 50, 51, 54, 55, 56]	3411	[0, 56]	[64, 65, 66, 67, 58, 59, 60, 61, 62, 63]	[43, 44, 45, 46, 48, 49, 50, 51, 54, 55]	2	10	10	16.67%
32	[0, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54]	3541	[0]	[64, 65, 66, 67, 56, 58, 59, 60, 61, 62, 63]	[44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54]	1	11	11	8.33%
33	[0, 41, 42, 43, 44, 45, 46, 47, 48, 49, 51, 53, 54, 55]	3787	[0]	[64, 65, 66, 67, 56, 58, 59, 60, 61, 62, 63]	[41, 42, 43, 44, 45, 46, 47, 48, 49, 51, 53, 54, 55]	1	11	13	2.78%
34	[0, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52]	3856	[0]	[64, 65, 66, 67, 56, 58, 59, 60, 61, 62, 63]	[43, 44, 45, 46, 47, 48, 49, 50, 51, 52]	1	11	10	11.11%

Tabel D-4 menunjukkan perbandingan *pitch detected* dan *pitch* di *dataset* pada lagu Brother John di model pertama. Sekuens *pitch* dari *dataset* berjumlah 13

LAMPIRAN D

Perbandingan Hasil Estimasi *Pitch* dengan *Dataset*

data: [0, 47, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68].

Tabel D-4 Hasil Perbandingan *Pitch* pada lagu Brother John di Model Pertama

No	Pitch Result	Dist	Match	Diff1	Diff2	Len_Match	Len_Diff1	Len_Diff2	Percentage
1	[0, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68]	185	[0, 65, 66, 64, 67, 68, 58, 59, 60, 61, 62, 63]	[47]	[]	12	1	0	94.87%
2	[0, 54, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67]	510	[0, 65, 66, 64, 67, 58, 59, 60, 61, 62, 63]	[68, 47]	[56, 57, 54]	11	2	3	82.05%
3	[0, 47, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67]	519	[0, 65, 66, 64, 67, 47, 58, 59, 60, 61, 62, 63]	[68]	[56, 57]	12	1	2	89.74%
4	[0, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69]	534	[0, 65, 66, 64, 67, 68, 59, 60, 61, 62, 63]	[58, 47]	[69]	11	2	1	87.18%
5	[0, 44, 56, 58, 59, 61, 62, 63, 64, 66, 67, 69]	669	[0, 64, 66, 67, 58, 59, 61, 62, 63]	[65, 68, 60, 47]	[56, 44, 69]	9	4	3	71.79%
6	[0, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70]	705	[0, 65, 66, 64, 67, 68, 59, 60, 61, 62, 63]	[58, 47]	[69, 70]	11	2	2	84.62%
7	[0, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67]	1076	[0, 65, 66, 64, 67, 58, 59, 60, 61, 62, 63]	[68, 47]	[56, 57]	11	2	2	84.62%
8	[0, 47, 48, 49, 51, 52, 54, 55, 56, 57, 58, 59, 60, 61, 64]	1657	[0, 64, 47, 58, 59, 60, 61]	[65, 66, 67, 68, 62, 63]	[48, 49, 51, 52, 54, 55, 56, 57]	7	6	8	48.72%
9	[0, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 60, 61]	1770	[0, 47, 58, 60, 61]	[64, 65, 66, 67, 68, 59, 62, 63]	[43, 44, 45, 46, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57]	5	8	14	23.08%
10	[0, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57]	1986	[0, 47]	[64, 65, 66, 67, 68, 58, 59, 60, 61, 62, 63]	[45, 46, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57]	2	11	12	12.82%

LAMPIRAN D

Perbandingan Hasil Estimasi *Pitch* dengan *Dataset*

Tabel D-4 Hasil Perbandingan *Pitch* pada lagu Brother John di Model Pertama

No	Pitch Result	Dist	Match	Diff1	Diff2	Len_Match	Len_Diff1	Len_Diff2	Percentage
11	[0, 48, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59]	2008	[0, 58, 59]	[64, 65, 66, 67, 68, 47, 60, 61, 62, 63]	[48, 50, 51, 52, 53, 54, 55, 56, 57]	3	10	9	25.64%
12	[0, 45, 46, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59]	2031	[0, 58, 59]	[64, 65, 66, 67, 68, 47, 60, 61, 62, 63]	[45, 46, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57]	3	10	12	17.95%
13	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58]	2149	[0, 58, 47]	[64, 65, 66, 67, 68, 59, 60, 61, 62, 63]	[46, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57]	3	10	11	20.51%
14	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58]	2238	[0, 58, 47]	[64, 65, 66, 67, 68, 59, 60, 61, 62, 63]	[46, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57]	3	10	11	20.51%
15	[0, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 61, 62]	2327	[0, 47, 58, 61, 62]	[64, 65, 66, 67, 68, 59, 60, 63]	[48, 49, 50, 51, 52, 53, 54, 55, 56, 57]	5	8	10	33.33%
16	[0, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57]	2414	[0, 47]	[64, 65, 66, 67, 68, 58, 59, 60, 61, 62, 63]	[43, 44, 45, 46, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57]	2	11	14	7.69%
17	[0, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58]	2415	[0, 58, 47]	[64, 65, 66, 67, 68, 59, 60, 61, 62, 63]	[48, 49, 50, 51, 52, 53, 54, 55, 56, 57]	3	10	10	23.08%
18	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57]	2431	[0, 47]	[64, 65, 66, 67, 68, 58, 59, 60, 61, 62, 63]	[46, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57]	2	11	11	15.38%
19	[0, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 61]	2457	[0, 61, 47]	[64, 65, 66, 67, 68, 58, 59, 60, 62, 63]	[45, 46, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57]	3	10	12	17.95%
20	[0, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69]	2493	[0, 65, 66, 64, 67, 68, 59, 60, 61, 62, 63]	[58, 47]	[69]	11	2	1	87.18%
21	[0, 47, 48, 49, 50, 51, 52, 53, 54, 55, 57, 59]	2520	[0, 59, 47]	[64, 65, 66, 67, 68, 58, 60, 61, 62, 63]	[48, 49, 50, 51, 52, 53, 54, 55, 57]	3	10	9	25.64%

LAMPIRAN D

Perbandingan Hasil Estimasi *Pitch* dengan *Dataset*

Tabel D-4 Hasil Perbandingan *Pitch* pada lagu Brother John di Model Pertama

No	Pitch Result	Dist	Match	Diff1	Diff2	Len_Match	Len_Diff1	Len_Diff2	Percentage
22	[0, 49, 50, 51, 52, 53, 54, 55, 56, 57]	2590	[0]	[64, 65, 66, 67, 68, 47, 58, 59, 60, 61, 62, 63]	[49, 50, 51, 52, 53, 54, 55, 56, 57]	1	12	9	15.38%
23	[0, 45, 46, 47, 49, 50, 51, 52, 53, 54, 55]	2651	[0, 47]	[64, 65, 66, 67, 68, 58, 59, 60, 61, 62, 63]	[45, 46, 49, 50, 51, 52, 53, 54, 55]	2	11	9	20.51%
24	[0, 44, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 58, 61]	2664	[0, 58, 61, 47]	[64, 65, 66, 67, 68, 59, 60, 62, 63]	[44, 48, 49, 50, 51, 52, 53, 54, 55, 56]	4	9	10	28.21%
25	[0, 44, 45, 46, 47, 48, 49, 50, 51]	2697	[0, 47]	[64, 65, 66, 67, 68, 58, 59, 60, 61, 62, 63]	[44, 45, 46, 48, 49, 50, 51]	2	11	7	25.64%
26	[0, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58]	2702	[0, 58, 47]	[64, 65, 66, 67, 68, 59, 60, 61, 62, 63]	[45, 46, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57]	3	10	12	17.95%
27	[0, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55]	2706	[0, 47]	[64, 65, 66, 67, 68, 58, 59, 60, 61, 62, 63]	[44, 45, 46, 48, 49, 50, 51, 52, 53, 54, 55]	2	11	11	15.38%
28	[0, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53]	2767	[0, 47]	[64, 65, 66, 67, 68, 58, 59, 60, 61, 62, 63]	[43, 44, 45, 46, 48, 49, 50, 51, 52, 53]	2	11	10	17.95%
29	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56]	2829	[0, 47]	[64, 65, 66, 67, 68, 58, 59, 60, 61, 62, 63]	[46, 48, 49, 50, 51, 52, 53, 54, 55, 56]	2	11	10	17.95%
30	[0, 42, 43, 44, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56]	2859	[0, 47]	[64, 65, 66, 67, 68, 58, 59, 60, 61, 62, 63]	[42, 43, 44, 46, 48, 49, 50, 51, 52, 53, 54, 55, 56]	2	11	13	10.26%
31	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56]	2921	[0, 47]	[64, 65, 66, 67, 68, 58, 59, 60, 61, 62, 63]	[46, 48, 49, 50, 51, 52, 53, 54, 55, 56]	2	11	10	17.95%
32	[0, 43, 44, 46, 47, 48, 49, 50, 51, 52, 53]	2981	[0, 47]	[64, 65, 66, 67, 68, 58, 59, 60, 61, 62, 63]	[43, 44, 46, 48, 49, 50, 51, 52, 53]	2	11	9	20.51%
33	[0, 44, 45, 46, 47, 48, 49, 50, 51, 52, 54]	3217	[0, 47]	[64, 65, 66, 67, 68, 58, 59, 60, 61, 62, 63]	[44, 45, 46, 48, 49, 50, 51, 52, 54]	2	11	9	20.51%

LAMPIRAN D

Perbandingan Hasil Estimasi *Pitch* dengan *Dataset*

Tabel D-4 Hasil Perbandingan *Pitch* pada lagu Brother John di Model Pertama

No	<i>Pitch Result</i>	<i>Dist</i>	<i>Match</i>	<i>Diff1</i>	<i>Diff2</i>	<i>Len_Match</i>	<i>Len_Diff1</i>	<i>Len_Diff2</i>	<i>Percentage</i>
34	[0, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51]	3562	[0, 47]	[64, 65, 66, 67, 68, 58, 59, 60, 61, 62, 63]	[41, 42, 43, 44, 45, 46, 48, 49, 50, 51]	2	11	10	17.95%
35	[0, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57]	3732	[0, 47]	[64, 65, 66, 67, 68, 58, 59, 60, 61, 62, 63]	[43, 44, 45, 46, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57]	2	11	14	7.69%

Tabel D-5 menunjukkan perbandingan *pitch detected* dan *pitch* di *dataset* pada lagu London Bridge is Falling Down di model pertama. Sekuens *pitch* dari *dataset* berjumlah 10 data: [0, 56, 57, 58, 59, 60, 61, 62, 63, 64].

Tabel D-5 Hasil Perbandingan *Pitch* pada lagu London Bridge is Falling Down di Model Pertama

No	<i>Pitch Result</i>	<i>Dist</i>	<i>Match</i>	<i>Diff1</i>	<i>Diff2</i>	<i>Len_Match</i>	<i>Len_Diff1</i>	<i>Len_Diff2</i>	<i>Percentage</i>
1	[0, 56, 58, 59, 60, 61, 62, 64]	10	[0, 64, 56, 58, 59, 60, 61, 62]	[57, 63]	[]	8	2	0	86.67%
2	[0, 55, 56, 58, 59, 60, 61, 63]	389	[0, 56, 58, 59, 60, 61, 63]	[64, 57, 62]	[55]	7	3	1	76.67%
3	[0, 49, 57, 58, 59, 60, 61, 62, 63, 64]	435	[0, 64, 57, 58, 59, 60, 61, 62, 63]	[56]	[49]	9	1	1	90.00%
4	[0, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64]	842	[0, 64, 56, 57, 58, 59, 60, 61, 62, 63]	[]	[55]	10	0	1	96.67%
5	[0, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62]	1068	[0, 56, 57, 58, 59, 60, 61, 62]	[64, 63]	[53, 54, 55]	8	2	3	76.67%
6	[0, 53, 54, 55, 56, 57, 58, 59, 60, 61]	1156	[0, 56, 57, 58, 59, 60, 61]	[64, 62, 63]	[53, 54, 55]	7	3	3	70.00%
7	[0, 45, 53, 54, 55, 56, 57, 58, 59, 60]	1195	[0, 56, 57, 58, 59, 60]	[64, 61, 62, 63]	[53, 45, 54, 55]	6	4	4	60.00%
8	[0, 50, 51, 52, 53, 54, 55, 56, 57, 58]	1287	[0, 57, 56, 58]	[64, 59, 60, 61, 62, 63]	[50, 51, 52, 53, 54, 55]	4	6	6	40.00%

LAMPIRAN D

Perbandingan Hasil Estimasi *Pitch* dengan *Dataset*

Tabel D-5 Hasil Perbandingan *Pitch* pada lagu London Bridge is Falling Down di Model Pertama

No	<i>Pitch Result</i>	<i>Dist</i>	<i>Match</i>	<i>Diff1</i>	<i>Diff2</i>	<i>Len_Match</i>	<i>Len_Diff1</i>	<i>Len_Diff2</i>	<i>Percentage</i>
9	[0, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60]	1427	[0, 56, 57, 58, 59, 60]	[64, 61, 62, 63]	[51, 52, 53, 54, 55]	6	4	5	56.67%
10	[0, 50, 51, 52, 53, 54, 55, 56, 57, 58]	1503	[0, 57, 56, 58]	[64, 59, 60, 61, 62, 63]	[50, 51, 52, 53, 54, 55]	4	6	6	40.00%
11	[0, 50, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61]	1615	[0, 56, 57, 58, 59, 60, 61]	[64, 62, 63]	[50, 52, 53, 54, 55]	7	3	5	63.33%
12	[0, 42, 43, 49, 50, 51, 52, 53, 54, 55, 56]	1616	[0, 56]	[64, 57, 58, 59, 60, 61, 62, 63]	[42, 43, 49, 50, 51, 52, 53, 54, 55]	2	8	9	16.67%
13	[0, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58]	1811	[0, 57, 56, 58]	[64, 59, 60, 61, 62, 63]	[48, 49, 50, 51, 52, 53, 54, 55]	4	6	8	33.33%
14	[0, 48, 49, 51, 52, 53, 54, 55, 56, 57]	1826	[0, 57, 56]	[64, 58, 59, 60, 61, 62, 63]	[48, 49, 51, 52, 53, 54, 55]	3	7	7	30.00%
15	[0, 48, 49, 50, 51, 52, 53, 54, 55, 56]	1916	[0, 56]	[64, 57, 58, 59, 60, 61, 62, 63]	[48, 49, 50, 51, 52, 53, 54, 55]	2	8	8	20.00%
16	[0, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57]	1980	[0, 57, 56]	[64, 58, 59, 60, 61, 62, 63]	[47, 48, 49, 50, 51, 52, 53, 54, 55]	3	7	9	23.33%
17	[0, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 58]	2009	[0, 58]	[64, 56, 57, 59, 60, 61, 62, 63]	[45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55]	2	8	11	10.00%
18	[0, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57]	2080	[0, 57, 56]	[64, 58, 59, 60, 61, 62, 63]	[48, 49, 50, 51, 52, 53, 54, 55]	3	7	8	26.67%
19	[0, 44, 45, 46, 48, 49, 50, 51, 52, 53]	2093	[0]	[64, 56, 57, 58, 59, 60, 61, 62, 63]	[44, 45, 46, 48, 49, 50, 51, 52, 53]	1	9	9	10.00%
20	[0, 45, 46, 48, 49, 50, 51, 52, 53, 54]	2173	[0]	[64, 56, 57, 58, 59, 60, 61, 62, 63]	[45, 46, 48, 49, 50, 51, 52, 53, 54]	1	9	9	10.00%
21	[0, 36, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53]	2226	[0]	[64, 56, 57, 58, 59, 60, 61, 62, 63]	[36, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53]	1	9	11	3.33%

LAMPIRAN D

Perbandingan Hasil Estimasi *Pitch* dengan *Dataset*

Tabel D-5 Hasil Perbandingan *Pitch* pada lagu London Bridge is Falling Down di Model Pertama

No	<i>Pitch Result</i>	<i>Dist</i>	<i>Match</i>	<i>Diff1</i>	<i>Diff2</i>	<i>Len_Match</i>	<i>Len_Diff1</i>	<i>Len_Diff2</i>	<i>Percentage</i>
22	[0, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58]	2237	[0, 57, 56, 58]	[64, 59, 60, 61, 62, 63]	[48, 49, 50, 51, 52, 53, 54, 55]	4	6	8	33.33%
23	[0, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64]	2273	[0, 64, 56, 57, 58, 59, 60, 61, 62, 63]	[]	[55]	10	0	1	96.67%
24	[0, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55]	2359	[0]	[64, 56, 57, 58, 59, 60, 61, 62, 63]	[45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55]	1	9	11	3.33%
25	[0, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53]	2573	[0]	[64, 56, 57, 58, 59, 60, 61, 62, 63]	[44, 45, 46, 47, 48, 49, 50, 51, 52, 53]	1	9	10	6.67%
26	[0, 45, 46, 47, 48, 49, 50, 51, 52, 53]	2647	[0]	[64, 56, 57, 58, 59, 60, 61, 62, 63]	[45, 46, 47, 48, 49, 50, 51, 52, 53]	1	9	9	10.00%
27	[0, 45, 46, 47, 48, 49, 50, 51, 52, 53]	2765	[0]	[64, 56, 57, 58, 59, 60, 61, 62, 63]	[45, 46, 47, 48, 49, 50, 51, 52, 53]	1	9	9	10.00%
28	[0, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56]	3490	[0, 56]	[64, 57, 58, 59, 60, 61, 62, 63]	[47, 48, 49, 50, 51, 52, 53, 54, 55]	2	8	9	16.67%
29	[0, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56]	3804	[0, 56]	[64, 57, 58, 59, 60, 61, 62, 63]	[45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55]	2	8	11	10.00%

Tabel D-6 menunjukkan perbandingan *pitch detected* dan *pitch* di *dataset* pada lagu Twinkle-Twinkle Little Star di model kedua. Sekuens *pitch* dari *dataset* berjumlah 14 data: [0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58].

Tabel D-6 Hasil Perbandingan *Pitch* pada lagu Twinkle-Twinkle Little Star di Model Kedua

No	<i>Pitch Result</i>	<i>Dist</i>	<i>Match</i>	<i>Diff1</i>	<i>Diff2</i>	<i>Len_Match</i>	<i>Len_Diff1</i>	<i>Len_Diff2</i>	<i>Percentage</i>
1	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57]	112	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57]	[58]	[]	13	1	0	95.24%

LAMPIRAN D
Perbandingan Hasil Estimasi *Pitch* dengan *Dataset*

Tabel D-6 Hasil Perbandingan *Pitch* pada lagu Twinkle-Twinkle Little Star di Model Kedua

No	<i>Pitch Result</i>	Dist	Match	Diff1	Diff2	Len_Match	Len_Diff1	Len_Diff2	Percentage
2	[0, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56]	489	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56]	[57, 58]	[44, 45]	12	2	2	85.71%
3	[0, 48, 49, 50, 52, 53, 54, 55, 56, 57, 58, 59]	570	[0, 48, 49, 50, 52, 53, 54, 55, 56, 57, 58]	[51, 46, 47]	[59]	11	3	1	83.33%
4	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55]	676	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55]	[56, 57, 58]	[]	11	3	0	85.71%
5	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56]	726	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56]	[57, 58]	[]	12	2	0	90.48%
6	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56]	732	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56]	[57, 58]	[]	12	2	0	90.48%
7	[0, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54]	1033	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54]	[56, 57, 58, 55]	[44, 45]	10	4	2	76.19%
8	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57]	1037	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57]	[58]	[]	13	1	0	95.24%
9	[0, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55]	1040	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55]	[56, 57, 58]	[45]	11	3	1	83.33%
10	[0, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54]	1070	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54]	[56, 57, 58, 55]	[43, 44, 45]	10	4	3	73.81%
11	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57]	1158	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57]	[58]	[]	13	1	0	95.24%
12	[0, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55]	1180	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55]	[56, 57, 58]	[45]	11	3	1	83.33%
13	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56]	1214	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56]	[57, 58]	[]	12	2	0	90.48%

LAMPIRAN D

Perbandingan Hasil Estimasi *Pitch* dengan *Dataset*

Tabel D-6 Hasil Perbandingan *Pitch* pada lagu Twinkle-Twinkle Little Star di Model Kedua

No	<i>Pitch Result</i>	Dist	Match	Diff1	Diff2	Len_Match	Len_Diff1	Len_Diff2	Percentage
14	[0, 44, 45, 46, 47, 48, 49, 50, 51, 52, 54, 55, 56, 57, 58, 59]	1249	[0, 46, 47, 48, 49, 50, 51, 52, 54, 55, 56, 57, 58]	[53]	[59, 44, 45]	13	1	3	88.10%
15	[0, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52]	1283	[0, 46, 47, 48, 49, 50, 51, 52]	[53, 54, 55, 56, 57, 58]	[42, 43, 44, 45]	8	6	4	61.90%
16	[0, 45, 46, 48, 49, 50, 51, 52, 53, 54, 55, 56]	1459	[0, 46, 48, 49, 50, 51, 52, 53, 54, 55, 56]	[57, 58, 47]	[45]	11	3	1	83.33%
17	[0, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54]	1474	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54]	[56, 57, 58, 55]	[44, 45]	10	4	2	76.19%
18	[0, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52]	1502	[0, 46, 47, 48, 49, 50, 51, 52]	[53, 54, 55, 56, 57, 58]	[40, 41, 42, 43, 44, 45]	8	6	6	57.14%
19	[0, 48, 49, 50, 52, 53, 54, 55, 56, 57, 58, 59]	1742	[0, 48, 49, 50, 52, 53, 54, 55, 56, 57, 58]	[51, 46, 47]	[59]	11	3	1	83.33%
20	[0, 43, 44, 45, 46, 47, 48, 49, 50]	1764	[0, 46, 47, 48, 49, 50]	[51, 52, 53, 54, 55, 56, 57, 58]	[43, 44, 45]	6	8	3	54.76%
21	[0, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54]	1791	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54]	[56, 57, 58, 55]	[41, 42, 43, 44, 45]	10	4	5	69.05%
22	[0, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63]	1804	[0, 54, 55, 56, 57, 58]	[46, 47, 48, 49, 50, 51, 52, 53]	[59, 60, 61, 62, 63]	6	8	5	50.00%
23	[0, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64]	2028	[0, 54, 55, 56, 57, 58]	[46, 47, 48, 49, 50, 51, 52, 53]	[64, 59, 60, 61, 62, 63]	6	8	6	47.62%
24	[0, 44, 45, 46, 47, 48, 49, 50, 51, 52, 54]	2080	[0, 46, 47, 48, 49, 50, 51, 52, 54]	[53, 55, 56, 57, 58]	[44, 45]	9	5	2	71.43%
25	[0, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62]	2118	[0, 52, 53, 54, 55, 56, 57, 58]	[46, 47, 48, 49, 50, 51]	[59, 60, 61, 62]	8	6	4	61.90%

LAMPIRAN D

Perbandingan Hasil Estimasi *Pitch* dengan *Dataset*

Tabel D-6 Hasil Perbandingan *Pitch* pada lagu Twinkle-Twinkle Little Star di Model Kedua

No	<i>Pitch Result</i>	<i>Dist</i>	<i>Match</i>	<i>Diff1</i>	<i>Diff2</i>	<i>Len_Match</i>	<i>Len_Diff1</i>	<i>Len_Diff2</i>	<i>Percentage</i>
26	[0, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54]	2130	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54]	[56, 57, 58, 55]	[44, 45]	10	4	2	76.19%
27	[0, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53]	2147	[0, 46, 47, 48, 49, 50, 51, 52, 53]	[54, 55, 56, 57, 58]	[42, 43, 44, 45]	9	5	4	66.67%
28	[0, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66]	2399	[0, 57, 58]	[46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56]	[64, 65, 66, 59, 60, 61, 62, 63]	3	11	8	28.57%
29	[0, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68]	2429	[0, 57, 58]	[46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56]	[64, 65, 66, 67, 68, 59, 60, 61, 62, 63]	3	11	10	23.81%
30	[0, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66]	2636	[0, 57, 56, 58]	[46, 47, 48, 49, 50, 51, 52, 53, 54, 55]	[64, 65, 66, 59, 60, 61, 62, 63]	4	10	8	33.33%
31	[0, 57, 58, 59, 60, 62, 63, 64, 65, 66]	2695	[0, 57, 58]	[46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56]	[64, 65, 66, 59, 60, 62, 63]	3	11	7	30.95%
32	[0, 54, 56, 57, 58, 59, 60, 62, 64]	2975	[0, 54, 56, 57, 58]	[46, 47, 48, 49, 50, 51, 52, 53, 55]	[64, 59, 60, 62]	5	9	4	47.62%

Tabel D-7 menunjukkan perbandingan *pitch detected* dan *pitch* di *dataset* pada lagu Old McDonald Had a Farm di model kedua. Sekuens *pitch* dari *dataset* berjumlah 12 data: [0, 56, 57, 58, 59, 60, 62, 63, 64, 65, 66, 67].

Tabel D-7 Hasil Perbandingan *Pitch* pada lagu Old McDonald Had a Farm di Model Kedua

No	<i>Pitch Result</i>	<i>Dist</i>	<i>Match</i>	<i>Diff1</i>	<i>Diff2</i>	<i>Len_Match</i>	<i>Len_Diff1</i>	<i>Len_Diff2</i>	<i>Percentage</i>
1	[0, 52, 56, 57, 58, 59, 62, 63, 64, 65, 66, 67]	133	[0, 65, 66, 64, 67, 56, 57, 58, 59, 62, 63]	[60]	[52]	11	1	1	91.67%
2	[0, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68]	688	[0, 65, 66, 64, 67, 56, 57, 58, 59, 60, 62, 63]	[]	[68, 61]	12	0	2	94.44%

LAMPIRAN D

Perbandingan Hasil Estimasi *Pitch* dengan *Dataset*

Tabel D-7 Hasil Perbandingan *Pitch* pada lagu Old McDonald Had a Farm di Model Kedua

No	<i>Pitch Result</i>	Dist	Match	Diff1	Diff2	Len_Match	Len_Diff1	Len_Diff2	Percentage
3	[0, 49, 56, 57, 58, 59, 61, 62, 63, 64, 65, 66]	744	[0, 65, 66, 64, 56, 57, 58, 59, 62, 63]	[67, 60]	[49, 61]	10	2	2	83.33%
4	[0, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63]	813	[0, 56, 57, 58, 59, 60, 62, 63]	[64, 65, 66, 67]	[61, 53, 54, 55]	8	4	4	66.67%
5	[0, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64]	871	[0, 64, 56, 57, 58, 59, 60, 62, 63]	[65, 66, 67]	[61, 55]	9	3	2	77.78%
6	[0, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65]	926	[0, 65, 64, 56, 57, 58, 59, 60, 62, 63]	[66, 67]	[61]	10	2	1	86.11%
7	[0, 59, 60, 61, 62, 63, 64, 66, 67, 68, 69, 70]	1044	[0, 64, 66, 67, 59, 60, 62, 63]	[56, 65, 58, 57]	[61, 68, 69, 70]	8	4	4	66.67%
8	[0, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64]	1100	[0, 64, 56, 57, 58, 59, 60, 62, 63]	[65, 66, 67]	[51, 52, 53, 54, 55, 61]	9	3	6	66.67%
9	[0, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62]	1211	[0, 56, 57, 58, 59, 60, 62]	[64, 65, 66, 67, 63]	[51, 52, 53, 54, 55, 61]	7	5	6	55.56%
10	[0, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59]	1579	[0, 56, 57, 58, 59]	[64, 65, 66, 67, 60, 62, 63]	[50, 51, 52, 53, 54, 55]	5	7	6	44.44%
11	[0, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60]	1703	[0, 56, 57, 58, 59, 60]	[64, 65, 66, 67, 62, 63]	[47, 48, 49, 50, 51, 52, 53, 54, 55]	6	6	9	41.67%
12	[0, 48, 49, 50, 51, 52, 54, 55, 56, 57, 58, 59, 60]	1817	[0, 56, 57, 58, 59, 60]	[64, 65, 66, 67, 62, 63]	[48, 49, 50, 51, 52, 54, 55]	6	6	7	47.22%
13	[0, 48, 49, 50, 53, 54, 55, 56, 57, 58]	1953	[0, 57, 56, 58]	[64, 65, 66, 67, 59, 60, 62, 63]	[48, 49, 50, 53, 54, 55]	4	8	6	38.89%
14	[0, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58]	2014	[0, 57, 56, 58]	[64, 65, 66, 67, 59, 60, 62, 63]	[47, 48, 49, 50, 51, 52, 53, 54, 55]	4	8	9	30.56%

LAMPIRAN D

Perbandingan Hasil Estimasi *Pitch* dengan *Dataset*

Tabel D-7 Hasil Perbandingan *Pitch* pada lagu Old McDonald Had a Farm di Model Kedua

No	Pitch Result	Dist	Match	Diff1	Diff2	Len_Match	Len_Diff1	Len_Diff2	Percentage
15	[0, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58]	2299	[0, 57, 56, 58]	[64, 65, 66, 67, 59, 60, 62, 63]	[47, 48, 49, 50, 51, 52, 53, 54, 55]	4	8	9	30.56%
16	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56]	2361	[0, 56]	[64, 65, 66, 67, 57, 58, 59, 60, 62, 63]	[46, 47, 48, 49, 50, 51, 52, 53, 54, 55]	2	10	10	16.67%
17	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56]	2593	[0, 56]	[64, 65, 66, 67, 57, 58, 59, 60, 62, 63]	[46, 47, 48, 49, 50, 51, 52, 53, 54, 55]	2	10	10	16.67%
18	[0, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53]	2662	[0]	[64, 65, 66, 67, 56, 57, 58, 59, 60, 62, 63]	[42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53]	1	11	12	5.56%
19	[0, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55]	2687	[0]	[64, 65, 66, 67, 56, 57, 58, 59, 60, 62, 63]	[44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55]	1	11	12	5.56%
20	[0, 42, 43, 44, 45, 47, 48, 49, 50, 51, 52]	2785	[0]	[64, 65, 66, 67, 56, 57, 58, 59, 60, 62, 63]	[42, 43, 44, 45, 47, 48, 49, 50, 51, 52]	1	11	10	11.11%
21	[0, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53]	2816	[0]	[64, 65, 66, 67, 56, 57, 58, 59, 60, 62, 63]	[42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53]	1	11	12	5.56%
22	[0, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53]	2821	[0]	[64, 65, 66, 67, 56, 57, 58, 59, 60, 62, 63]	[43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53]	1	11	11	8.33%
23	[0, 48, 49, 50, 51, 53, 54, 55, 56, 57, 58]	2888	[0, 57, 56, 58]	[64, 65, 66, 67, 59, 60, 62, 63]	[48, 49, 50, 51, 53, 54, 55]	4	8	7	36.11%
24	[0, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53]	2928	[0]	[64, 65, 66, 67, 56, 57, 58, 59, 60, 62, 63]	[44, 45, 46, 47, 48, 49, 50, 51, 52, 53]	1	11	10	11.11%
25	[0, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54]	2983	[0]	[64, 65, 66, 67, 56, 57, 58, 59, 60, 62, 63]	[41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54]	1	11	14	0.00%

LAMPIRAN D

Perbandingan Hasil Estimasi *Pitch* dengan *Dataset*

Tabel D-7 Hasil Perbandingan *Pitch* pada lagu Old McDonald Had a Farm di Model Kedua

No	<i>Pitch Result</i>	Dist	Match	Diff1	Diff2	Len_Match	Len_Diff1	Len_Diff2	Percentage
26	[0, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52]	3036	[0]	[64, 65, 66, 67, 56, 57, 58, 59, 60, 62, 63]	[41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52]	1	11	12	5.56%
27	[0, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52]	3146	[0]	[64, 65, 66, 67, 56, 57, 58, 59, 60, 62, 63]	[41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52]	1	11	12	5.56%
28	[0, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52]	3321	[0]	[64, 65, 66, 67, 56, 57, 58, 59, 60, 62, 63]	[42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52]	1	11	11	8.33%
29	[0, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54]	3363	[0]	[64, 65, 66, 67, 56, 57, 58, 59, 60, 62, 63]	[42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54]	1	11	13	2.78%
30	[0, 43, 44, 45, 46, 47, 48, 49, 51, 52, 53, 54]	3403	[0]	[64, 65, 66, 67, 56, 57, 58, 59, 60, 62, 63]	[43, 44, 45, 46, 47, 48, 49, 51, 52, 53, 54]	1	11	11	8.33%
31	[0, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51]	4093	[0]	[64, 65, 66, 67, 56, 57, 58, 59, 60, 62, 63]	[42, 43, 44, 45, 46, 47, 48, 49, 50, 51]	1	11	10	11.11%

Tabel D-8 menunjukkan perbandingan *pitch detected* dan *pitch* di *dataset* pada lagu Happy Birthday di model kedua. Sekuens *pitch* dari *dataset* berjumlah 12 data: [0, 56, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67].

Tabel D-8 Hasil Perbandingan *Pitch* pada lagu Happy Birthday di Model Kedua

No	<i>Pitch Result</i>	Dist	Match	Diff1	Diff2	Len_Match	Len_Diff1	Len_Diff2	Percentage
1	[0, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67]	147	[0, 65, 66, 64, 67, 58, 59, 60, 61, 62, 63]	[56]	[]	11	1	0	94.44%
2	[0, 56, 57, 58, 59, 60, 61, 63, 64, 65, 66]	294	[0, 65, 66, 64, 56, 58, 59, 60, 61, 63]	[67, 62]	[57]	10	2	1	86.11%
3	[0, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66]	813	[0, 65, 66, 64, 56, 58, 59, 60, 61, 62, 63]	[67]	[57, 53, 54, 55]	11	1	4	83.33%

LAMPIRAN D

Perbandingan Hasil Estimasi *Pitch* dengan *Dataset*

Tabel D-8 Hasil Perbandingan *Pitch* pada lagu Happy Birthday di Model Kedua

No	Pitch Result	Dist	Match	Diff1	Diff2	Len_Match	Len_Diff1	Len_Diff2	Percentage
4	[0, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67]	947	[0, 65, 66, 64, 67, 58, 59, 60, 61, 62, 63]	[56]	[]	11	1	0	94.44%
5	[0, 57, 58, 59, 60, 61, 62, 63, 64, 66, 67, 68]	1056	[0, 64, 66, 67, 58, 59, 60, 61, 62, 63]	[56, 65]	[57, 68]	10	2	2	83.33%
6	[0, 59, 60, 61, 62, 64, 65, 67, 68]	1081	[0, 65, 64, 67, 59, 60, 61, 62]	[56, 66, 58, 63]	[68]	8	4	1	75.00%
7	[0, 54, 55, 56, 57, 59, 60, 62, 67]	1329	[0, 67, 56, 59, 60, 62]	[64, 65, 66, 58, 61, 63]	[57, 54, 55]	6	6	3	58.33%
8	[0, 47, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 66, 67, 68]	1417	[0, 64, 66, 67, 56, 58, 59, 60, 61, 62, 63]	[65]	[68, 47, 54, 55, 57]	11	1	5	80.56%
9	[0, 53, 54, 55, 56, 57, 58, 59, 60]	1747	[0, 56, 58, 59, 60]	[64, 65, 66, 67, 61, 62, 63]	[57, 53, 54, 55]	5	7	4	50.00%
10	[0, 51, 52, 53, 54, 55, 56, 57, 58, 59]	1841	[0, 56, 59, 58]	[64, 65, 66, 67, 60, 61, 62, 63]	[51, 52, 53, 54, 55, 57]	4	8	6	38.89%
11	[0, 46, 47, 48, 49, 51, 52, 53, 54, 55, 56, 57, 58, 59]	2415	[0, 56, 59, 58]	[64, 65, 66, 67, 60, 61, 62, 63]	[46, 47, 48, 49, 51, 52, 53, 54, 55, 57]	4	8	10	27.78%
12	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60]	2433	[0, 56, 58, 59, 60]	[64, 65, 66, 67, 61, 62, 63]	[46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 57]	5	7	11	30.56%
13	[0, 48, 49, 50, 51, 53, 54, 56, 57]	2439	[0, 56]	[64, 65, 66, 67, 58, 59, 60, 61, 62, 63]	[48, 49, 50, 51, 53, 54, 57]	2	10	7	25.00%
14	[0, 48, 49, 50, 51, 52, 54, 55, 56]	2636	[0, 56]	[64, 65, 66, 67, 58, 59, 60, 61, 62, 63]	[48, 49, 50, 51, 52, 54, 55]	2	10	7	25.00%
15	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54]	2643	[0]	[64, 65, 66, 67, 56, 58, 59, 60, 61, 62, 63]	[46, 47, 48, 49, 50, 51, 52, 53, 54]	1	11	9	13.89%

LAMPIRAN D

Perbandingan Hasil Estimasi *Pitch* dengan *Dataset*

Tabel D-8 Hasil Perbandingan *Pitch* pada lagu Happy Birthday di Model Kedua

No	Pitch Result	Dist	Match	Diff1	Diff2	Len_Match	Len_Diff1	Len_Diff2	Percentage
16	[0, 45, 46, 47, 48, 50, 51, 52, 53, 54]	2650	[0]	[64, 65, 66, 67, 56, 58, 59, 60, 61, 62, 63]	[45, 46, 47, 48, 50, 51, 52, 53, 54]	1	11	9	13.89%
17	[0, 45, 46, 47, 48, 50, 51, 52, 53, 54, 55, 56, 57, 58]	2675	[0, 56, 58]	[64, 65, 66, 67, 59, 60, 61, 62, 63]	[45, 46, 47, 48, 50, 51, 52, 53, 54, 55, 57]	3	9	11	19.44%
18	[0, 46, 47, 48, 49, 50, 52, 53, 54, 55]	2753	[0]	[64, 65, 66, 67, 56, 58, 59, 60, 61, 62, 63]	[46, 47, 48, 49, 50, 52, 53, 54, 55]	1	11	9	13.89%
19	[0, 46, 47, 48, 49, 50, 52, 53, 54, 55]	2837	[0]	[64, 65, 66, 67, 56, 58, 59, 60, 61, 62, 63]	[46, 47, 48, 49, 50, 52, 53, 54, 55]	1	11	9	13.89%
20	[0, 44, 45, 46, 47, 48, 50, 51, 52, 53, 55, 56, 58]	2840	[0, 56, 58]	[64, 65, 66, 67, 59, 60, 61, 62, 63]	[44, 45, 46, 47, 48, 50, 51, 52, 53, 55]	3	9	10	22.22%
21	[0, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55]	2877	[0]	[64, 65, 66, 67, 56, 58, 59, 60, 61, 62, 63]	[44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55]	1	11	12	5.56%
22	[0, 44, 45, 46, 47, 49, 50, 51, 52, 54, 55, 56, 57, 58]	2905	[0, 56, 58]	[64, 65, 66, 67, 59, 60, 61, 62, 63]	[44, 45, 46, 47, 49, 50, 51, 52, 54, 55, 57]	3	9	11	19.44%
23	[0, 44, 45, 46, 47, 48, 49, 50, 51, 52, 54, 55, 56, 57, 58]	2929	[0, 56, 58]	[64, 65, 66, 67, 59, 60, 61, 62, 63]	[44, 45, 46, 47, 48, 49, 50, 51, 52, 54, 55, 57]	3	9	12	16.67%
24	[0, 44, 45, 46, 47, 49, 50, 51, 52, 53]	2995	[0]	[64, 65, 66, 67, 56, 58, 59, 60, 61, 62, 63]	[44, 45, 46, 47, 49, 50, 51, 52, 53]	1	11	9	13.89%
25	[0, 46, 47, 48, 49, 51, 52, 53, 54, 55, 56, 57, 58, 59]	3023	[0, 56, 59, 58]	[64, 65, 66, 67, 60, 61, 62, 63]	[46, 47, 48, 49, 51, 52, 53, 54, 55, 57]	4	8	10	27.78%
26	[0, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58]	3040	[0, 56, 58]	[64, 65, 66, 67, 59, 60, 61, 62, 63]	[43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 57]	3	9	14	11.11%

LAMPIRAN D

Perbandingan Hasil Estimasi *Pitch* dengan *Dataset*

Tabel D-8 Hasil Perbandingan *Pitch* pada lagu Happy Birthday di Model Kedua

No	<i>Pitch Result</i>	<i>Dist</i>	<i>Match</i>	<i>Diff1</i>	<i>Diff2</i>	<i>Len_Match</i>	<i>Len_Diff1</i>	<i>Len_Diff2</i>	<i>Percentage</i>
27	[0, 43, 44, 45, 46, 48, 49, 50, 51, 52, 54, 55, 56]	3122	[0, 56]	[64, 65, 66, 67, 58, 59, 60, 61, 62, 63]	[43, 44, 45, 46, 48, 49, 50, 51, 52, 54, 55]	2	10	11	13.89%
28	[0, 45, 46, 47, 48, 50, 51, 52, 53, 54]	3145	[0]	[64, 65, 66, 67, 56, 58, 59, 60, 61, 62, 63]	[45, 46, 47, 48, 50, 51, 52, 53, 54]	1	11	9	13.89%
29	[0, 45, 46, 47, 48, 49, 50, 51, 52]	3300	[0]	[64, 65, 66, 67, 56, 58, 59, 60, 61, 62, 63]	[45, 46, 47, 48, 49, 50, 51, 52]	1	11	8	16.67%
30	[0, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54]	3311	[0]	[64, 65, 66, 67, 56, 58, 59, 60, 61, 62, 63]	[44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54]	1	11	11	8.33%
31	[0, 43, 44, 45, 46, 48, 49, 50, 51, 54, 55]	3338	[0]	[64, 65, 66, 67, 56, 58, 59, 60, 61, 62, 63]	[43, 44, 45, 46, 48, 49, 50, 51, 54, 55]	1	11	10	11.11%
32	[0, 44, 45, 46, 47, 49, 50, 51, 52, 53, 54]	3364	[0]	[64, 65, 66, 67, 56, 58, 59, 60, 61, 62, 63]	[44, 45, 46, 47, 49, 50, 51, 52, 53, 54]	1	11	10	11.11%
33	[0, 41, 42, 43, 44, 45, 46, 47, 48, 49, 51, 52, 54, 55]	3753	[0]	[64, 65, 66, 67, 56, 58, 59, 60, 61, 62, 63]	[41, 42, 43, 44, 45, 46, 47, 48, 49, 51, 52, 54, 55]	1	11	13	2.78%
34	[0, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52]	3885	[0]	[64, 65, 66, 67, 56, 58, 59, 60, 61, 62, 63]	[43, 44, 45, 46, 47, 48, 49, 50, 51, 52]	1	11	10	11.11%

Tabel D-9 menunjukkan perbandingan *pitch detected* dan *pitch* di *dataset* pada lagu Brother John di model kedua. Sekuens *pitch* dari *dataset* berjumlah 13 data: [0, 47, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68].

Tabel D-9 Hasil Perbandingan *Pitch* pada lagu Brother John di Model Kedua

No	<i>Pitch Result</i>	<i>Dist</i>	<i>Match</i>	<i>Diff1</i>	<i>Diff2</i>	<i>Len_Match</i>	<i>Len_Diff1</i>	<i>Len_Diff2</i>	<i>Percentage</i>
1	[0, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68]	25	[0, 65, 66, 64, 67, 68, 58, 59, 60, 61, 62, 63]	[47]	[]	12	1	0	94.87%

LAMPIRAN D

Perbandingan Hasil Estimasi *Pitch* dengan *Dataset*

Tabel D-9 Hasil Perbandingan *Pitch* pada lagu Brother John di Model Kedua

No	<i>Pitch Result</i>	Dist	Match	Diff1	Diff2	Len_Match	Len_Diff1	Len_Diff2	Percentage
2	[0, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67]	450	[0, 65, 66, 64, 67, 58, 59, 60, 61, 62, 63]	[68, 47]	[57]	11	2	1	87.18%
3	[0, 58, 59, 61, 62, 63, 64, 66, 67, 68]	466	[0, 64, 66, 67, 68, 58, 59, 61, 62, 63]	[65, 60, 47]	[]	10	3	0	84.62%
4	[0, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67]	469	[0, 65, 66, 64, 67, 58, 59, 60, 61, 62, 63]	[68, 47]	[]	11	2	0	89.74%
5	[0, 57, 58, 59, 60, 61, 62, 63, 64, 65, 67]	537	[0, 65, 64, 67, 58, 59, 60, 61, 62, 63]	[66, 68, 47]	[57]	10	3	1	82.05%
6	[0, 49, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68]	607	[0, 65, 66, 64, 67, 68, 59, 60, 61, 62, 63]	[58, 47]	[49]	11	2	1	87.18%
7	[0, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70]	878	[0, 65, 66, 64, 67, 68, 59, 60, 61, 62, 63]	[58, 47]	[69, 70]	11	2	2	84.62%
8	[0, 48, 50, 51, 52, 54, 55, 56, 57, 58, 59, 60, 61, 64]	1649	[0, 64, 58, 59, 60, 61]	[65, 66, 67, 68, 47, 62, 63]	[48, 50, 51, 52, 54, 55, 56, 57]	6	7	8	43.59%
9	[0, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 60]	1693	[0, 58, 60, 47]	[64, 65, 66, 67, 68, 59, 61, 62, 63]	[45, 46, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57]	4	9	12	23.08%
10	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59]	1899	[0, 58, 59, 47]	[64, 65, 66, 67, 68, 60, 61, 62, 63]	[46, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57]	4	9	11	25.64%
11	[0, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57]	1994	[0, 47]	[64, 65, 66, 67, 68, 58, 59, 60, 61, 62, 63]	[45, 46, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57]	2	11	12	12.82%
12	[0, 50, 51, 52, 53, 54, 55, 56, 57, 58]	2013	[0, 58]	[64, 65, 66, 67, 68, 47, 59, 60, 61, 62, 63]	[50, 51, 52, 53, 54, 55, 56, 57]	2	11	8	23.08%

LAMPIRAN D

Perbandingan Hasil Estimasi *Pitch* dengan *Dataset*

Tabel D-9 Hasil Perbandingan *Pitch* pada lagu Brother John di Model Kedua

No	Pitch Result	Dist	Match	Diff1	Diff2	Len_Match	Len_Diff1	Len_Diff2	Percentage
13	[0, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57]	2234	[0, 47]	[64, 65, 66, 67, 68, 58, 59, 60, 61, 62, 63]	[45, 46, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57]	2	11	12	12.82%
14	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58]	2256	[0, 58, 47]	[64, 65, 66, 67, 68, 59, 60, 61, 62, 63]	[46, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57]	3	10	11	20.51%
15	[0, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69]	2285	[0, 65, 66, 64, 67, 68, 59, 60, 61, 62, 63]	[58, 47]	[69]	11	2	1	87.18%
16	[0, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 62]	2326	[0, 62, 47]	[64, 65, 66, 67, 68, 58, 59, 60, 61, 63]	[48, 49, 50, 51, 52, 53, 54, 55, 56, 57]	3	10	10	23.08%
17	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58]	2338	[0, 58, 47]	[64, 65, 66, 67, 68, 59, 60, 61, 62, 63]	[46, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57]	3	10	11	20.51%
18	[0, 47, 48, 49, 50, 51, 52, 53, 54, 55, 57, 58]	2415	[0, 58, 47]	[64, 65, 66, 67, 68, 59, 60, 61, 62, 63]	[48, 49, 50, 51, 52, 53, 54, 55, 57]	3	10	9	25.64%
19	[0, 42, 43, 44, 45, 46, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57]	2421	[0]	[64, 65, 66, 67, 68, 47, 58, 59, 60, 61, 62, 63]	[42, 43, 44, 45, 46, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57]	1	12	15	0.00%
20	[0, 48, 49, 50, 51, 52, 53, 54, 55, 56]	2468	[0]	[64, 65, 66, 67, 68, 47, 58, 59, 60, 61, 62, 63]	[48, 49, 50, 51, 52, 53, 54, 55, 56]	1	12	9	15.38%
21	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56]	2498	[0, 47]	[64, 65, 66, 67, 68, 58, 59, 60, 61, 62, 63]	[46, 48, 49, 50, 51, 52, 53, 54, 55, 56]	2	11	10	17.95%
22	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57]	2562	[0, 47]	[64, 65, 66, 67, 68, 58, 59, 60, 61, 62, 63]	[46, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57]	2	11	11	15.38%
23	[0, 49, 50, 51, 52, 53, 54, 55, 56, 57]	2583	[0]	[64, 65, 66, 67, 68, 47, 58, 59, 60, 61, 62, 63]	[49, 50, 51, 52, 53, 54, 55, 56, 57]	1	12	9	15.38%

LAMPIRAN D

Perbandingan Hasil Estimasi *Pitch* dengan *Dataset*

Tabel D-9 Hasil Perbandingan *Pitch* pada lagu Brother John di Model Kedua

No	Pitch Result	Dist	Match	Diff1	Diff2	Len_Match	Len_Diff1	Len_Diff2	Percentage
24	[0, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58]	2598	[0, 58, 47]	[64, 65, 66, 67, 68, 59, 60, 61, 62, 63]	[45, 46, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57]	3	10	12	17.95%
25	[0, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53]	2612	[0, 47]	[64, 65, 66, 67, 68, 58, 59, 60, 61, 62, 63]	[44, 45, 46, 48, 49, 50, 51, 52, 53]	2	11	9	20.51%
26	[0, 47, 48, 49, 50, 51, 52, 53, 54, 55]	2656	[0, 47]	[64, 65, 66, 67, 68, 58, 59, 60, 61, 62, 63]	[48, 49, 50, 51, 52, 53, 54, 55]	2	11	8	23.08%
27	[0, 45, 46, 47, 49, 50, 51, 52, 53, 54, 55]	2712	[0, 47]	[64, 65, 66, 67, 68, 58, 59, 60, 61, 62, 63]	[45, 46, 49, 50, 51, 52, 53, 54, 55]	2	11	9	20.51%
28	[0, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55]	2775	[0, 47]	[64, 65, 66, 67, 68, 58, 59, 60, 61, 62, 63]	[44, 45, 46, 48, 49, 50, 51, 52, 53, 54, 55]	2	11	11	15.38%
29	[0, 42, 43, 44, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56]	2794	[0, 47]	[64, 65, 66, 67, 68, 58, 59, 60, 61, 62, 63]	[42, 43, 44, 46, 48, 49, 50, 51, 52, 53, 54, 55, 56]	2	11	13	10.26%
30	[0, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56]	2804	[0, 47]	[64, 65, 66, 67, 68, 58, 59, 60, 61, 62, 63]	[46, 48, 49, 50, 51, 52, 53, 54, 55, 56]	2	11	10	17.95%
31	[0, 44, 45, 46, 47, 48, 49, 50, 51]	2839	[0, 47]	[64, 65, 66, 67, 68, 58, 59, 60, 61, 62, 63]	[44, 45, 46, 48, 49, 50, 51]	2	11	7	25.64%
32	[0, 43, 44, 46, 47, 48, 49, 50, 51, 52, 53]	3048	[0, 47]	[64, 65, 66, 67, 68, 58, 59, 60, 61, 62, 63]	[43, 44, 46, 48, 49, 50, 51, 52, 53]	2	11	9	20.51%
33	[0, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54]	3220	[0, 47]	[64, 65, 66, 67, 68, 58, 59, 60, 61, 62, 63]	[44, 45, 46, 48, 49, 50, 51, 52, 53, 54]	2	11	10	17.95%
34	[0, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50]	3545	[0, 47]	[64, 65, 66, 67, 68, 58, 59, 60, 61, 62, 63]	[41, 42, 43, 44, 45, 46, 48, 49, 50]	2	11	9	20.51%
35	[0, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57]	3738	[0, 47]	[64, 65, 66, 67, 68, 58, 59, 60, 61, 62, 63]	[43, 44, 45, 46, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57]	2	11	14	7.69%

LAMPIRAN D

Perbandingan Hasil Estimasi *Pitch* dengan *Dataset*

Tabel **D-10** menunjukkan perbandingan *pitch detected* dan *pitch* di *dataset* pada lagu London Bridge is Falling Down di model kedua. Sekuens *pitch* dari *dataset* berjumlah 10 data: [0, 56, 57, 58, 59, 60, 61, 62, 63, 64].

Tabel D-10 Hasil Perbandingan *Pitch* pada lagu London Bridge is Falling Down di Model Kedua

No	<i>Pitch Result</i>	<i>Dist</i>	<i>Match</i>	<i>Diff1</i>	<i>Diff2</i>	<i>Len_Match</i>	<i>Len_Diff1</i>	<i>Len_Diff2</i>	<i>Percentage</i>
1	[0, 56, 58, 59, 60, 61, 62, 64]	9	[0, 64, 56, 58, 59, 60, 61, 62]	[57, 63]	[]	8	2	0	86.67%
2	[0, 57, 58, 59, 60, 61, 62, 64]	621	[0, 64, 57, 58, 59, 60, 61, 62]	[56, 63]	[]	8	2	0	86.67%
3	[0, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64]	800	[0, 64, 56, 57, 58, 59, 60, 61, 62, 63]	[]	[55]	10	0	1	96.67%
4	[0, 53, 54, 56, 57, 58, 59, 60, 62]	884	[0, 56, 57, 58, 59, 60, 62]	[64, 61, 63]	[53, 54]	7	3	2	73.33%
5	[0, 53, 54, 55, 56, 57, 58, 59, 60]	1061	[0, 56, 57, 58, 59, 60]	[64, 61, 62, 63]	[53, 54, 55]	6	4	3	63.33%
6	[0, 55, 56, 58, 59, 60, 61, 62, 63]	1342	[0, 56, 58, 59, 60, 61, 62, 63]	[64, 57]	[55]	8	2	1	83.33%
7	[0, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61]	1365	[0, 56, 57, 58, 59, 60, 61]	[64, 62, 63]	[50, 51, 52, 53, 54, 55]	7	3	6	60.00%
8	[0, 48, 49, 50, 51, 52, 53, 54, 55, 57]	1539	[0, 57]	[64, 56, 58, 59, 60, 61, 62, 63]	[48, 49, 50, 51, 52, 53, 54, 55]	2	8	8	20.00%
9	[0, 53, 54, 55, 56, 57, 58, 59, 60]	1586	[0, 56, 57, 58, 59, 60]	[64, 61, 62, 63]	[53, 54, 55]	6	4	3	63.33%
10	[0, 50, 51, 52, 53, 54, 55, 56, 57, 58]	1734	[0, 57, 56, 58]	[64, 59, 60, 61, 62, 63]	[50, 51, 52, 53, 54, 55]	4	6	6	40.00%
11	[0, 50, 51, 52, 53, 54, 55, 56, 57, 58]	1797	[0, 57, 56, 58]	[64, 59, 60, 61, 62, 63]	[50, 51, 52, 53, 54, 55]	4	6	6	40.00%
12	[0, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58]	1812	[0, 57, 56, 58]	[64, 59, 60, 61, 62, 63]	[48, 49, 50, 51, 52, 53, 54, 55]	4	6	8	33.33%

LAMPIRAN D

Perbandingan Hasil Estimasi *Pitch* dengan *Dataset*

Tabel D-10 Hasil Perbandingan *Pitch* pada lagu London Bridge is Falling Down di Model Kedua

No	Pitch Result	Dist	Match	Diff1	Diff2	Len_Match	Len_Diff1	Len_Diff2	Percentage
13	[0, 55, 56, 57, 58, 59, 60, 61, 62, 64]	1860	[0, 64, 56, 57, 58, 59, 60, 61, 62]	[63]	[55]	9	1	1	90.00%
14	[0, 48, 49, 51, 52, 53, 54, 55, 56, 57]	1896	[0, 57, 56]	[64, 58, 59, 60, 61, 62, 63]	[48, 49, 51, 52, 53, 54, 55]	3	7	7	30.00%
15	[0, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60]	2072	[0, 56, 57, 58, 59, 60]	[64, 61, 62, 63]	[51, 52, 53, 54, 55]	6	4	5	56.67%
16	[0, 48, 49, 50, 51, 52, 53, 54, 55, 56]	2085	[0, 56]	[64, 57, 58, 59, 60, 61, 62, 63]	[48, 49, 50, 51, 52, 53, 54, 55]	2	8	8	20.00%
17	[0, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56]	2095	[0, 56]	[64, 57, 58, 59, 60, 61, 62, 63]	[45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55]	2	8	11	10.00%
18	[0, 45, 46, 48, 49, 50, 51, 52, 53, 54]	2148	[0]	[64, 56, 57, 58, 59, 60, 61, 62, 63]	[45, 46, 48, 49, 50, 51, 52, 53, 54]	1	9	9	10.00%
19	[0, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57]	2172	[0, 57, 56]	[64, 58, 59, 60, 61, 62, 63]	[48, 49, 50, 51, 52, 53, 54, 55]	3	7	8	26.67%
20	[0, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55]	2369	[0]	[64, 56, 57, 58, 59, 60, 61, 62, 63]	[45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55]	1	9	10	6.67%
21	[0, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53]	2370	[0]	[64, 56, 57, 58, 59, 60, 61, 62, 63]	[44, 45, 46, 47, 48, 49, 50, 51, 52, 53]	1	9	10	6.67%
22	[0, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57]	2421	[0, 57, 56]	[64, 58, 59, 60, 61, 62, 63]	[48, 49, 50, 51, 52, 53, 54, 55]	3	7	8	26.67%
23	[0, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53]	2568	[0]	[64, 56, 57, 58, 59, 60, 61, 62, 63]	[44, 45, 46, 47, 48, 49, 50, 51, 52, 53]	1	9	10	6.67%
24	[0, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57]	2580	[0, 57, 56]	[64, 58, 59, 60, 61, 62, 63]	[47, 48, 49, 50, 51, 52, 53, 54, 55]	3	7	9	23.33%
25	[0, 44, 45, 46, 48, 49, 50, 51, 52, 53]	2660	[0]	[64, 56, 57, 58, 59, 60, 61, 62, 63]	[44, 45, 46, 48, 49, 50, 51, 52, 53]	1	9	9	10.00%

LAMPIRAN D

Perbandingan Hasil Estimasi *Pitch* dengan *Dataset*

Tabel D-10 Hasil Perbandingan *Pitch* pada lagu London Bridge is Falling Down di Model Kedua

No	<i>Pitch Result</i>	<i>Dist</i>	<i>Match</i>	<i>Diff1</i>	<i>Diff2</i>	<i>Len_Match</i>	<i>Len_Diff1</i>	<i>Len_Diff2</i>	<i>Percentage</i>
26	[0, 45, 46, 47, 48, 49, 50, 51, 52, 53]	2671	[0]	[64, 56, 57, 58, 59, 60, 61, 62, 63]	[45, 46, 47, 48, 49, 50, 51, 52, 53]	1	9	9	10.00%
27	[0, 45, 46, 47, 48, 49, 50, 51, 52, 53]	3284	[0]	[64, 56, 57, 58, 59, 60, 61, 62, 63]	[45, 46, 47, 48, 49, 50, 51, 52, 53]	1	9	9	10.00%
28	[0, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56]	3673	[0, 56]	[64, 57, 58, 59, 60, 61, 62, 63]	[45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55]	2	8	11	10.00%
29	[0, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56]	3952	[0, 56]	[64, 57, 58, 59, 60, 61, 62, 63]	[47, 48, 49, 50, 51, 52, 53, 54, 55]	2	8	9	16.67%